Mariusz MAKUCHOWSKI
Politechnika Wrocławska

OPERATOR KRZYŻOWANIA GOX DLA PROBLEMU KOMIWOJAŻERA

Streszczenie. W pracy przedstawiono nowe operatory krzyżowania, mające zastosowanie w algorytmach genetycznych. Proponowane operatory dedykowane są dla algorytmów rozwiązujących problem komiwojażera. Praca zawiera wyniki porównawcze badań efektywności proponowanych operatorów z operatorami klasycznymi. Badania przeprowadzone są na znanych w literaturze benchmarkach testowych.

GREEDY ORDERED CROSSOVER (GOX) FOR THE TRAVELING SALESMAN PROBLEM

Summary. The study presents a new greedy ordered crossover operators (GOX), applying genetic algorithms. Proposed operators are dedicated algorithms for solving traveling salesman problem, which allows to find good quality solution in the shortest time then original crossover operators. The efficiency of the algorithms is tested on well–known literature benchmarks.

1. Wstęp

Jednym z klasycznych i zarazem bardzo ciekawych podejść rozwiązywania problemów kombinatorycznych są algorytmy genetyczne GA (ang. Genetic Algorithms). Naśladują one naturę, a dokładnie ewolucję organizmów żywych. W każdym algorytmie GA można wydzielić charakterystyczne etapy działania. Efektywność działania całego algorytmu zależy od właściwego zaprojektowania, wysterowania i współpracy tych bloków. Jednym z nich jest procedura tworzenie nowych rozwiązań przy pomocy genetycznych operatorów krzyżowania. W niniejszej pracy zaprezentowano nowy operator krzyżowania GOX polecany dla algorytmów GA w szczególności dedykowanych problemowi komiwojażera TSP (ang. Traveling Salesman Problem) oraz jego zmodyfikowaną wersję GOX^S dedykowaną dla szczególnego przypadku TSP jakim jest symetryczny problem komiwojażera.

Niniejsza praca zawiera porównanie efektywności algorytmów GA z zastosowaniem przedstawionych w pracy operatorów krzyżowania GOX i GOX^S w stosunku do zastosowania klasycznych operatorów krzyżowania CX, PMX, OX. Badania przeprowadzone zostały na znanych w literaturze przykładach testowych powszechnie stosowanych do porównywania efektywności badanych algorytmów.

2. Problem komiwojażera

Jednym z najwcześniej, a zarazem najczęściej studiowanym problemem kombinatorycznym jest problem komiwojażera. Fakt ten wynika z bardzo prostego jego sformułowania oraz trudnością w wyznaczeniu rozwiązania optymalnego. Ogólny problem TSP można zdefiniować następująco:

Dany jest zbiór $C = \{1, 2, \ldots, n\}$ reprezentujący n miast oraz kwadratowa macierz d o rozmiarze $n \times n$. Wartość $d_{ij} \in \mathbb{N}^+$ definiuje odległość z miasta i do miasta j. Niech π oznacza permutację miast C. Zbiór wszystkich permutacji oznaczać będziemy przez Π. Permutacja π utożsamiana jest z cykliczną trasą kolejno odwiedzanych miast. Dla każdej trasy π możemy wyznaczyć jej długość, oznaczanej przez $D(\pi)$;

$$D(\pi) = \sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)} + d_{\pi(n),\pi(0)}. \tag{1}$$

Problem polega na wyznaczeniu permutacji π^* minimalizującej długość cyklu,

$$\pi^* \in \arg\min_{\pi \in \Pi} D(\pi). \tag{2}$$

W literaturze często można spotkać specjalne przypadki TSP, w którym na macierz odległości nałożone są pewne naturalne ograniczenia. Najczęściej spotykane w literaturze przypadki TSP to:

- Symetryczny TSP. Wariant w którym macierz odległości jest symetryczna; $d_{i,j} = d_{j,i}$.
- Metryczny TSP. Wariant w którym odległości są pewną metryką, czyli spełniają:
  - — warunek identyczności: $d_{i,j} = 0 \Leftrightarrow i = j$,
  - — warunek symetrii: $d_{i,j} = d_{j,i}$,
  - — warunek trójkąta: $d_{i,j} \leqslant d_{i,k} + d_{k,j}$.
- Euklidesowy TSP. Wariant w którym miasto położone jest w przestrzeni k wymiarowej a odległość dana jest wzorem:

  $$d_{i,j} = \sqrt{|x_1 - y_1|^2 + |x_2 - y_2|^2 + \ldots + |x_k - y_k|^2},$$

  dla współrzędnych miasta $i = (x_1, x_2, \cdots, x_k)$ i miasta $j = (y_1, y_2, \cdots, y_k)$.

Problem Euklidesowy TSP jest szczególnym przypadkiem problemu metrycznego TSP, a ten z kolei jest szczególnym przypadkiem problemu symetrycznego TSP. Problem TSP dla wszystkich opisanych powyżej przypadków należy do klasy problemów *NP-trudnych*. Z wymienionych przypadków tylko dla euklidesowej wersji problemu istnieje algorytm typu PTAS [1].

3. Algorytm genetyczny

Algorytmy genetyczne GA zaproponowane zostały po raz pierwszy przez Johan'a Hollanda'a w 1975 roku [5]. Algorytmy genetyczne są przybliżonymi algorytmami typu popraw. Oznacza to, iż w każdej iteracji starają się poprawić bieżące rozwiązanie (rozwiązania), z jednoczesnym zapobieganiem stagnacji obliczeń w minimum lokalnym. Swoim działaniem naśladują przyrodę a dokładnie ewolucję organizmów żywych.

Każda iteracja algorytmu symuluje jedno pokolenie, w którym najmniej przystosowane osobniki giną bezpotomnie. Osobniki lepiej przystosowane (reprezentujące lepsze rozwiązania) stają się rodzicami kolejnego pokolenia, przekazując swoje geny, czyli cechy rozwiązań, potomstwu.

W każdym algorytmie genetycznym można wyróżnić, powtarzane z każdym pokoleniem, bazowe bloki [3, 9]:

- selekcja: odpowiedzialna za wybór rodziców dla kolejnego pokolenia,
- krzyżowanie: tworzenie nowych rozwiązań o cechach swoich rodziców,
- mutacja: niewielkie zaburzenia tworzonych potomków.

Ogólna zasada powyższych bloków jest jednakowa w każdym algorytmie GA. Niemniej ze względu na zastosowanie tych algorytmów w różnorakich problemach optymalizacyjnych powoduje, iż każdy element powinien zostać zaprojektowany indywidualnie.

4. Operatory genetyczne dla problemu TSP

W przypadku problemu TSP rozwiązanie $\pi \in \Pi$ może być pamiętane w postaci ciągu kolejnych wartości $(\pi(1), \pi(2), \ldots, \pi(n))$. Dla takiego kodowania rozwiązań należy mieć na uwadze, iż ostatecznie w wyniku krzyżowania i mutacji otrzymane rozwiązanie musi reprezentować poprawną permutację miast.

Konstrukcje genetycznych operatorów krzyżowania dedykowanych problemowi TSP były wielokrotnie poruszane w literaturze. Jako pierwsze, już w roku 1985, zostały opisane operatory: OX (ang. Ordered Crossover) [2] oraz PMX (ang. Partition Crossover) [4]. Operator CX (ang.Cycle Crossover) zaproponowany został w pracy [10]. Wymienione operatory są klasycznymi operatorami dla problemu TSP. Idea klasycznych operatorów krzyżowania polega na utworzeniu potomka, w którym część sekwencji pochodzi od jednego z rodziców a pozostała część sekwencji od drugiego.

Kolejne pomysły na krzyżowanie rozwiązań bazują na kopiowaniu krawędzi występujących w rozwiązaniach rodzicielskich. Krawędzią nazywana jest para sąsiednich miast występująca w rozwiązaniu. Pierwszy z tego rodzaju operatorów to EX (ang. Edge Crossover) [14]. Kolejny bardzo efektywny operator tego rodzaju to EAX (ang. Edge Assembly Crossover) [7, 6]. Na bazie tego operatora zaproponowano efektywny algorytm GA [8]. Inny operator bazujący na krawędziach to PX (ang. Partition Crossover) opisany w [13].

Innym podejściem są operatory oparte na metodzie ścieżek łączących (w przestrzeni rozwiązań) dwa rozwiązania będącymi rodzicami. Operatory krzyżowania oparte na tym podejściu to MSX (ang. Multi Step Crossover) opisany w pracy [15] oraz MSXF opisany w pracy [11].

5. Operator GOX

Złożoność obliczeniowa klasycznych operatorów CX, PMX, OX jest rzędu $\mathcal{O}(n)$. W pracy prezentowany jest operator GOX (ang. Greedy Ordered Crossover). Jest to zachłanna wersja operatora OX. Złożoność obliczeniowa operatora GOX pozostaje na poziomie $\mathcal{O}(n)$.

Niech α oraz β oznaczają rozwiązania rodzicielskie (permutacje miast), poddawane operacji krzyżowania. Operator GOX jest operatorem dwupunktowym, więc dodatkowo niech $start$ oznacza punkt przecięcia rodzica α oraz $length$ ilość kopiowanych pozycji z rodzica α.

Rozwiązanie potomne $\gamma = GOX(\alpha, \beta, start, length)$ powstaje w 3 fazach.

- W fazie 1 tworzona jest ścieżka $path$ na podstawie rodzica α. Powstaje ona w wyniku skopiowania $length$ miast począwszy pod pozycji $start$ (zakładamy, iż po ostatniej pozycji występuje pozycja pierwsza).
- W fazie 2 tworzony jest cykl $cycle$ pozostałych $n - length$ miast. Powstaje ono poprzez usunięcie z rozwiązania β elementów skopiowanych w fazie 1.
- W fazie 3 tworzone jest rozwiązanie poprzez włączenie ścieżki $path$ w wybrane miejsce w cyklu $cycle$. Wybór miejsca rozerwania cyklu i uzupełnienia go ścieżką, odbywa się w sposób zachłanny (optymalny) pod względem długości tworzonego potomka. Wyznaczenie względnej długości nowego potomka odbywa się w czasie $\mathcal{O}(1)$. Niech $p1$ oznacza pierwsze miasto w ścieżce $path$, a $p2$ miasto ostatnie. Ponadto niech $c1$ i $c2$ oznaczają sąsiednie miasta w cyklu pomiędzy które włączana będzie ścieżka. Długość utworzonego potomka γ można wyznaczyć jako:

$$D(\gamma) = X + d_{p2,c2} + d_{c1,p1} - d_{c1,c2}, \tag{3}$$

  gdzie X jest stałą sumą długości ścieżki $path$ i długością cyklu. Sprawdzając wszystkie $n - length$ możliwości możemy wybrać najlepsze połączenie w czasie $\mathcal{O}(n)$.

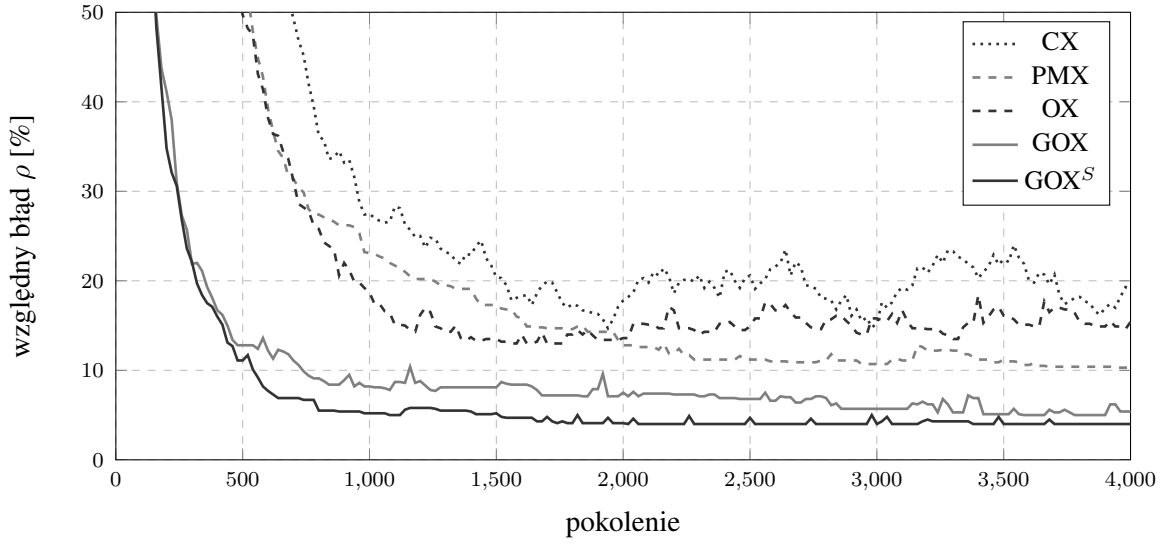
W przypadku symetrycznego TSP proponuje się rozszerzyć 3 fazę operatora. Zaproponowano sprawdzić także połączenie odwróconej ścieżki $path$. W tym wypadku w celu wyznaczenia długości powstającego rozwiązania należy skorzystać z wzoru 3 w którym należy zamienić indeksy $p1$ z $p2$. Operator z powyższą poprawką w dalszej części pracy oznaczany będzie przez GOX^S .

6. Badania numeryczne

Wszystkie eksperymenty numeryczne zrealizowano na komputerze MacBook Pro z procesorem Intel i5 2.5GHz w systemie Windows XP uruchomionym na wirtualnej maszynie działającej w systemie OSX 10.9.5. Zastosowane instancje testowe są literaturowymi benchmarkami pochodzącymi z biblioteki TSPLIB [12]. Dla większości instancji biblioteka TSPLIB zawiera także rozwiązania optymalne. Dla każdego rozwiązania π możemy więc wyznaczyć, liczony w stosunku do rozwiązania optymalnego π^*, błąd względny długości trasy:

$$\rho(\pi) = \frac{D(\pi) - D(\pi^*)}{D(\pi^*)} \cdot 100\%. \tag{4}$$

Ponieważ w algorytmach GA występują pewne losowe wartości, wyniki nie są powtarzalne. Z powyższego powodu dla danej instancji problemu i każdego z zestawu parametrów sterujących, algorytm GA uruchamiany był 10 razy. Na podstawie otrzymanej serii rozwiązań wyznaczono odpowiednio: wartość średnią błędu względnego ρ_{av},



Rys. 1. Zbieżność algorytmu GA dla różnych operatorów krzyżowania (*KroA200.tsp*)

odchylenie standardowe błędu względnego ρ_{dev} oraz całkowity czas działania 10 uruchomień algorytmu t_{sum}.

W eksperymentach zamianie podlegać będzie tylko operator krzyżowania na jeden z CX, PMX, OX, GOX i GOX^S . Pozostałe parametry algorytmu nie są modyfikowane i wynoszą odpowiednio:

- Pokolenie startowe składa się z losowych permutacji;
- Liczebność populacji przez cały czas trwania algorytmu wynosi 100 osobników;
- Kryterium stopu następuje po wystąpieniu 4000 pokolenia (10000 w 3 eksperymencie);
- Selekcja następuje poprzez wybranie k-atego rozwiązania pod względem jakości, gdzie k jest zaokrągloną zmienną losową o rozkładzie wykładniczym i wartości oczekiwanej 4;
- Operatorem mutacji jest operator inwersji losowego fragmentu permutacji;
- Prawdopodobieństwo mutacji jest zmienne. W przypadku, gdy wartość funkcji celu najlepszego osobnika w pokoleniu jest taka sama jak dla pokolenia wcześniejszego prawdopodobieństwo mutacji dla kolejnego pokolenia wzrasta do wartości $P_{mut} = 0,95$, w przeciwny wypadku wynosi $P_{mut} = 0,4$.

Pierwszy eksperyment polegał na wygenerowaniu przebiegów algorytmów GA w zależności od zastosowanego operatora krzyżowania. Przez przebieg rozumie się wykres przedstawiający wartość względnego błędu ρ najlepszego rozwiązania dla kolejnych pokoleń algorytmu. Na rysunku 1 przedstawiono przebiegi algorytmu GA dla przykładowej instancji *KroA200.tsp*.

Autor wygenerował setki wykresów przedstawiających przebiegi algorytmów GA dla różnych jego wysterowań oraz różnych instancji. Różne wysterowania dotyczyły zmiany sposobu generowania pokolenia startowego, zmiany operatora mutacji, zmiany prawdopodobieństwa mutacji, zmiany ilości osobników w pokoleniu oraz całkowitej liczby pokoleń. Operatory GOX i GOX^S były zawsze bardziej efektywne niż

Tabela 1

Wpływ operatora krzyżowania na błąd ρ[%] i czas t[s], 4000 pokoleń

instancja	CX			PMX			OX			GOX			GOX^S		
	ρ_{avg}	ρ_{dev}	t_{sum}	ρ_{avg}	ρ_{dev}	t_{sum}	ρ_{avg}	ρ_{dev}	t_{sum}	ρ_{avg}	ρ_{dev}	t_{sum}	ρ_{avg}	ρ_{dev}	t_{sum}
eil51	0.7	0.7	6.5	1.4	1.0	7.6	1.2	0.8	6.5	1.1	0.6	9.5	1.0	0.7	11.5
eil76	2.7	1.1	8.5	3.9	1.0	10.0	4.2	1.3	8.3	2.3	0.5	12.6	2.2	0.7	16.0
kroA100	4.9	2.9	10.5	3.9	2.6	12.8	3.3	2.4	11.0	1.6	1.1	16.5	1.4	1.0	20.6
kroB100	3.6	1.1	10.5	4.1	2.0	12.5	4.4	1.5	10.5	2.5	1.2	16.0	2.1	1.1	20.4
kroA150	9.2	1.8	14.6	6.5	1.5	17.6	6.5	2.3	14.6	2.8	1.6	22.8	3.1	1.1	28.9
kroB150	8.2	1.4	14.6	6.4	3.2	17.5	7.5	2.2	14.6	3.6	1.6	22.7	2.1	1.2	29.1
rat195	17.0	2.0	18.1	10.3	1.6	22.5	11.3	1.7	18.1	5.8	0.8	28.8	4.8	1.4	36.8
kroA200	13.7	1.2	18.5	8.9	1.9	22.8	9.8	1.7	18.6	4.3	1.6	29.6	3.2	1.2	37.8
kroB200	16.1	1.5	18.6	9.7	2.6	22.8	10.6	2.7	18.6	4.7	1.1	29.6	3.8	1.3	38.2
a280	27.5	2.0	24.1	17.1	2.3	30.3	16.2	2.1	24.6	8.3	1.6	39.2	7.1	2.6	50.0
rd400	37.5	1.3	35.1	25.7	3.6	43.6	22.9	2.6	35.5	11.4	1.2	56.6	8.6	1.0	72.9
wszystko	12.8	-	180.1	8.9	-	220.6	8.9	-	181.3	**4.4**	-	**284.5**	**3.6**	-	**362.5**

Tabela 2

Wpływ operatora krzyżowania na błąd ρ[%] i czas t[s], 10 000 pokoleń

instancja	CX			PMX			OX			GOX			GOX^S		
	ρ_{avg}	ρ_{dev}	t_{sum}	ρ_{avg}	ρ_{dev}	t_{sum}	ρ_{avg}	ρ_{dev}	t_{sum}	ρ_{avg}	ρ_{dev}	t_{sum}	ρ_{avg}	ρ_{dev}	t_{sum}
eil51	0.3	0.3	14.9	0.7	0.6	17.8	0.7	0.6	14.6	0.7	0.7	22.5	0.6	0.7	27.9
eil76	1.9	0.5	19.8	2.3	1.0	24.0	2.7	1.1	19.8	1.8	0.5	30.8	1.6	0.9	38.8
kroA100	2.2	1.3	24.6	2.4	2.4	30.0	2.2	1.7	24.6	0.9	1.2	38.9	1.2	0.9	49.1
kroB100	2.5	0.8	24.6	2.7	1.0	29.8	3.0	1.6	24.8	2.1	0.9	38.7	1.8	1.1	49.2
kroA150	6.8	1.2	34.9	4.7	1.6	42.5	4.8	1.7	34.6	2.2	1.3	55.6	2.6	1.1	70.6
kroB150	6.0	1.2	34.7	4.5	1.6	42.3	5.5	2.6	34.6	2.7	1.1	55.3	2.0	1.1	70.7
rat195	14.8	1.5	43.8	8.0	1.8	54.3	9.1	1.6	44.1	3.8	0.7	70.9	3.8	1.1	90.5
kroA200	11.3	2.0	45.2	6.4	2.0	55.5	7.8	1.3	45.1	3.2	1.0	72.9	2.7	1.0	92.8
kroB200	11.9	0.9	45.0	6.7	2.0	55.6	8.0	1.5	45.1	3.3	0.8	72.7	3.3	1.2	92.5
a280	21.9	1.5	58.7	12.2	2.0	72.8	12.1	2.8	58.5	6.4	1.6	95.9	5.4	2.3	123.1
rd400	32.2	1.3	86.3	14.4	1.2	107.6	12.7	1.5	86.2	7.2	1.2	140.7	6.2	1.2	180.0
wszystko	**10.2**	-	**432.8**	**5.9**	-	**532.6**	**6.2**	-	**432.4**	3.1	-	695.3	2.8	-	885.7

rozważane operatory klasyczne. Efektywność ta rozumiana jest jako szybkość zmniejszania się błędu względnego ρ w kolejnych pokoleniach algorytmu. Wadą w stosunku do klasycznych operatorów jest wolniejsze opuszczanie minimów lokalnych, konieczność odczytywania odległości pomiędzy miastami oraz około dwukrotnie dłuższy czas działania algorytmu.

Drugi eksperyment polega na wyznaczeniu średnich błędów względnych ρ. W tym celu dla zestawu instancji dla każdego testowanego operatora krzyżowania, algorytm genetyczny uruchomiony był 10 razy. Otrzymane rezultaty przedstawione zostały w tabeli 1. Dla testowanych ustawień algorytmu GA zastosowanie operatora GOX powoduje około 2-krotne zmniejszenie średniego błędu generowanych rozwiązań w stosunku do operatorów klasycznych. Operator GOX^S wypada jeszcze lepiej, generuje rozwiązania z błędem względnym ponad 2,4- krotnie mniejszym w stosunku do klasycznych operatorów. Niemniej czas działania algorytmu GA z zastosowaniem proponowanych

Tabela 3
Czas t[s] działania algorytmu GA, z macierzą odległości, dla różnych operatorów

iteracje	CX	PMX	OX	GOX	GOX^S
4.000	79.3	137.0	96.7	137.3	144.1
10.000	179.2	321.0	222.4	321.5	339.5

operatorów wzrasta prawie dwukrotnie. Dlatego też autor zdecydował się zamieścić wyniki badań dla zwiększonej do 10 000 liczby pokoleń algorytmu GA w kolejnej tabeli.

Porównamy teraz średnie błędy ρ_{avg} i czasy działania algorytmów t_{sum} dla operatorów GOX i GOX^S dla 4 000 pokoleń (tabela 1) z algorytmami z operatorami klasycznymi dla 10 000 pokoleń (tabela 2). Najistotniejsze z analizowanych wartości zostały zapisane pogrubionym fontem.

W powyższym porównaniu zastosowanie proponowanych operatorów wraz ze zmniejszeniem liczby pokoleń algorytmu, skutkuje uzyskaniem lepszych rozwiązań w krótszym czasie, w stosunku do operatorów klasycznych. Przykładowo dla operatora GOX w stosunku do OX w czasie 35% krótszym ($t^{GOX} = 285$s, $t^{OX} = 432$s), otrzymano rozwiązania z błędem 30% mniejszym ($\rho^{GOX} = 4.4\%$, $\rho^{OX} = 6,2\%$).

Dodatkowe badania wykazały, iż przyczyną prawie dwukrotnego wzrostu czasu działania algorytmu z zastosowaniem GOX GOX^S jest konieczność korzystania z odległości pomiędzy parami miast. Format danych wejściowych zawiera tylko współrzędne miast, więc proponowane operatory muszą dokonywać obliczeń odległości na ich podstawie. Czas wyznaczania długości pomiędzy miastami jest stosunkowo długi w porównaniu do reszty operacji numerycznych wykonywanych w omawainych operatorach.

W przypadku, gdy macierz odległości pomiędzy wszystkimi parami miast, jest daną wejściową, lub macierz taka wyznaczana jest na początku algorytmu czasy działania algorytmów z zaproponowanymi operatorami są bliższe czasom działania algorytmów wykorzystujących klasyczne operatory krzyżowania. Tabela 3 zawiera czasy działania 5 testowanych algorytmów, w których na wstępie dokonano stworzenia macierzy odległości. Krok ten ma złożoność czasową i pamięciową rzędu $\mathcal{O}(n^2)$, gdzie n jest liczbą miast. Dokonując wstępnego wyznaczenia macierzy odległości, pomimo zwiększenia złożoności całego algorytmu z $\mathcal{O}(n)$ do $\mathcal{O}(n^2)$, dla testowanych, stosunkowo niewielkich instancji, czas działania algorytmu ulega skróceniu.

7. Podsumowanie

W algorytmach genetycznych, dedykowanych symetrycznemu problemowi komiwojażera, przedstawione w pracy operatory krzyżowania GOX i GOX^S okazały się dużo bardziej efektywne niż klasyczne operatory CX, PMX i OX.

Algorytmy genetyczne z proponowanymi operatorami dostarczały rozwiązań znacznie lepszych (o dwukrotnie mniejszym błędzie) niż klasyczne operatory, jednak przy tej samej ilości pokoleń działały także prawie dwukrotnie dłużej.

Przewaga jakościowa algorytmów z proponowanymi operatorami jest tak duża, że przy wykonywaniu mniejszej liczby iteracji w stosunku do algorytmów z klasycznymi operatorami, czas działania jest krótszy, a otrzymane rozwiązania są zdecydowanie lepsze.

LITERATURA

1. Arora S., Polynomial-time approximation schemes for Euclidean traveling salesman and other geometric problems, Journal of the ACM 45, 1998, p.753-–782.
2. Davis, L. Applying adaptive Algorithms to Epistatic Domains, Proceeding of the International Joint Conference on Artificial Intelligence, IEEE Computer Society Press, Los Angeles, 1985, p. 162–164.
3. Goldberg D.E., Algorytmy genetyczne i ich zastosowania, WNT, Warszawa, 1998.
4. Goldberg D.E., Lingle R., Alleles, Loci, and the traveling salesman problem. In: Grefenstette (ed), Proceedings of the Second International Conference on Genetic Algorithms and Their Applications, Carnegie-Mellon University, 1985, p. 154–159.
5. Holland J.H, Adaptation in natural and artificial systems, University of Michigan Press, 1975.
6. Nagata Y., New EAX crossover for large TSP instances. Parallel Problem Solving from Nature-PPSN IX, 2006, p. 372-–381.
7. Nagata Y., Kobayashi S., Edge Assembly Crossover: A High-Power Genetic Algorithm for the Traveling Salesman Problem. 7th International Conf. on GAs, 1997, p. 450–457.
8. Nagata Y., Kobayashi S., A Powerful Genetic Algorithm Using Edge Assembly Crossover for the Traveling Salesman Problem, Journal on Computing archive Volume 25 Issue 2, Spring, 2013, p. 346–363.
9. Michalewicz Z., Algorytmy genetyczne+struktury danych=programy ewolucyjne, WNT, Warszawa, 1999.
10. Oliver I.M., Smith D.J., Holland J.R.C., A Study of Permutation Crossover Operator on the TSP, Proceedings of the Second International Conference , Lawrence Erolaum, New Jersy, 1987, p. 224–230.
11. Reeves C. R., Yamada T., Solving the Csum Permutation Flowshop Scheduling Problem by Genetic Local Search, IEEE International Conference on Evolutionary Computation, 1998, p.230–234.
12. Reinhelt, G.: TSPLIB, Zbiór instancji problemu TSP,
Internet: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/,
data pobrania: 01.01.2016.
13. Whitley D., Hains D., Howe A., Tunneling between optima: partition crossover for the traveling salesman problem, Proceedings of the 11th Annual conference on Genetic and evolutionary computation, 2009, p. 915–922,
14. Whitley D., Starkweather T., Fuquay D., Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator, Proc. of the 3ed Int. Conf. on Genetic Algorithms, 1989, p. 133–140.
15. Yamada T., Nakano R., A GA with multi-step crossover for job-shop scheduling problems, Proc. of Int. Conf. on GAs in Engneering Systems: Innovations and Applications, 1995, p. 146–151.