AUTOMATYKA • 2009 • Tom 13 • Zeszyt 2

Bogusław Filipowicz*, Wojciech Chmiel*, Piotr Kadłuczka*

Ukierunkowane przeszukiwanie przestrzeni rozwiązań w algorytmach rojowych

1. Wprowadzenie

Algorytmy rojowe, w tym przedstawiony w artykule algorytm pszczeli (*BA – Bees Algorithm*) należą do szerokiej klasy algorytmów optymalizacyjnych, które swoją inspirację czerpią z natury ([5, 8]). Rozwiązania jakie występują w świecie roślin i zwierząt z powodzeniem zastosowano w wielu dziedzinach działalności człowieka. Także w dziedzinie optymalizacji powstały metody naśladujące naturę:

- mechanizm ewolucji naturalnej – algorytmy ewolucyjne, genetyczne (*EA*, *GA – Evolution*, *Genetic Algorithm*);
- mechanizm ewolucji kulturowej – algorytmy memetyczne (*MA – Memetic Algorithm*);
- zasady organizacji koloni owadów – algorytmy rojowe (*SOA – Swarm-based Optimization Algorithm*), których przedstawicielem są: algorytmy mrówkowe (*AA – Ant Algorithm*, *ACO – Ant Colony Optimization*) oraz algorytmy pszczele;
- mechanizm działania układu odpornościowego – sztuczne systemy immunologiczne (*AIS – Artificial Immune Systems*);
- sposób organizacji komórek neuronowych – sieci neuronowe (*NN – Neural Networks*).

Powyższe metody nie operują na pojedynczym rozwiązaniu bieżącym, jak klasyczne metody optymalizacji, ale na zbiorze rozwiązań. Cecha ta pozwala na równoczesne ich przetwarzanie (implementacja równoległa) i wykorzystanie dużej mocy obliczeniowej i zasobów pamięci współczesnych komputerów lub sieci komputerowych (przetwarzanie rozproszone). Metody te są dedykowane do rozwiązywania zagadnień o dużej złożoności obliczeniowej, a trudność implementacji praktycznie nie zależy od stopnia komplikacji problemu.

1.1. Algorytm pszczeli

Pszczoły zbierające pyłek kwiatowy operują na znacznym obszarze. Odległość, na jaką docierają owady, często przekracza 10 km ([5, 8]). Jak udaje się im na tak dużym terytorium

* Katedra Automatyki, Akademia Górniczo-Hutnicza w Krakowie

odnaleźć kwiaty zasobne w pyłek oraz kolektywnie organizować eksplorację, przy dynamicznie zmieniających się warunkach? W celu przeprowadzenia rozpoznania rój wysyła „zwiadowców", którzy przemierzając w sposób losowy terytorium, poszukują kwiatów – zbierając informacje o ilości, jakości pyłku oraz odległości od ula. Po powrocie przekazują swoją wiedzę pozostałym pszczołom, wykonując specjalny taniec, będący formą komunikowania się. Kierunek wykonywania tańca obrazuje kierunek, w którym znajduje się rozpoznany obszar, czas trwania tańca – odległość, częstotliwość tańca – zasobność obszaru. W zależności od jakości obszaru rój proporcjonalnie dzieli się, odwiedzając liczniej obszary bardziej zasobne. Wracające z pyłkiem do ula pszczoły przekazują następnym informacje o aktualnym stanie obszarów, regulując w ten sposób liczbę wylatujących tam osobników.

Sposób zachowania się roju pszczelego stanowi inspirację w kilku elementach algorytmu. Zbiór rozwiązań, na którym operuje algorytm, oraz równoległe przeszukiwanie przestrzeni rozwiązań odzwierciedla liczność roju pszczelego. Losowe inicjowanie rozwiązań początkowych jest odpowiednikiem przeszukiwania terytorium przez pszczoły-zwiadowców. Wykorzystywana informacja dotycząca funkcji celu uzyskanych rozwiązań i innych parametrów charakteryzujących przestrzeń rozwiązań, która implikuje osiągnięcie kolejnych rozwiązań w algorytmie, odpowiada wiedzy przekazywanej przez pszczoły tańcem innym osobnikom. Sposób podziału roju pomiędzy obszary i jego dalsza eksploracja, definiują w pojęciach algorytmów ewolucyjnych: mechanizm selekcji proporcjonalnej, operatory (otoczenie) i sposób przetwarzania populacji rozwiązań.

Schemat podstawowy algorytmu *BA*:

Krok 1. Utworzenie populacji n rozwiązań początkowych
- obliczenie wartości funkcji celu dla rozwiązań początkowych.

Krok 2. Selekcja:
- wybór m przeszukiwanych sąsiedztw,
- określenie wielkości przeszukiwanych sąsiedztw,
- określenie liczności rozwiązań w sąsiedztwie.

Krok 3. Utworzenie nowych rozwiązań:
- dla każdego z m sąsiedztw – zgodnie z zadaną wielkością i licznością,
- obliczenie wartości funkcji celu dla utworzonych rozwiązań.

Krok 4. Utworzenie nowej populacji rozwiązań przez:
- wybór najlepszego rozwiązania, dla każdego z m sąsiedztw,
- uzupełnienie brakujących (n–m) rozwiązań.

Krok 5. Sprawdzenie warunku zakończenia obliczeń:
- rozwiązanie suboptymalne (gdy STOP),
- idź do kroku 2 (w przypadku niespełnienia warunku).

1.2. Ukierunkowanie przeszukiwania przestrzeni rozwiązań za pomocą warunkowej wartości oczekiwanej funkcji celu

Ukierunkowanie przeszukiwania przestrzeni rozwiązań w algorytmach przybliżonych jest realizowanie na podstawie wartości funkcji celu kolejno uzyskiwanych rozwiązań.

Wprowadzenie warunkowej wartości oczekiwanej funkcji celu pozwala na ocenę jakości rozwiązań częściowo ustalonych i jest sposobem szerszego (statystycznego) spojrzenia na pewną podprzestrzeń, przestrzeni rozwiązań. Zastosowanie w konstrukcji algorytmów wartości oczekiwanej, jako dodatkowego elementu będącego źródłem informacji o własnościach przestrzeni i perspektywach przeszukiwania, jest możliwe w większości metod przybliżonych. W algorytmie poszukiwania z zabronieniami (*TS*) [3] – w mechanizmie zabronień, w algorytmach ewolucyjnych (*EA*) [7] – w konstrukcji operatorów pseudogenetycznych, algorytmie symulowanego wyżarzania (*SA*) [2] – w mechanizmie akceptacji rozwiązań pogarszających wartość funkcji celu.

Ukierunkowanie przeszukiwania przestrzeni musi pogodzić dwa sprzeczne podejścia: swobodne poszukiwanie optimum globalnego w całej przestrzeni rozwiązań (eksploracja) oraz możliwie dokładne przeszukiwanie otoczenia optimum lokalnego (eksploatacja). Kompromis pozwalający podążać w stronę optimum lokalnego, a następnie opuścić jego obszar przyciągania realizowany jest za pomocą przeciwstawnych mechanizmów: intensyfikacji i różnicowania.

W celu poprawy efektywności algorytmów populacyjnych należy stworzyć odpowiednie proporcje między naporem selekcyjnym a różnorodnością populacji. Zwiększenie naporu selekcyjnego zmniejsza różnorodność populacji, co prowadzi na ogół do przedwczesnej zbieżności algorytmu. Zastosowanie warunkowej wartości oczekiwanej może być propozycją realizacji wspomnianego kompromisu.

2. Kwadratowe zagadnienie przydziału

Rozpatrywane kwadratowe zagadnienie przydziału – *QAP* (*Quadratic Assignment Problem*), występuje w literaturze także pod innymi nazwami – jako problem przydziału z kwadratowym wskaźnikiem jakości czy też z kwadratową funkcją celu. Należy ono do klasy zagadnień *NP-trudnych*, co wymusza stosowanie do jego rozwiązania metod przybliżonych już dla zadań o niewielkim rozmiarze (powyżej 30). Mimo że jest ono znacznie trudniejsze od innych zagadnienia optymalizacji kombinatorycznej, to cieszy się powszechnym zainteresowaniem, gdyż modeluje ważną klasę problemów decyzyjnych, dla których rozwiązania można przedstawić w postaci permutacji ([1, 3, 6]).

2.1. Model matematyczny zagadnienia *QAP*

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.

2.2. Warunkowa wartość oczekiwana funkcji celu rozwiązań częściowo ustalonych

Warunkowa wartość oczekiwana funkcji celu pozwala na ocenę jakości rozwiązań częściowo ustalonych. Poniższy wzór (2) został wyprowadzony dla postaci permutacyjnej rozwiązania zagadnienia *QAP* [3].

Określmy zbiór liczb naturalnych z zakresu od 1 do n jako $L = [1, ..., n]$, zbiór $M = [1, ..., n] \setminus \{c(s_1), ..., c(s_i), ..., c(s_k)\}$ (numery nieprzydzielonych obiektów, gdzie $c(s_i)$ – jest numerem obiektu przydzielonym do pozycji s_i) oraz zbiór $H = \{s_1, ..., s_i, ..., s_k\}$ (numery zajętych pozycji). W takim przypadku warunkowa wartość oczekiwana funkcji celu dla problemu *QAP* przy k ustalonych pozycjach, wyniesie:

$$\begin{aligned} E(\phi / \pi(s_1) = c(s_1),...,\pi(s_i) = c(s_i),...,\pi(s_k) = c(s_k)) = \sum_{i\in H} \sum_{j\in H} f_{ij} d_{\pi(i)\pi(j)} + \\ + \frac{1}{n-k} \sum_{i\in H} \sum_{j\in L\setminus H} f_{ij} \sum_{m\in M} d_{\pi(i)m} + \frac{1}{n-k} \sum_{i\in L\setminus H} \sum_{j\in H} f_{ij} \sum_{m\in M} d_{m\pi(j)} + \\ + \frac{1}{n-k} \sum_{i\in L\setminus H} f_{ii} \sum_{m\in M} d_{mm} + \frac{1}{n-k} \frac{1}{n-k-1} \sum \{f_{ij} : j \neq i \in L \setminus H\} \sum \{d_{ij} : j \neq i \in M\} \end{aligned} \tag{2}$$

3. Implementacja algorytmu

Ogólny schemat algorytmu *BA* pozostawia wiele swobody przy jego implementacji. Zastosowane szczegółowe rozwiązania dotyczące:

- sposobu utworzenia populacji początkowej,
- zastosowanego mechanizmu selekcji,
- definicji przeszukiwanego sąsiedztwa (wielkość, liczność, przegląd pełny lub częściowy),
- sposobu utworzenia nowej populacji,
- warunku zakończenia obliczeń,

oraz wartości kluczowych parametrów algorytmu decydują o jego efektywności.

W celu oceny możliwości skutecznego ukierunkowania przeszukiwania przestrzeni rozwiązań, przy wykorzystaniu warunkowej wartości oczekiwanej funkcji celu rozwiązań częściowo ustalonych, dla algorytmu *BA* zaimplementowano dwie jego wersje. W pierwszej stosuje się wartość funkcji celu jako klasyczny i jedyny sposób oceny perspektyw dalszego przetwarzania rozwiązań, a w drugiej wersji funkcję oceny rozwiązań częściowo zastąpiono wartością oczekiwaną dla losowo wybranej maski pozycji ustalonych rozwiązań. Implementowany klasyczny algorytm *BA* wygląda następująco:

Krok 1. Utworzenie populacji λ rozwiązań początkowych:
- losowa generacja λ permutacji,
- obliczenie wartości funkcji celu dla rozwiązań początkowych,
- posortowanie rozwiązań populacji według wartości funkcji celu,
- inicjujemy π_{best} – jako najlepsze, dotychczas znalezione rozwiązanie populacji.

Krok 2. Dla m najlepszych rozwiązań populacji:
- definiujemy sąsiedztwo rozwiązania $\pi = (\pi_1, \pi_2, ..., \pi_i, ..., \pi_j, ..., \pi_n)$: $S(\pi) = \{\pi' : \pi' = (\pi_1, \pi_2, ...\pi_j, ..., \pi_i, , ..., \pi_n),\ i, j = 1, ..., n \ \wedge\ i \neq j\}$
- znajdujemy najlepsze rozwiązanie w sąsiedztwie: $\pi'' = \arg\ \max\{\phi(\pi) : \pi' \in S(\pi), \pi' \neq \pi\}$.

Krok 3. Utworzenie nowej populacji rozwiązań:
- m najlepszych rozwiązań (po jednym każdego z sąsiedztwa),
- uzupełnienie brakujących (λ–m) rozwiązań, losowo wybranymi rozwiązaniami z poprzedniej populacji,
- posortowanie rozwiązań populacji według wartości funkcji celu.

Krok 4. Sprawdzenie, czy nie nastąpiła poprawa π_{best} (porównanie z najlepszym rozwiązaniem populacji i ewentualne podstawienie).

Krok 5. Sprawdzenie warunku zakończenia obliczeń (zadana liczba I_{max} iteracji):
- zwracamy π_{best} – rozwiązanie suboptymalne (gdy STOP),
- idziemy do kroku 2 (w przypadku niespełnienia warunku).

Parametrami algorytmu są:

λ – rozmiar populacji,
m – liczba przeglądanych sąsiedztw,
I_{max} – zadana liczba iteracji.

Wersja algorytmu *BA-EX*, stosująca wartość oczekiwaną rozwiązań częściowo ustalonych – $E(\phi(\pi)/_{i \in D})$, gdzie D określa zbiór ustalonych pozycji, różni się krokiem 2.:

Krok 2. W celu utworzenia m nowych rozwiązań populacji:
- generujemy losowo maskę d ustalonych pozycji rozwiązania (definiujemy zbiór D),
- dla wylosowanej maski obliczamy $E(\phi(\pi)/_{i \in D})$, dla wszystkich rozwiązań populacji,

- sortujemy rozwiązania w populacji wg $E(\phi(\pi)/_{i\in D})$,
- wybieramy m rozwiązań populacji o najmniejszej wartości oczekiwanej,
- definiujemy sąsiedztwo rozwiązania $\pi = (\pi_1, \pi_2, ..., \pi_i, ..., \pi_j, ..., \pi_n)$:
  $S(\pi) = \{\pi' : \pi' = (\pi_1, \pi_2, ...\pi_j, ..., \pi_i, , ..., \pi_n),\ i, j = 1, ..., n \ \wedge\ i \neq j \ \wedge\ i, j \notin D\}$,
- znajdujemy najlepsze rozwiązanie w sąsiedztwie:
  $\pi'' = \arg\ \max\{\phi(\pi) : \pi' \in S(\pi), \pi' \neq \pi\}$.

Dodatkowym parametrem algorytmu jest d – liczba pozycji ustalonych rozwiązania. Obie wersje algorytmu zaimplementowano w języku C# na platformie .NET.

4. Wyniki eksperymentów obliczeniowych

Na podstawie opracowanych algorytmów wykonano eksperymentalne oprogramowanie dla zagadnienia *QAP*. Testy przeprowadzono dla 39 zadań testowych o rozmiarze n = 22–64, zaczerpniętych z biblioteki *QAPLIB-A* [1]. Biblioteka zawiera instancje testowe opisujące rzeczywiste i wygenerowane zadania kwadratowego zagadnienia przydziału, stworzone w 1991 r. (i stale rozwijane) przez: R. Burkarda, S. Karischa i F. Rendla.

Główny nacisk w eksperymentach obliczeniowych położono na porównanie efektywności algorytmów *BA* oraz *BA-EX*, stosującego warunkową wartość oczekiwaną funkcji celu. Dla wersji *BA-EX*, wykorzystującej losową maskę pozycji ustalonych, istotnym okazał się problem wielkości tej maski – stąd przebadano wpływ d – liczby pozycji ustalonych rozwiązania. Z racji zróżnicowania rozmiaru zadań parametr ten wyrażono w [%], jako stosunek liczby pozycji ustalonych do rozmiaru zadania. Pozostałe nie zmieniane parametry algorytmu wynoszą: I_{max} = 500 iteracji, λ = 100 (rozmiar populacji), m = 5 (liczba sąsiedztw). W tabeli 1 przedstawiono wyniki badań eksperymentalnych zagadnienia *QAP*.

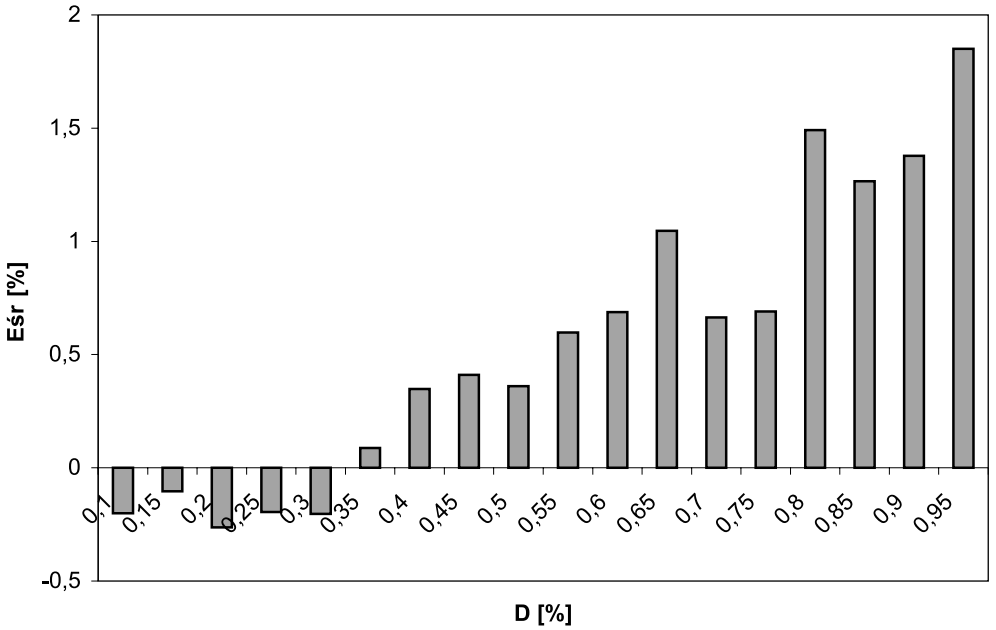
W tabeli 1 przyjęto następujące oznaczenia:

- ϕ_{BA} – wartość funkcji przystosowania najlepszego znalezionego rozwiązania dla algorytmu *BA*,
- $\phi_{BA\text{-}EX}$ – wartość funkcji przystosowania najlepszego znalezionego rozwiązania dla algorytmu *BA-EX*,
- E = 100%*($\phi_{BA\text{-}EX} - \phi_{BA}$)/$\phi_{BA}$ – względna [%] różnica funkcji celu najlepszego rozwiązania dla algorytmu *BA* w stosunku do *BA-EX*,
- I_{best} – numer iteracji w której uzyskano najlepszą wartość funkcji przystosowania,
- $E_{śr}$ – średnia [%] wartość błędu E dla kolejnych serii eksperymentów.

Rysunek 1 zawiera wyniki eksperymentów komputerowych dla różnych wielkości maski D (każda wartość przedstawiona na wykresie jest średnią z dziesięciu przebiegów dla ustalonej wielkości maski).

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

			D=20%			D=80%		
Nazwa	ϕ_{BA}	I_{best}	$\phi_{BA\text{-}EX}$	I_{best}	E [%]	$\phi_{BA\text{-}EX}$	I_{best}	E [%]
BUR26A	5436918	17	5435723	34	-0,02	5434234	361	-0,05
BUR26B	3827840	17	3825228	34	-0,07	3834281	490	0,17
BUR26C	5443738	23	5427308	160	-0,30	5427227	223	-0,30
BUR26D	3828852	21	3832582	28	0,10	3823195	449	-0,15
BUR26E	5389340	23	5387837	249	-0,03	5396997	224	0,14
BUR26F	3782051	16	3782044	137	0,00	3782747	178	0,02
BUR26G	10124306	17	10118634	104	-0,06	10138385	184	0,14
BUR26H	7098658	19	7099875	31	0,02	7102958	348	0,06
CHR22A	6832	14	6474	168	-5,24	7070	45	3,48
CHR22B	6858	10	6498	50	-5,25	6956	450	1,43
CHR25A	5218	14	4900	357	-6,09	6090	351	16,71
ESC32A	152	14	136	39	-10,53	158	492	3,95
ESC32B	200	12	192	168	-4,00	208	54	4,00
ESC32C	642	8	642	8	0,00	642	22	0,00
ESC32D	200	12	200	8	0,00	208	155	4,00
ESC32E	2	0	2	0	0,00	2	0	0,00
ESC32F	2	1	2	1	0,00	2	1	0,00
ESC32G	6	0	6	1	0,00	6	2	0,00
ESC32H	442	8	440	89	-0,45	442	104	0,00
ESC64A	116	9	116	11	0,00	116	66	0,00
KRA30A	93360	18	93540	22	0,19	97460	451	4,39
KRA30B	97490	15	96850	28	-0,66	100480	298	3,07
LIPA30A	13430	16	13406	15	-0,18	13386	252	-0,33
LIPA30B	176610	16	174829	36	-1,01	175041	341	-0,89
LIPA40A	32003	21	31969	210	-0,11	31953	488	-0,16
LIPA40B	564495	28	562076	301	-0,43	561346	389	-0,56
LIPA50A	62897	27	62804	38	-0,15	62803	455	-0,15
LIPA50B	1434253	27	1424316	497	-0,69	1427682	494	-0,46
LIPA60A	108306	43	108308	455	0,00	108336	452	0,03
LIPA60B	3001326	40	3014080	184	0,42	2993828	496	-0,25
SKO42	16178	35	16114	410	-0,40	16636	184	2,83
SKO49	23662	43	24050	243	1,64	24568	358	3,83
SKO56	34842	51	35556	87	2,05	35804	334	2,76
STE36A	10102	24	10086	153	-0,16	11462	378	13,46
STE36B	18024	27	19024	349	5,55	20016	390	11,05
STE36C	8468504	26	8687268	467	2,58	9537950	178	12,63
THO30	159112	16	153742	52	-3,37	157214	461	-1,19
THO40	247282	38	249698	342	0,98	248974	484	0,68
WIL50	49890	45	49286	464	-1,21	49590	464	-0,60
$E_{śr}$[%]					-0,69			2,15



Rys. 1. Średnia waroś błędu $E_{śr}$ (dla dziesięciu przebiegów testowych)

4.1. Wnioski

Zamieszczone w tabeli 1 oraz na rysunku 1 wyniki pozwalają zauważyć istotny wpływ liczby ustalonych pozycji rozwiązania (wielkość losowej maski) na jakość uzyskiwanych rozwiązań, dla wersji algorytmu *BA-EX*, stosującej warunkową wartość oczekiwaną. Najlepsze wyniki uzyskiwano dla maski obejmującej niewielką liczbę pozycji – ok. 10–20%. Ustalenie znacznej części rozwiązania (powyżej 50%), przylosowym wyborze pozycji, daje dużo mniejsze prawdopodobieństwo poprawy funkcji celu.

Porównanie uzyskanych wyników z wersją podstawową algorytmu *BA*, wskazuje na istotną poprawę efektywności algorytmu przez wprowadzenie sposobu ukierunkowania wartością oczekiwaną. Jest ona uzyskana, mimo że algorytm z częściowo ustalonym rozwiązaniem przegląda sąsiedztwo o mniejszej liczności. Wyznaczenie wartości oczekiwanych wymaga natomiast dodatkowego nakładu obliczeniowego. Ponadto, uzyskane wyniki wskazują, że wprowadzenie maski wyraźnie zmniejsza zbieżność algorytmu.

Obszarami badań rokującymi dalszą poprawę jakości uzyskiwanych wyników są zagadnienia związane z:

- doborem pozycji maski (nielosowym),
- rozpiętością ustalonych pozycji maski (analogicznie do rozpiętości schematu),
- innymi sposobami definiowania sąsiedztwa (wielkość zależna od jakości rozwiązania).

Literatura

[1] Burkard R.E., Karisch S.E., Rendl F., *QAPLIB-A Quadratic Assignment Problem Library*. European Journal of Operational Research, 55, 1991, 115–119.

[2] Chmiel W., Kadłuczka P., Jedrusik S., *Efektywność algorytmu ewolucyjnego wykorzystującego mechanizm różnicowania*. Zeszyty Naukowe Politechniki Śląskiej, seria: AUTOMATYKA, zeszyt 143, 33–44, Wydawnictwo Naukowo-Dydaktyczne Politechniki Śląskiej, Gliwice 2006.

[3] Chmiel W., *Algorytmy ewolucyjne w optymalizacji przydziału zadań z kwadratową funkcją celu*. AGH, Kraków 2004.

[4] Eberhart R., Shi Y., Kenedy J., *Swarm Intelligence*. Morgan Kaufman, San Francisco 2001.

[5] Filipowicz B., *„Algorytm pszczeli", praca zespołowa pod kierunkiem prof. B. Filipowicza*. LBOiS, Katedra Automatyki AGH, Kraków 2008 (niepublikowana).

[6] Filipowicz B., Wala K., *Algorytmy optymalizacji kwadratowego zagadnienia przydziału*. Kwartalnik Elektrotechnika, z. 1, Wydawnictwo AGH, Kraków 1992.

[7] Kadłuczka P., Chmiel W., *Zastosowanie własności zagadnienia QAP w konstrukcji algorytmów ewolucyjnych*. AGH, Kraków 2004, 112–120.

[8] Pham D.T., Ghanbarzadeh A., Koc E., Otri S., Rahim S., Zaidi M., *The Bees Algorithm – A Novel Tool for Complex Optimisation Problems.* Technical Note, Manufacturing Engineering Centre, Cardiff University, UK, 2005.