ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2006
Seria: AUTOMATYKA z. 143 Nr kol. 1726

Wojciech CHMIEL, Piotr KADŁUCZKA, Stanisław JĘDRUSIK
Akademia Górniczo-Hutnicza

EFEKTYWNOŚĆ ALGORYTMU EWOLUCYJNEGO WYKORZYSTUJĄCEGO MECHANIZM RÓŻNICOWANIA

Streszczenie. Artykuł prezentuje wyniki prac związanych z implementacją i badaniem efektywności algorytmu ewolucyjnego, wykorzystującego operatory różnicujące. Bazują one na warunkowej wartości oczekiwanej funkcji celu rozwiązań częściowo ustalonych. Badania testowe wykonano dla standardowych zadań testowych kwadratowego zagadnienia przydziału (*QAP*).

EFFICIENCY OF THE EVOLUTIONARY ALGORITHM USING THE DIVERSIFICATION MECHANISM

Summary. The paper presents our work on implementation and evaluation of evolutionary algorithm using diversification operators. They are based on expected conditional value of objective function for partially fixed solutions. The experiments were performed for standard test problems of quadratic assignment problems (*QAP*).

1. Zapobieganie przedwczesnej zbieżności algorytmu ewolucyjnego

Powszechny problem przedwczesnej zbieżności algorytmu ewolucyjnego [4],[9] jest konsekwencją stosowanego w algorytmie mechanizmu selekcji oraz sposobu przetwarzania populacji (zjawisko *naporu selekcyjnego*, wzgl. *nacisku selektywnego*). 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* a *różnorodnością populacji*. Zwiększenie naporu selekcyjnego zmniejsza różnorodność populacji, co prowadzi na ogół do przedwczesnej zbieżności algorytmu.

W algorytmie należy pogodzić więc 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*. Precyzyjne wskazanie miejsca występowania tych

mechanizmów w algorytmie ewolucyjnym jest kłopotliwe, ponieważ odpowiada za nie sposób implementacji wielu elementów wchodzących w skład algorytmu. Podstawowe znaczenie ma tutaj rodzaj stosowanej *selekcji* oraz sam schemat algorytmu. Elementy te odpowiadają za wystąpienie tendencji do poprawy (mniej lub bardziej zdecydowanej) średniej wartości funkcji przystosowania w populacji. Selekcja jest realizowana w dwóch etapach: wyboru rozwiązań rodzicielskich (*reprodukcja*) poddawanych następnie krzyżowaniu i mutacji oraz utworzeniu nowej populacji (*sukcesja*) na podstawie starej populacji i powstałych rozwiązań potomnych.

Intensyfikującymi będą te metody selekcji (reprodukcji), które w bardziej zdecydowany sposób preferują rozwiązania o lepszej wartości funkcji przystosowania. Należą do nich (wymienione malejąco):

- metoda proporcjonalna – „koła ruletki",
- metoda oparta na rankingu,
- metoda turniejowa,
- metoda progowa.

Jako przykład typowego mechanizmu różnicującego należy wymienić schemat przetwarzania, w którym populacja potomna zastępuje rodzicielską (sukcesja z całkowitym zastępowaniem – ang. *generational replacement*). Zastępowanie częściowe (np. ochrona elity) czy wielokrotne powtórzenia rozwiązań w populacji stanowią osłabienie mechanizmu dywersyfikacji i wzmocnienie naporu selekcyjnego. Klasyfikacja strategii selekcji ze względu na jej pewne własności [7] daje pogląd co do możliwości wykorzystania przeciwstawnych podejść, wpływających na zbieżność algorytmu. Mimo szerokiej gamy powszechnie znanych możliwości sterowania zbieżnością algorytmu za pomocą selekcji, podejście to nie daje w pełni zadawalających wyników i nadal poszukiwane są alternatywne sposoby rozwiązania tego problemu. Z racji ścisłego powiązania mechanizmu selekcji z zastosowanym podejściem do tworzenia pokolenia oraz stosowanymi operatorami genetycznymi, jedynie łączne rozpatrzenie tych elementów pozwala obiektywnie spojrzeć na mechanizm selekcji. Rozważając algorytm klasy *Steady-State*, tylko niektóre osobniki populacji zostają zastąpione w kolejnym pokoleniu. Uzależniając tę liczbę 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. Zadanie selekcji – stworzenie *naporu selekcyjnego*, dokonuje się tu przez zwielokrotnienie szansy wyboru rozwiązania lepszego – utrzymującego się w populacji przez większą liczbę pokoleń. Zadbanie o *różnorodność populacji* skutkuje zabronieniem wstawienia do populacji duplikatu już istniejącego rozwiązania. Przy tak prosto realizowanej selekcji sterowanie zbieżnością optymalizacji przesuwa się z selekcji na *zbiór operatorów, organizację populacji i inne mechanizmy hybrydowe*.

Realizacja mechanizmu intensyfikacji i różnicowania za pomocą operatorów genetycznych (pseudogenetycznych) jest drugim, ważnym obszarem badań. Intensyfikacja, częściej stosowana, korzysta na ogół ze znanych, bogatych dokonań metod przeszukiwań lokalnych włączonych do algorytmu jako dodatkowy operator lub element konstrukcji algorytmu hybrydowego. Różnicowanie natomiast jest specyfiką algorytmu ewolucyjnego i nie posiada gotowych, wypracowanych rozwiązań, które można by wprost zapożyczyć. Dlatego w przedstawionych badaniach położono

szczególny nacisk na mechanizm różnicowania realizowany przez operatory, wykorzystujące warunkową wartość oczekiwaną funkcji celu rozwiązań częściowo ustalonych. Prezentowane rozważania dotyczą problemów permutacyjnych, których przykładem jest kwadratowe zagadnienie przydziału.

2. Model matematyczny zagadnienia *QAP*

Zagadnienie *QAP* (ang. *Quadratic Assignment Problem*), należy 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 niż inne zagadnienia optymalizacji kombinatorycznej, to cieszy się powszechnym zainteresowaniem, gdyż modeluje ważną klasę problemów decyzyjnych, dla których rozwiązania można przedstawić w postaci permutacji [4].

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.

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 [2]:

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

Warunkowa wartość oczekiwana funkcji celu pozwala na ocenę jakości rozwiązań częściowo ustalonych [3].

3. Operatory różnicujące a warunkowa wartość oczekiwana funkcji celu

Wykorzystanie warunkowej wartości oczekiwanej funkcji celu dla rozwiązań częściowo ustalonych w konstrukcji algorytmów przybliżonych opiera się na założeniu, że parametr ten niesie w sobie informację na temat perspektywiczności dokonywanych zmian w rozwiązaniach. Przeprowadzone eksperymenty komputerowe [3], [8] wykazały brak korelacji pomiędzy poprawą wartości funkcji celu oraz poprawą warunkowej wartości oczekiwanej dla pary rozwiązań rodzic – potomek. Dla standardowych zadań testowych jedynie dla operatora *PMX* wystąpiła słaba korelacja, niezależnie od jakości rozwiązań w populacji. Badano także zależność wartości oczekiwanej od liczby pozycji ustalonych dla rozwiązań o zróżnicowanej jakości. Zaobserwowano, że przyrasta ona wolniej dla małej liczby pozycji ustalonych, osiągając mniejsze wartości w przypadku rozwiązań o lepszym przystosowaniu. Uzyskane wyniki pozwalają przyjąć, że warunkowa wartość oczekiwana może być istotnym parametrem mechanizmu różnicowania realizowanym przez wytypowane operatory: mutacji *RM* oraz krzyżowania *PMX* i *OX*.

Różnicujące operatory są modyfikacją klasycznych operatorów genetycznych, w których losowe działanie zastąpiono działaniem ukierunkowanym warunkową wartością oczekiwaną. Może to dotyczyć, na przykład w operatorach krzyżowania algorytmu naprawczego, usuwającego „kolizje" i „dziury", wyboru pozycji przecięcia, długości sekcji kojarzenia lub ustalenia pozycji rozwiązań nie podlegających modyfikacji itp. Osłabienie zbieżności algorytmu jest realizowane przez wybór pozycji ustalonych rozwiązania, charakteryzujących się największą wartością oczekiwaną (przy minimalizacji funkcji celu). Mając na względzie złożoność obliczeniową, przyjęto możliwie proste sposoby określenia pozycji ustalonych, rezygnując z wielokrotnego wyznaczania wartości oczekiwanej dla znalezienia ich lokalnych ekstremów. Zaproponowano dwa niedeterministyczne sposoby: pierwszy z losową maską bitową i jej negacją, a drugi dla kilku losowo wybranych masek krzyżowania. W obu przypadkach wyznaczane są warunkowe wartości oczekiwane konkurujących zbiorów pozycji ustalonych i wybierana jest jedna z propozycji. Dla binarnych operatorów (np. krzyżowania) wartości oczekiwane mogą być wyznaczane dla jednego z rozwiązań rodzicielskich, względnie dla obu oddzielnie lub łącznie.

Szczegółowemu opisowi implementacji powyższych operatorów i stosowanego dla celów testowych algorytmu ewolucyjnego wraz z uzyskanymi wynikami badań testowych, poświęcono następny artykuł.

4. Opis implementacji bazowego algorytmu ewolucyjnego - *modGA*

Jako podstawowy schemat algorytmu ewolucyjnego w badaniach eksperymentalnych wykorzystano zmodyfikowany algorytm genetyczny klasy *Steady-State, modG,* zaproponowany przez Michalewicza [9]. Wprowadzone w nim uproszczenia polegają 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. *Napór selekcyjny*, wynikający z długotrwałego utrzymywania w kolejnych populacjach lepszych rozwiązań, jest osłabiany przez sposób reprodukcji (wyboru) rozwiązań rodzicielskich

i eliminację powtarzających się osobników. Zastosowanie dodatkowego operatora optymalizacji lokalnej, obok typowych operatorów genetycznych (dla zagadnień permutacyjnych), znacznie poprawiło zbieżność algorytmu.
Parametrami sterującymi 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.

U jest podzbiorem zbioru operatorów pseudogenetycznych O, do którego należą operatory unarne tzn. takie, które na podstawie jednego rodzica tworzą jednego potomka. Przez P oznaczono populację rozwiązań przetwarzaną przez algorytm, natomiast przez R – wygenerowany zbiór rozwiązań wstępnych.

W algorytmie *modGA* zastosowano następujące procedury:

- GenerujRozwiązaniaWstępne() – generuje losowo, w oparciu o rozkład równomierny, populację rozwiązań początkowych,
- Oceń() – określa wartość funkcji przystosowania (celu) dla rozwiązania lub rozwiązań,
- UtwórzPopulacjęPoczątkową() – 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() – losuje, w oparciu o generator liczb pseudolosowych o rozkładzie równomiernym, operator pseudogenetyczny ze zbioru O, z zadanym 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.

```
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*

5. Operatory genetyczne, mechanizm intensyfikacji i różnicowania

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

- Operator krzyżowania z częściowym odwzorowaniem *PMX (ang. partially matched crossover)* [6].
- Operator krzyżowania z porządkowaniem *OX (ang. order crossover)* [10].
- Operator mutacji *RM (ang. random mutation)* [11].
- Operator optymalizacji lokalnej *LO (ang. local optimization)* [9].

Operator *LO* – dwuoptymalny (stosujący *regułę Heidera*), dokonuje przestawienia wartości na dwóch pozycjach rozwiązania (każda z każdą), wyznaczając najlepsze rozwiązanie w otoczeniu. W przypadku poprawy funkcji celu działanie powtarzane jest dla nowego rozwiązania.

Ostatni z opisanych operatorów realizuje mechanizm intensyfikacji. Jest to często stosowana metoda poprawy efektywności algorytmu, która także sprawdziła się w naszych eksperymentach. Niestety konsekwencją użycia tego operatora jest przedwczesna zbieżność algorytmu, gdyż uzyskane rozwiązania (lokalnie optymalne) stanowią w pewnym sensie pułapkę minimum lokalnego. Konkluzja ta jest na pewno słuszna w odniesienia do operatora *LO*, gdyż nie może on poprawić swojego rozwiązania potomnego, a także operatora mutacji *RM*, który z racji podobieństwa sposobu modyfikacji rozwiązania rodzicielskiego (także przestawienie dwóch pozycji) tworzy rozwiązanie potomne w obszarze przyciągania wyjściowego rozwiązania lokalnie optymalnego. W ten sposób rola operatora mutacji, eksploracja przestrzeni rozwiązań, uległa wyraźnemu ograniczeniu, czemu na ogół przeciwdziała się przez wprowadzenie *makromutacji,* tzn. operatora o dużym zasięgu. Podobne zadanie może realizować jeden z operatorów krzyżowania. Paradoksalnie lepiej nadaje się do tego celu ten, który jest „gorszy", a co za tym idzie, nie poprawia zbieżności algorytmu. Pożądany efekt „psucia" rozwiązania (*różnicowanie*, *dywersyfikacja*) może się różnić skalą zjawiska. Generując w sposób losowy nowe rozwiązanie, dokonujemy losowego skoku w przestrzeni rozwiązań, ewentualnie możemy ograniczyć zasięg operatora, umożliwiając jednak opuszczenie obszaru przyciągania optimum lokalnego. Pierwszy sposób jest prosty i znany (*odświeżanie populacji*), natomiast drugi znacznie trudniejszy do osiągnięcia.

Przedstawione propozycje wykorzystania warunkowej wartości oczekiwanej funkcji celu dotyczą konstrukcji operatorów, realizujących mechanizm różnicowania. Przez zastąpienie losowego działania operatora klasycznego działaniem ukierunkowanym za pomocą warunkowej wartości oczekiwanej, otrzymujemy następujące zmodyfikowane operatory pseudogenetyczne:

- Różnicujący operator krzyżowania *PMX-D*

Różnicowanie dokonuje się w oparciu o wybór maski krzyżowania, która definiuje sekcję kojarzenia wylosowanych rozwiązań rodzicielskich. Ze zbioru losowo ustalonych K masek wybieramy jedną o maksymalnej wartości oczekiwanej obliczonej dla obu rozwiązań (ustalone pozycje pierwszego rozwiązania na zewnątrz sekcji kojarzenia i drugiego wewnątrz), która określa punkty przecięcia rozwiązania. Dalej krzyżowanie przebiega według znanego schematu działania operatora *PMX*.

- Różnicujący operator krzyżowania *OX-D*

Analogiczny przebieg w oparciu o wybraną jedną z wylosowanych masek krzyżowania.

- Różnicujący operator mutacji *RM-D*

Na wstępie losujemy maskę binarną i określamy dla rozwiązania rodzicielskiego wartość oczekiwaną przy ustalonych pozycjach zgodnych z maską i jej negacją. Wybór pozycji ustalonych dokonywany jest na podstawie większej wartości oczekiwanej. Następnie operator *RM-D* dokonuje przestawienia wartości na dwóch wylosowanych pozycjach nieustalonych rozwiązania.

6. Eksperymenty komputerowe, monitorowanie wydajności operatorów

Na podstawie algorytmu *modGA* wykonano eksperymentalne oprogramowanie, zaimplementowane dla zagadnienia *QAP* w dwóch wersjach – podstawowej z wykorzystaniem operatora *LO* oraz wersję stosującą dodatkowo warunkową wartość oczekiwaną jako mechanizm różnicowania (*modGA-D*).

W tabeli 1 przedstawiono wyniki badań eksperymentalnych zagadnienia *QAP* – średnia z 10-ciu przebiegów algorytmu, dla każdego z 39 zagadnień testowych o rozmiarze n=22-64, zaczerpniętych z biblioteki *QAPLIB-A* [1]. Porównując obie wersje algorytmu, w *modGA-D* stosujemy kolejne operatory różnicujące: *PMX-D, OX-D, RM-D*.

Eksperymenty obliczeniowe wykonano dla następującego zbioru parametrów algorytmu: I_{max} =10000 iteracji, λ=100 (rozmiar populacji), zbioru operatorów $O=\{PMX, OX, RM, LO\}$ oraz prawdopodobieństw losowania operatorów (także zmodyfikowanych), wynoszących odpowiednio p_{PMX}=0.3, p_{OX}=0.2, p_{RM}=0.1, p_{LO}=0.4.

Implementując algorytm, zdefiniowano struktury danych, pozwalające na monitorowanie wydajności stosowanych operatorów. Określa się je jako stosunek liczby rozwiązań wstawionych do populacji, do całkowitej liczby wszystkich rozwiązań utworzonych przez dany operator. Rysunek 2 przedstawia porównanie średnich wydajności operatorów, wyznaczonych z 10 przebiegów algorytmu dla 39 zadań.

W tabeli 1, ilustrującej wyniki eksperymentów, przyjęto następujące oznaczenia: $\phi_{\text{QAPLIB-A}}$ – najlepsza znana wartość funkcji przystosowania, zaczerpnięta z biblioteki *QAPLIB-A*, ϕ_{ap} – średnia (z 10 przebiegów) wartość funkcji przystosowania najlepszych znalezionych rozwiązań – π_{bestp}, $E = 100\%*(\phi_{ap} - \phi_{\text{QAPLIB-A}})/\ \phi_{\text{QAPLIB-A}}$.

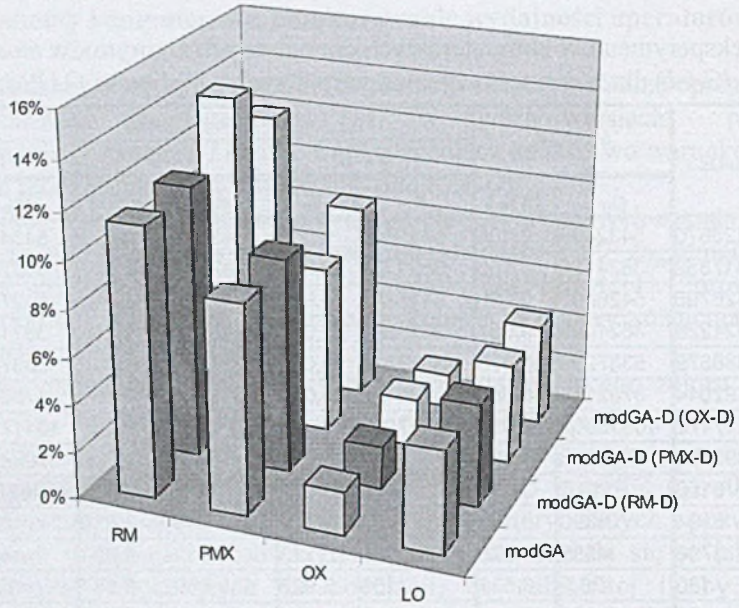
Dodatkowo w trakcie przebiegu algorytmu zbierana była wiedza, pozwalająca odtworzyć historię utworzenia rozwiązań populacji końcowej. Na jej podstawie wnioskować można o skuteczności operatorów, rozumianej jako zdolność do utworzenia, bądź współtworzenia rozwiązań o najwyższej jakości. Wiedza ta jest użyteczna w trakcie prac nad konstrukcją algorytmu, ponieważ pozwala właściwie zrozumieć rolę stosowanych operatorów. Niestety jej prezentacja w zwięzłej formie nastręcza pewne trudności.

Rysunek 3 stanowi próbę uproszczonego przedstawienia historii utworzenia populacji końcowej dla zadania *Bur26f*. Prezentuje on porównanie procentowego udziału poszczególnych operatorów, w trzech ostatnich modyfikacjach rozwiązań populacji końcowych, dla algorytmów *modGA* oraz *modGA-D*.

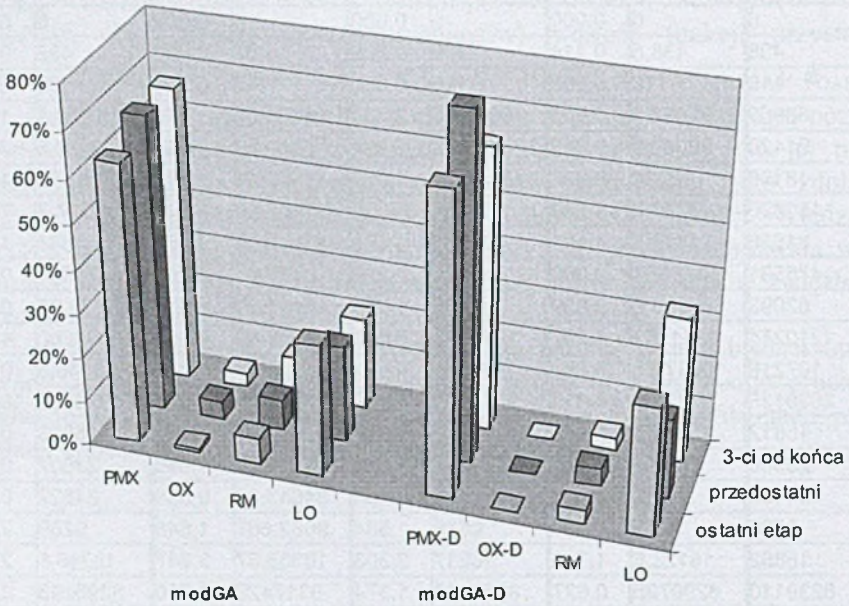
Tabela 1

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

Nazwa	$\phi_{QAPLIB-A}$	ModGA		modGA-D					
				PMX-D		OX-D		RM-D	
		ϕ_{ap}	E[%]	ϕ_{ap}	E[%]	ϕ_{ap}	E[%]	ϕ_{ap}	E[%]
BUR26A	5426670	5432408	0,106	5429665	0,055	5432063	0,099	5434578	0,146
BUR26B	3817852	3824135	0,165	3821123	0,086	3820403	0,067	3822033	0,109
BUR26C	5426795	5426800	0,000	5426800	0,000	5426800	0,000	5427063	0,005
BUR26D	3821228	3821495	0,007	3821525	0,008	3821360	0,003	3821370	0,004
BUR26E	5386879	5387288	0,008	5387330	0,008	5387440	0,010	5387418	0,010
BUR26F	3782044	3782145	0,003	3782100	0,001	3782373	0,009	3782250	0,005
BUR26G	10117172	10118300	0,011	10117925	0,007	10118133	0,010	10117725	0,005
BUR26H	7098658	7098723	0,001	7099105	0,006	7098910	0,004	7098925	0,004
CHR22A	6156	6319,5	2,656	6386	3,736	6324,667	2,740	6355,5	3,241
CHR22B	6194	6390	3,164	6408	3,455	6418	3,616	6420	3,649
CHR25A	3796	4539	19,573	4559,5	20,113	4134	8,904	4682	23,340
ESC32A	130	138,5	6,538	139	6,923	138,6667	6,667	139,5	7,308
ESC32B	160	168	5,000	176	10,000	173,3333	8,333	172	7,500
ESC32C	642	642	0,000	642	0,000	642	0,000	642	0,000
ESC32D	200	200	0,000	200	0,000	200	0,000	200	0,000
ESC32E	2	2	0,000	2	0,000	2	0,000	2	0,000
ESC32F	2	2	0,000	2	0,000	2	0,000	2	0,000
ESC32G	6	6	0,000	6	0,000	6	0,000	6	0,000
ESC32H	438	438,5	0,114	439	0,228	438	0,000	438	0,000
ESC64A	116	116	0,000	116	0,000	116	0,000	116	0,000
KRA30A	88900	90067,5	1,313	90887,5	2,236	90206,67	1,470	90397,5	1,684
KRA30B	91420	92007,5	0,643	91780	0,394	91746,67	0,357	92157,5	0,807
LIPA30A	13178	13301	0,933	13319,25	1,072	13315	1,040	13363,75	1,410
LIPA30B	151426	157027,3	3,699	151426	0,000	159069,7	5,048	162787,3	7,503
LIPA40A	31538	31887,5	1,108	31886	1,103	31840,67	0,960	31884	1,097
LIPA40B	476581	476581	0,000	476581	0,000	476581	0,000	476581	0,000
LIPA50A	62093	62688,5	0,959	62735	1,034	62661,67	0,916	62712,75	0,998
LIPA50B	1210244	1261163	4,207	1210240	0,000	1210240	0,000	1262600	4,326
LIPA60A	107218	108170,8	0,889	108133	0,853	108137,3	0,857	108146,5	0,866
LIPA60B	2520135	2870905	13,919	2755053	9,322	2989680	18,632	2983495	18,386
SKO42	15812	15892,5	0,509	15919,5	0,680	15908	0,607	15893	0,512
SKO49	23386	23478	0,393	23506	0,513	23548,67	0,696	23507	0,517
SKO56	34458	34743	0,827	34729	0,786	34643,33	0,538	34622	0,476
STE36A	9526	9682	1,638	9675	1,564	9682,667	1,645	9769	2,551
STE36B	15852	16123,5	1,713	16217	2,303	16366,67	3,247	16246,5	2,489
STE36C	8239110	8290795	0,627	8352340	1,374	8317420	0,950	8295293	0,682
THO30	149936	150828,5	0,595	150627	0,461	150890	0,636	150609	0,449
THO40	240516	243126	1,085	243072	1,063	243788	1,360	242612	0,871
WIL50	48816	48974	0,324	48971	0,318	48902,67	0,178	48911,5	0,196
Błąd (suma)			**72,727**		**69,703**		**69,598**		**91,146**



Rys. 2. Porównanie wydajności operatorów [%] dla algorytmów *modGA* i *modGA-D*



Rys. 3. Procentowy udział poszczególnych operatorów w trzech ostatnich etapach utworzenia rozwiązań populacji końcowej dla algorytmów *modGA* oraz *modGA-D*

7. Podsumowanie

Zamieszczone w tabeli 1 wyniki pozwalają zauważyć, że modyfikacja działania operatorów krzyżowania, przez wprowadzenie warunkowej wartości oczekiwanej, pozwala osiągać wyniki nieco lepsze niż dla algorytmu *modGA*. Dla operatorów *PMX-D*, *OX-D*, średnia poprawa uzyskanych wyników wynosi ok. 0.1%, natomiast dla operatora *RM-D* zaobserwowano wyraźne pogorszenie uzyskanych wyników (o ok. 0.3%). Średnią ocenę jakości uzyskanych wyników pogarsza występowanie specyficznych zadań testowych: *CHR25A* oraz *ESC32B*.

Oczywistym jest, że wraz ze wzrostem liczby iteracji algorytmu wydajności wszystkich operatorów maleją (rośnie średnia wartość funkcji przystosowania dla rozwiązań w populacji, przez co trudniejsza jest ich poprawa).

Analizując rysunek 2, w przypadku mechanizmu różnicowania wydajność zmodyfikowanego operatora maleje, a jego kosztem rosną wydajności pozostałych operatorów (przejmują jego rolę). Nie dotyczy to operatora *LO*, dla którego wydajność jest praktycznie stała (ok. 4.4%), a jej zaskakująco niska wartość (przy wysokiej skuteczności *LO*) wynika z odrzucenia powtarzających się rozwiązań.

Zastosowanie dwóch etapów realizacji operatora *RM-D* (wybór maski i losowanie wymienianych pozycji) znacznie ogranicza różnicowanie, przez co bardziej upodabnia jego działanie do operatora *RM*.

Porównując na obu wykresach (rys. 3) udział operatorów w utworzeniu rozwiązań populacji końcowej widać, że, pomimo spadku wydajności dla operatorów różnicujących, dla *PMX-D* nastąpił wzrost jego znaczenia w utworzeniu populacji końcowej (odpowiednio z 63%, 67%, 67% dla *modGA* do 68%, 74% i 64% dla *modGA-D*). Dla przedstawionego zadania wprowadzenie różnicowania zaowocowało poprawą wartości funkcji celu najlepszego znalezionego rozwiązania. Dla *modGA* pierwsze pozycje populacji końcowej zajmują rozwiązania, w historii których dominuje operator *LO* (najczęściej sekwencja *LO – PMX - LO*), natomiast dla *modGA-D* w gronie najlepszy rozwiązań pojawiają się też utworzone przez operator *PMX-D* (sekwencja *PMX-D – PMX-D – LO* oraz 3 razy *PMX-D*).

LITERATURA

1. Burkard R.E., Karisch S.E., Rendl F.: QAPLIB-A Quadratic Assignment Problem Library. European Journal of Operational Resarch, 1991, 15, 115.
2. Chmiel W.: Algorytmy ewolucyjne w optymalizacji przydziału zadań z kwadratową funkcją celu. AGH, Kraków 2004.
3. Chmiel W., Kadłuczka P.: Warunkowa wartość oczekiwana funkcji celu w konstrukcji algorytmów przybliżonych dla zagadnień permutacyjnych. Automatyka, Wydawnictwo AGH, Kraków 2005, s. 47-56.
4. Filipowicz B., Wala K.: Algorytmy optymalizacji kwadratowego zagadnienia przydziału. Kwartalnik Elektrotechnika, z. 1, Wydawnictwo AGH, Kraków 1992.
5. Goldberg D. E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company, 1989.

6. Goldberg D., Lingle R.: Alleles, loci, and the Traveling Salesman Problem. Proc. International Conference on Genetic Algorithms and their Applications, 1985.
7. Hoffmeister F., Bäck T.: Genetic Algorithms and Evolution Strategies: Similarities and Differences. Technical report SYS-1/92, Universität Dortmund, Fachbereich Informatik, Dortmund 1992.
8. Kadłuczka P., Chmiel W.: Efektywność algorytmu ewolucyjnego wykorzystującego warunkową wartość oczekiwaną funkcji celu. Automatyka, Wydawnictwo AGH, Kraków 2005, s. 105-114.
9. Michalewicz Z.: Genetic Algorithms + Data Strructures = Evolution Programs. Springer Verlag, 1995.
10. 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.
11. Wala K., Chmiel W.: Evolution Algorithm for Quadratic Assignment Problem. University of Mining and Metallurgy Press, Elektrotechnika 1, Kraków 1997, p. 409-419.

Recenzent: Prof. dr hab. inż. Zdzisław Duda

Abstract

The paper presents our work on implementation and evaluation of evolutionary algorithm using diversification operators. They are based on expected conditional value of objective function for partially fixed solutions. The experiments were performed for standard test problems of quadratic assignment problems (*QAP*) from *QAPLIB-A* library. We compare the results of modified evolutionary algorithm *modGA-D* with the results of classical algorithm proposed by Z. Michalewicz extended by the operator of local optimization. For testing purposes we have introduced in both algorithms data structures facilitating tracking the history of evolution leading to the generation of final population.