Optymalizacja kombinatoryczna			
Kierunek *Informatyka*	**Specjalność** –	**Rok studiów** *II*	**Symbol grupy lab.** *I3*
Zadanie *Problem 2.*		**Autor** *Mateusz Bąk*	
Rozwiązanie *Algorytm genetyczny*		**Data** *12.01.2019*	

1 Opis problemu

Realizacja projektu polega na implementacji algorytmu metaheurystycznego wyszukującego rozwiązań problemu szeregowania zadań w systemie przepływowym (flow shop) dla liczby maszyn $m = 2$, gdzie funkcją celu jest suma czasów zakończenia wszystkich operacji. Ponadto na drugiej maszynie mogą występować okresy bezczynności ustalane przez algorytm wyszukiwania. Realny czas wykonywania danej operacji na tej maszynie wynosi $(1+k)t_i$, gdzie t_i jest domyślnym czasem przetwarzania operacji drugiej zadania i określonym w instancji problemu, natomiast k jest zmienną o początkowej wartości 0, zwiększaną o stałą wartość po każdej wykonanej operacji na drugiej maszynie i resetowaną po każdym okresie bezczynności. Długość zaplanowanych przerw nie może być mniejsza, niż $1.5\times$ średni czas operacji na drugiej maszynie.

2 Opis metody rozwiązania problemu

Do rozwiązania problemu użyto scharakteryzowanego poniżej algorytmu genetycznego, który zaimplementowano w języku C#.

2.1 Reprezentacja chromosomów

Pojedyncze rozwiązanie opisywane jest przez 4 tablice o rozmiarze równym liczbie zadań n określonej w instancji problemu. Pierwsze dwie tablice (typu int[]) są permutacjami ciągu $1..n$ i określają kolejność wykonywania zadań odpowiednio na maszynach M1 i M2. Kolejna tablica (typu bool[]) służy do przechowywania decyzji o rozpoczęciu przerw technicznych na maszynie M2. Wartość *true* elementu o indeksie j (numerowany od 0) oznacza, że po przetworzeniu j operacji następuje przerwa techniczna. W ostatniej tablicy (typu int[]) przechowywane są długości przerw technicznych (decyzja o przerwie pod indeksem j odpowiada przerwie o długości zapisanej pod tym samym indeksem).

2.2 Selekcja

Do implementacji algorytmu selekcji wykorzystano metodę ruletki. Aby uzyskać efekt minimalizacji funkcji celu, prawdopodobieństwo wyboru rozwiązania i jest proporcjonalne do odwrotności wartości funkcji celu $\mathrm{f}(i)$ podniesionej do potęgi W, gdzie W jest parametrem określającym wagę selekcji – im większa wartość W, tym większa jest rozbieżność prawdopodobieństw p dla rozwiązań różniących się wartością funkcji celu.

$$p_i \sim \left(\frac{1}{\mathrm{f}(i)}\right)^W$$

Uzyskany po normalizacji wzór na prawdopodobieństwo p_i ma zatem postać:

$$p_i = \frac{\left(\frac{1}{\mathrm{f}(i)}\right)^W}{\sum_{k=1}^{N}\left(\frac{1}{\mathrm{f}(k)}\right)^W}$$

gdzie N jest rozmiarem populacji.

Ruletka wykonywana jest L razy, gdzie L jest parametrem określającym limit selekcji. Wylosowane rozwiązania przechodzą do następnej generacji. Liczebność populacji po selekcji może się różnić ze względu na możliwość wielokrotneego wylosowania tego samego rozwiązania (nie jest ono w tym przypadku powielane). Ponadto najlepsze rozwiązanie z populacji jest zawsze wybierane, co oznacza, że rozmiar populacji po selekcji wynosi co najwyżej $L+1$.

Po etapie selekcji na członkach nowego pokolenia stosowane są operatory krzyżowania i mutacji.

2.3 Krzyżowanie

Dla każdego członka populacji z prawdopodobieństwem P_c wybierany jest losowo inny członek populacji do operacji krzyżowania. W procesie krzyżowania tworzony jest nowy członek populacji, początkowo będący kopią pierwszego rodzica. Następnie jedna z czterech tablic kodujących rozwiązanie zostaje losowo wybrana do wstawienia materiału genetycznego pochodzącego od drugiego rodzica. Ze względu na ograniczenia wynikające z konieczności zachowania prawidłowych permutacji dla dwóch pierwszych tablic, szczegóły procesu dziedziczenia są zależne od wybranej tablicy:

- w przypadku tablic określających kolejność wykonywania zadań zastosowano krzyżowanie dwupunktowe polegające na zmianie kolejności elementów w losowo wybranym przedziale na kolejność ich występowania w drugim rodzicu,
- w przypadku pozostałych dwóch tablic zastosowano krzyżowanie dwupunktowe z bezpośrednim kopiowaniem losowo wybranego przedziału z drugiego rodzica.

Możliwy jest również wybór krzyżowania jednopunktowego dla pierwszych dwóch tablic poprzez zmianę odpowiedniego parametru w pliku ustawień metaheurystyki.

2.4 Mutacja

Każdy członek populacji, z wyjątkiem aktualnie najlepszego rozwiązania, może być poddany mutacji z prawdopodobieństwem P_m. Podczas mutacji zostaje losowo wybrana jedna z tablic kodujących rozwiązanie. Szczegóły procesu, tak, jak w przypadku krzyżowania, są zależne od wybranej tablicy:

- w przypadku tablic określających kolejność wykonywania zadań mutacja zamienia ze sobą dwa losowo wybrane elementy tablicy,
- w przypadku pozostałych dwóch tablic mutacja zmienia wartość losowo wybranego elementu.

2.5 Inne operacje

Oprócz opisanych podstawowych operatorów, algorytm może śledzić wiek najlepszego rozwiązania w populacji. Brak poprawy jakości rozwiązania po upływie dużej liczby pokoleń może sugerować, że osiągnięto optimum lokalne, a populacja stała się zbyt jednorodna, żeby pozwolić mutacji na istotną zmianę zakresu przeszukiwania. W tej sytuacji populacja zostaje zastąpiona rozwiązaniami losowymi.

3 Badanie efektywności zastosowanej metody

3.1 Strojenie metaheurystyki

Na podstawie przeprowadzonych testów dokonano strojenia metaheurystyki ze względu na następujące parametry:

- prawdopodobieństwo krzyżowania,
- prawdopodobieństwo mutacji,
- krzyżowanie dwupunktowe/jednopunktowe,
- waga selekcji.

Dla parametrów, których nie dotyczy dany test, przyjęto następujące wartości:

limit czasu działania	10 s
limit pokoleń	1000000
początkowy rozmiar populacji	100
maksymalny czas trwania przerwy technicznej	100
limit wieku	10000
limit selekcji	60
waga selekcji	2
prawdopodobieństwo krzyżowania	0.9
prawdopodobieństwo mutacji	0.15
krzyżowanie dwupunktowe listy operacji	tak

Do strojenia wyznaczono zbiór 50 wygenerowanych instancji. Dla każdej z nich liczba zadań wynosi 50, natomiast różnią się one zakresem długości operacji. Dla każdej badanej wartości parametru uzyskano po jednym rozwiązaniu każdej instancji ze zbioru. Aby porównać wyniki, ze względu na zróżnicowanie instancji, przyjęto najtępujący sposób oceniania:

$$a_S = \frac{\mathrm{f}(S) - \mathrm{f}_R}{\mathrm{f}(S_B) - \mathrm{f}_R}$$

gdzie a_S jest oceną rozwiązania S uzyskanego dla pewnych ustalonych wartości badanego parametru, $\mathrm{f}(S)$ jest wartością funkcji celu tego rozwiązania, f_R jest średnią wartością funkcji celu 1000 rozwiązań wygenerowanych losowo, a $\mathrm{f}(S_B)$ jest wartością funkcji celu najlepszego znanego rozwiązania danej instancji. Wartość a_S nie przekracza 1 i osiąga tą wartość jedynie dla najlepszego znanego rozwiązania instancji. Następnie dla każdej badanej wartości parametru uśredniono uzyskane dla całego zbioru instancji wartości a_S.

$$A = \bar{a}$$

Otrzymane w ten sposób dla każdej badanej wartości parametru liczby można już porównywać oraz zestawić na wykresie (większa wartość oznacza lepsze uzyskane rozwiązania).

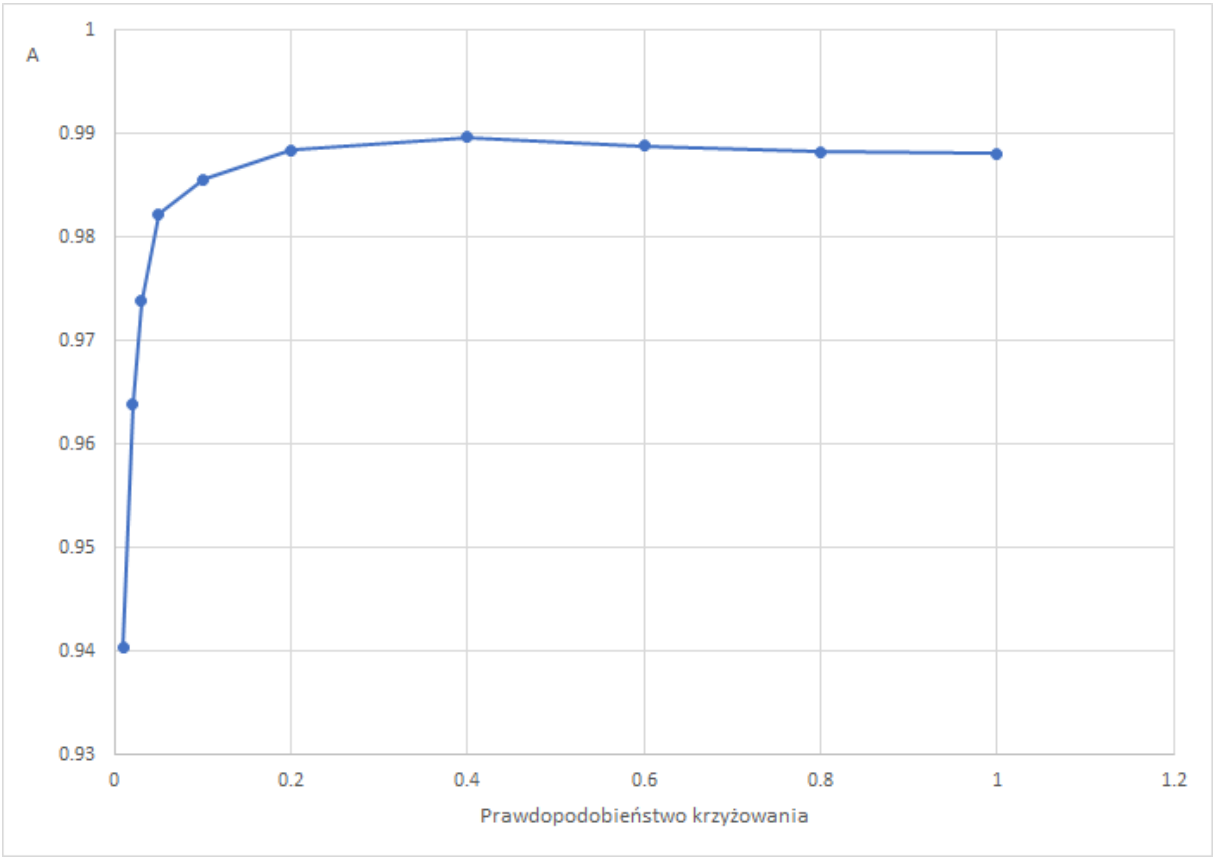
Zależność uzyskanych wyników od wartości prawdopodobieństw krzyżowania oraz mutacji

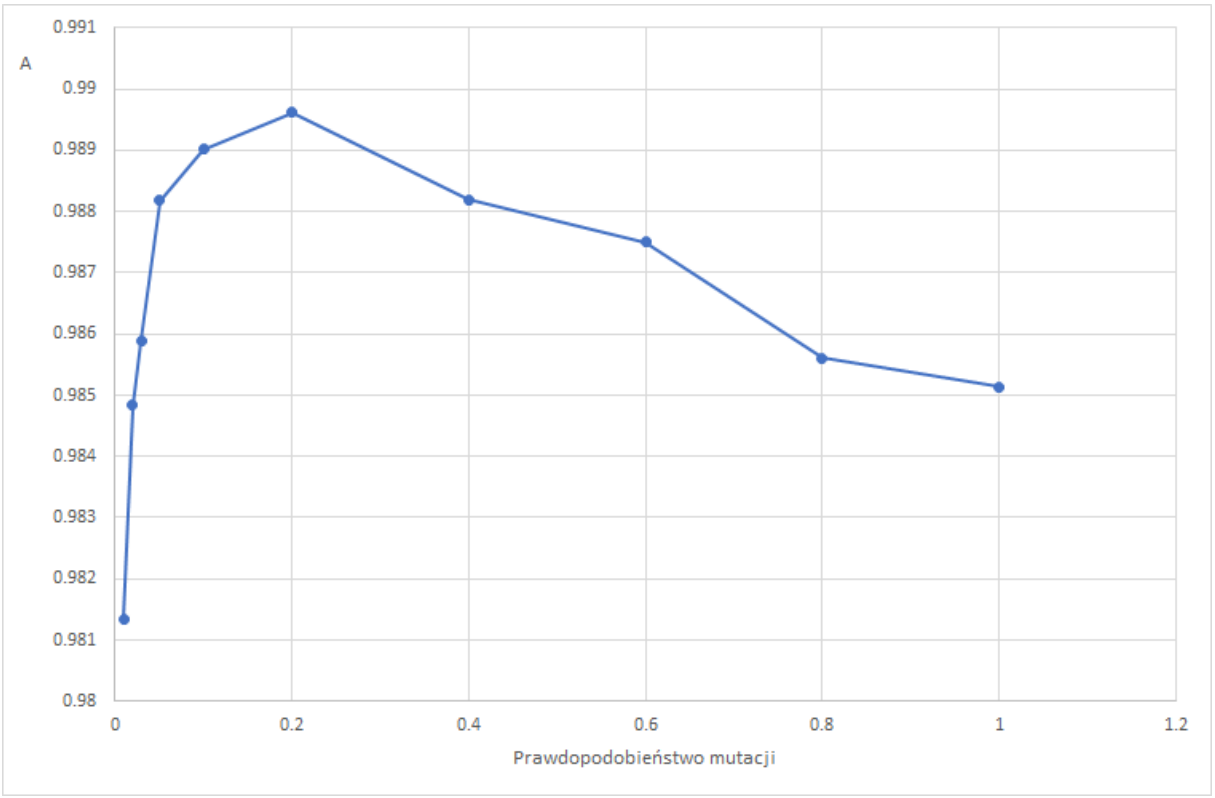
Prawdopodobieństwo krzyżowania i prawdopodobieństwo mutacji to podstawowe paramerty wpływające na skuteczność działania algorytmu genetycznego. Z tego względu przywiązano szczególną wagę do ich strojenia. Poniższa tabela przedstawia wartości miary A dla poszczególnych kombinacji parametrów P_c i P_m:

P_m \ P_c	0	0.01	0.02	0.03	0.05	0.1	0.2	0.4	0.6	0.8	1
0	0.3632	0.4546	0.463	0.4821	0.4856	0.5175	0.5363	0.5751	0.5899	0.6043	0.6245
0.01	0.3719	0.9715	0.9777	0.9806	0.984	0.9826	0.9843	0.9814	0.9797	0.9764	0.9762
0.02	0.3909	0.9699	0.9799	0.9829	0.9845	0.9867	0.9845	0.9848	0.985	0.9804	0.9813
0.03	0.3994	0.9666	0.9799	0.9838	0.9867	0.9881	0.9868	0.9859	0.9844	0.9851	0.9847
0.05	0.4192	0.959	0.9797	0.9834	0.9853	0.9887	0.9893	0.9882	0.9852	0.9853	0.9834
0.1	0.4366	0.9489	0.9741	0.9817	0.9837	0.9882	0.9889	0.989	0.9888	0.9864	0.985
0.2	0.4565	0.9403	0.9639	0.9738	0.9822	0.9855	0.9884	**0.9896**	0.9888	0.9882	0.988
0.4	0.4704	0.9274	0.9584	0.971	0.9747	0.983	0.9882	0.9882	0.9883	0.9885	0.9883
0.6	0.4819	0.9206	0.9562	0.9649	0.978	0.9829	0.9849	0.9875	0.9871	0.9869	0.9869
0.8	0.4875	0.9233	0.952	0.9655	0.9729	0.9811	0.9846	0.9856	0.9864	0.9884	0.9872
1	0.4935	0.9173	0.9533	0.9609	0.9707	0.9796	0.9816	0.9851	0.9874	0.9884	0.9864

Analizując dane z tabeli można zauważyć, że najwyższą skuteczność uzyskano dla wartości $P_c = 0.4$ oraz $P_m = 0.2$. Wyniki w pierwszym wierszu oraz w pierwszej kolumnie ($P_c = 0$ lub $P_m = 0$) wyraźnie odbiegają od pozostałych. Oznacza to, że zarówno krzyżowanie, jak i mutacja odgrywają pewną rolę w działaniu algorytmu, czego obrazem są względnie słabe wyniki przy braku jednego z operatorów.

Na poniższych wykresach przedstawiono zależności wartości miary A od prawdopodobieństwa krzyżowania przy ustalonym $P_m = 0.2$ oraz od prawdopodobieństwa mutacji przy ustalonym $P_c = 0.4$:





Obydwie zależności przejawiają wyraźny wzrost skuteczności w przedziale $[0;0.2]$, a następnie powolny, ale monotoniczny spadek w przedziale $[0.2;1]$. Zgodnie z przewidywaniami zarówno skrajnie duże, jak i skrajnie małe wartości prawdopodobieństw operacji genetycznych nie prowadziły do uzyskania najlepszych wyników. Dzięki doświadczalnej analizie udało się wyznaczyć takie wartości tych parametrów, dla których skuteczność algorytmu była największa.

Zależność uzyskanych wyników od metody krzyżowania

Drugim przeprowadzonym testem było badanie wpływu metody krzyżowania na jakość uzyskanych rozwiązań. Po uśrednieniu uzyskano następujące wyniki:

- krzyżowanie jednopunktowe
  $A = 0.992$
- krzyżowanie dwupunktowe
  $A = 0.988$

Przedstawiony algorytm rozwiązuje badany problem lepiej w przypadku zastosowania krzyżowania jednopunktowego. Może być to związane z faktem, że metoda ta bezpośrednio kopiuje z przodka jedynie początkowy fragment tablicy uszeregowania operacji, natomiast dla rozważanej funkcji celu szczególnie istotne jest dobre uporządkowanie właśnie na początku.

Zależność uzyskanych wyników od wartości wagi selekcji

Strojeniu poddano również wartość wagi selekcji. Wraz ze wzrostem wartości parametru W selekcja przypisuje coraz wyższe wartości prawdopodobieństwa rozwiązaniom dominującym, zmniejszając szanse pozostałych rozwiązań na przejście do następnej generacji.

W	0.25	0.5	0.75	1	1.25	1.5	1.75	2	2.25	2.5
A	0.9841	0.9853	0.9868	0.986	0.988	0.9882	**0.9892**	0.9883	0.9882	0.9864

Najlepsze rezultaty osiągnięto dla wartości $W = 1.75$. Odpowiednio dobrana skala w przydzielaniu fragmentów koła ruletki może zatem przyczynić się do poprawy jakości rozwiązań.

Dane z tabeli ilustruje poniższy wykres:

Skutki przeprowadzonego strojenia

Działanie algorytmu przetestowano na 600 instancjach problemu o różnych parametrach, uzyskując dwa zestawy rozwiązań – jeden dla domyślnych parametrów metaheurystyki przed strojeniem, a drugi dla wartości parametrów uzyskanych w procesie strojenia. Porównując je odczytano średnią poprawę wartości funkcji celu otrzymanych rozwiązań, która wyniosła 0.26%. Dla większości testowanych instancji wynik uzyskany po strojeniu był niegorszy od wyniku uzyskanego przed strojeniem.

3.2 Badanie efektywności dla różnych parametrów instancji

Wykorzystując parametry generatora instancji przetestowano zależności uzyskiwanych wyników od właściwości rozwiązywanych instancji problemu, takich, jak ilość zadań oraz ich długość. Parametry metaheurystyki dostosowano na podstawie wyników przeprowadzonego wcześniej strojenia:

limit czasu działania	10 s
limit pokoleń	1000000
początkowy rozmiar populacji	100
maksymalny czas trwania przerwy technicznej	100
limit wieku	10000
limit selekcji	60
waga selekcji	1.75
prawdopodobieństwo krzyżowania	0.4
prawdopodobieństwo mutacji	0.2
krzyżowanie dwupunktowe listy operacji	nie

Dla każdej badanej wartości parametru wygenerowano 20 różnych instancji problemu, z których każda była następnie rozwiązywana jednokrotnie przez 10 s. Za miarę jakości rozwiązania przyjęto stosunek średniej wartości funkcji celu 1000 rozwiązań wygenerowanych losowo do wartości funkcji celu otrzymanego rozwiązania:

$$b_S = \frac{\mathrm{f}_R}{\mathrm{f}(S)}$$

Następnie dla każdej badanej wartości parametru uśredniono uzyskane dla dotyczącego jej zbioru instancji wartości b_S.

$$B = \bar{b}$$

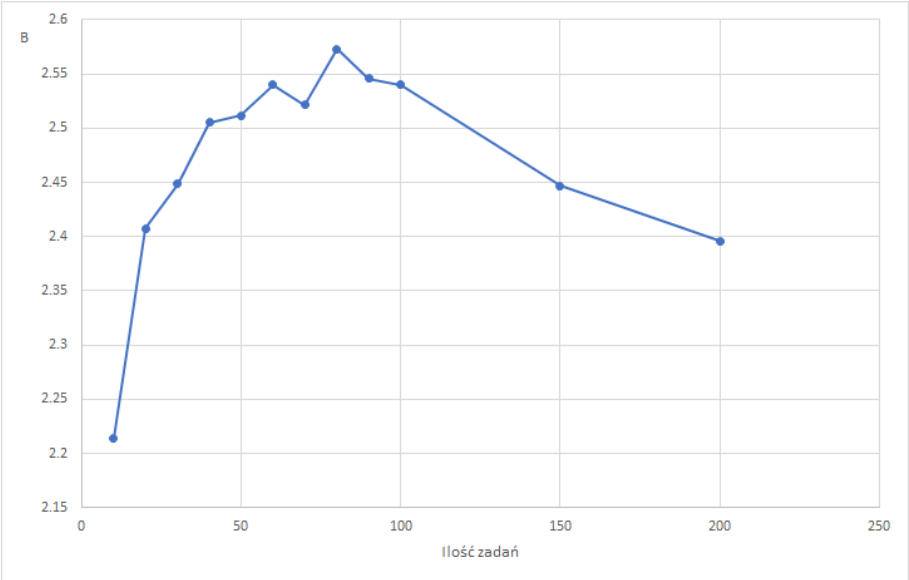
Miarę B można interpretować jako średni współczynnik poprawy jakości rozwiązania względem rozwiązań losowych.

Dla parametrów, których nie dotyczy dany test, przyjęto następujące wartości:

ilość zadań	50
minimalny czas trwania operacji	5
maksymalny czas trwania operacji	40

Zależność uzyskanych wyników od ilości zadań

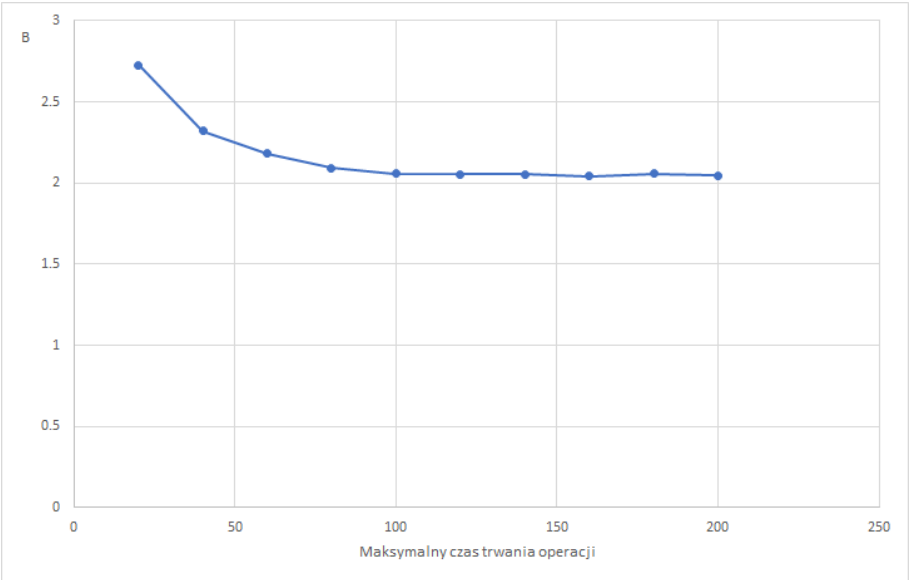
Pierwszy test przeprowadzono na 240 instancjach problemu różniących się ilością zadań w zakresie od 10 do 200. Wyniki przedstawiono na wykresie poniżej.



Dla każdej badanej wartości parametru uzyskana wartość funkcji celu jest ponad dwukrotnie niższa, niż średnia dla rozwiązań losowych. Współczynnik osiąga wartość przekraczającą 2.5 dla 40 zadań, natomiast po przekroczeniu 100 zadań zaczyna spadać. Największą poprawę wyników uzyskano dla instancji z ilością zadań w przedziale od 50 do 100, co może być konsekwencją przeprowadzonego strojenia, do którego użyto instancji z ilością zadań równą 50.

Zależność uzyskanych wyników od długości zadań

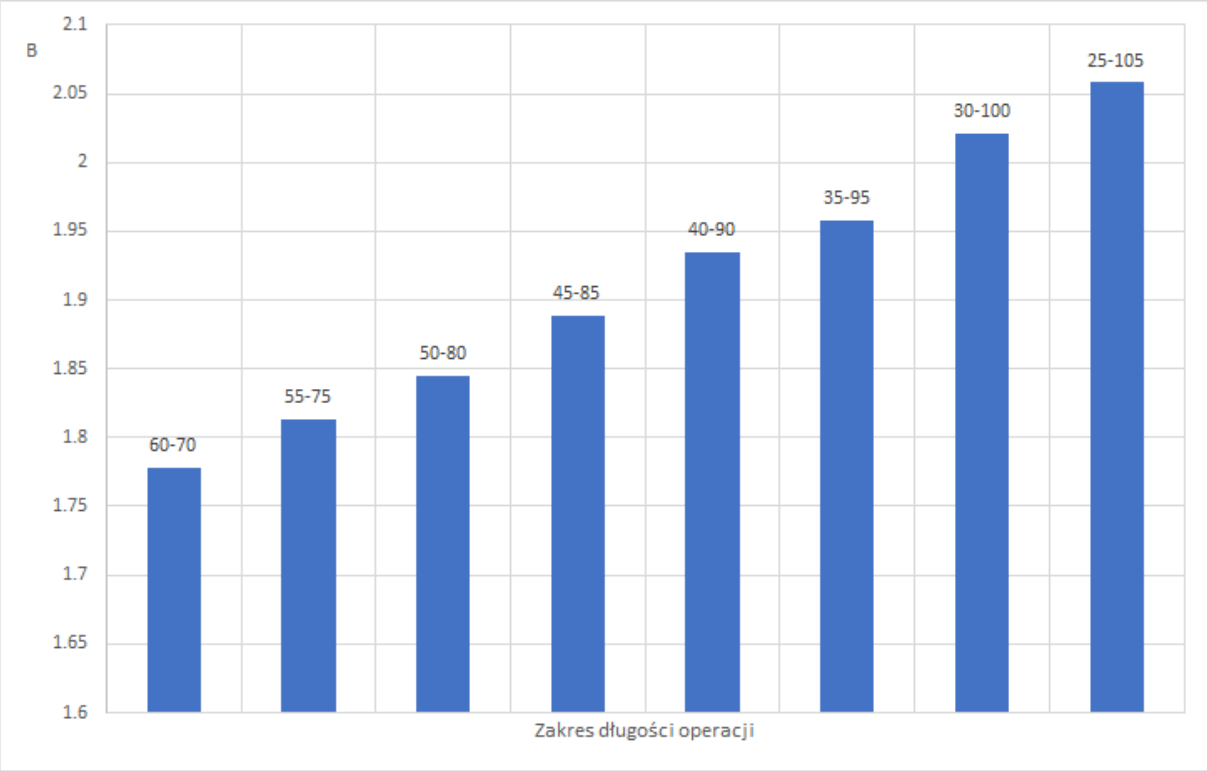
W kolejnym teście zostały porównane rozwiązania instancji różniących się średnią długością zadań. Dla każdego kolejnego zestawu instancji maksymalna długość operacji była zwiększana o 20, zaczynając od 20, natomiast minimalna długość operacji wynosi $\frac{1}{4}$ długości maksymalnej.



W przedziale do wartości 100 badana zależność jest wyraźnie malejąca, natomiast dla dłuższych zadań wartość współczynnika B utrzymuje się blisko 2.

Zależność uzyskanych wyników od rozbieżności w długości zadań

Ostatni test polega na zestawieniu ze sobą instancji różniących się przedziałem wartości długości operacji, przy jednoczesnym zachowaniu średniej długości operacji równej dla wszystkich zestawów. Na wykresie zaznaczono badane przedziały wraz z odpowiadającymi wartościami współczynnika B:



Dla rosnącej wielkości przedziału wzrasta wartość B, co w kontekście rozważanego problemu może być rezultatem niskiego zróżnicowania wartości funkcji celu w przypadku, gdy wszystkie operacje mają zbliżoną długość. W konsekwencji rozwiązania losowe różnią się od optymalnych zdecydowanie mniej, niż w przypadku większej rozbieżności, co pozwala metaheurystyce uzyskać większą względną poprawę.

4 Wnioski

Algorytm genetyczny jest w stanie bardzo szybko dostarczyć dobrej jakości rozwiązania dla badanego problemu. Ze względu na wielkość przestrzeni przeszukiwania może on dochodzić do różnych wyników, w zależności od przydzielonego czasu na rozwiązanie, jednak dla prostszych instancji, po przeprowadzonym strojeniu, zwracane rozwiązania są najczęściej takie same, być może optymalne. Największą trudnością okazało się być strojenie, a dokładniej odpowiedni dobór miary umożliwiającej porównywanie wyników działania programu, która w dostatecznym stopniu odzwierciedlałaby jakość uzyskanych rozwiązań dla poszczególnych wartości badanych parametrów. Przeprowadzone testy pokazały jednak poprawę wyników działania algorytmu, co sugeruje, że udało się wybrać odpowiednie miary. Niezależnie od rozwiązywanej instancji, algorytm uzyskiwał około dwukrotną poprawę wartości funkcji celu względem rozwiązań wygenerowanych losowo.