

# Projektowanie Efektywnych Algorytmów

## Projekt

27/01/2023

259198 Lena Kuźma

### 1) Algorytm genetyczny

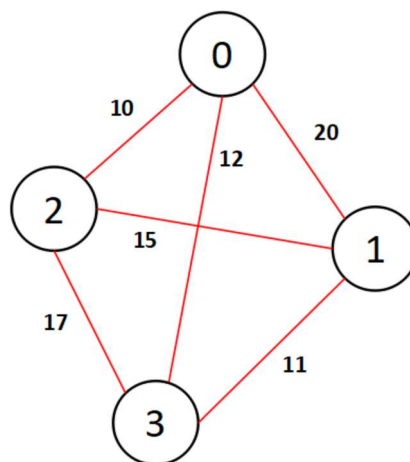
<i>spis treści</i>	<i>strona</i>
<i>Sformułowanie zadania</i>	2
<i>Metoda</i>	3
<i>Algorytm</i>	5
<i>Dane testowe</i>	6
<i>Procedura badawcza</i>	7
<i>Wyniki</i>	8
<i>Analiza wyników i wnioski</i>	19

## 1. Sformułowanie zadania

Zadanie polega na opracowaniu i implementacji algorytmu rozwiązywania problemu komiwojażera w oparciu o algorytm genetyczny. Problem komiwojażera (*eng. TSP – Travelling Salesman Problem*) polega na znalezieniu minimalnego cyklu Hamiltona (każdy wierzchołek grafu odwiedzany jest dokładnie raz) w pełnym grafie ważonym (*Rysunek 1*).

W zadaniu należy zbadać wpływ, na jakość i czas uzyskiwanych rozwiązań oraz zużycia pamięci:

- wielkości populacji,
- warunku stopu,
- prawdopodobieństwa krzyżowania,
- prawdopodobieństwa mutacji,
- metody selekcji,
- metody sukcesji,
- metody krzyżowania,
- metody mutacji.



*Rysunek 1: Graf pełny ważony - symboliczna reprezentacja problemu komiwojażera*

## 2. Metoda

Algorytm genetyczny należy do grupy algorytmów ewolucyjnych (*ang. Evolutionary Algorithms*). Są one szeroko stosowaną heurystyką przeszukiwania i optymalizacji opartą na zasadach przejętych z teorii ewolucji. Algorytmy ewolucyjne nie gwarantują znalezienia optimum globalnego, jednak zapewniają znalezienie rozwiązania wystarczająco dobrego w akceptowalnym przedziale czasu. Cechą algorytmów ewolucyjnych, odróżniającą je od innych metod optymalizacji jest przetwarzanie populacji rozwiązań, prowadzące do równoległego przeszukiwania przestrzeni rozwiązań z różnych punktów.

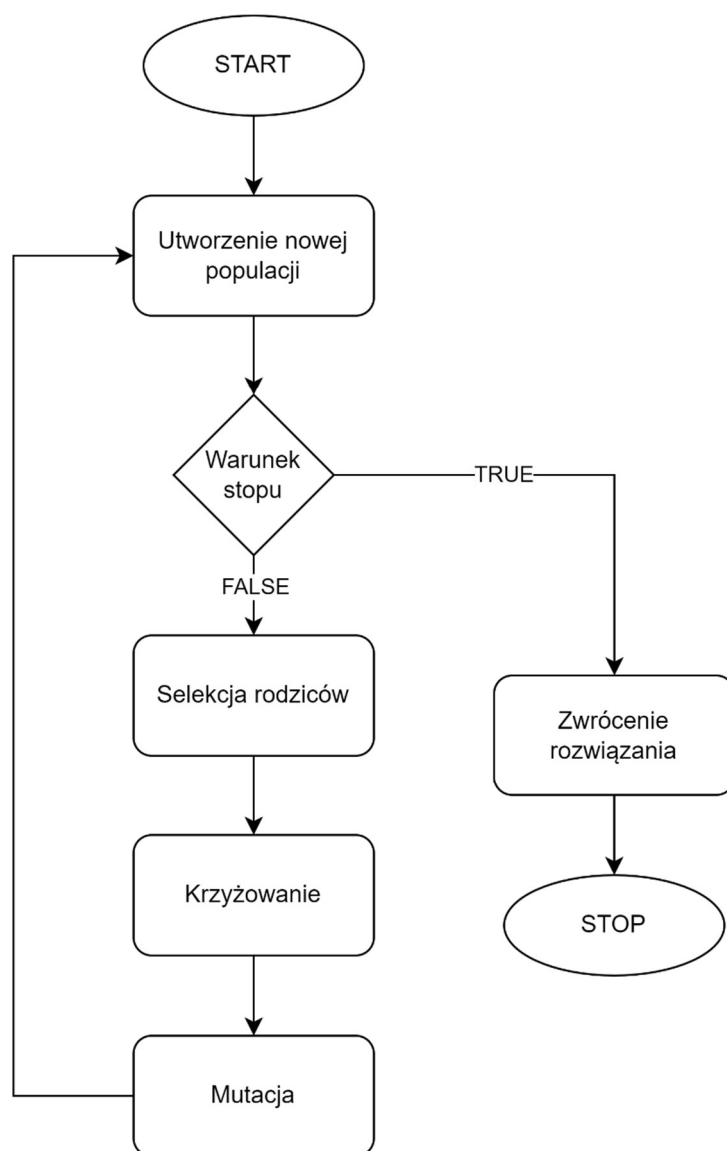
Algorytm genetyczny jest iteracyjną procedurą poprawiania rozwiązań zawartych w populacji. W przypadku problemu komiwojażera, chromosomem nazywamy pojedynczy cykl Hamiltona (ciąg kolejnych wierzchołków), a populacją zbiór chromosomów o zadanej liczbie.

Ewolucyjny rozwój populacji chromosomów odbywa się poprzez mechanizm reprodukcji, na który składają się procesy selekcji rodziców, krzyżowania, mutacji oraz sukcesji. W procesie krzyżowania, z dwóch chromosomów rodzicielskich wybierane są geny, które po zespoleniu tworzą jeden lub więcej chromosomów potomnych. W procesie mutacji dochodzi do przekłamania kodu poprzez zmianę jednego genu lub ich ciągu. Operacja mutacji powinna być stosowana stosunkowo rzadko (mutacji powinno podlegać znacznie mniej osobników, niż krzyżowaniu). Przy pomocy tych mechanizmów tworzą się kolejne pokolenia (populacje chromosomów), zawierające coraz lepsze osobniki.

Na działanie algorytmu genetycznego mają wpływ:

- zadana wielkość populacji,
- warunek stopu – zastosowano: liczbę generacji / liczbę iteracji bez poprawy wyniku / osiągnięcie wymaganego progu błędu / przekroczenie zadanego czasu,
- utworzenie populacji początkowej – zastosowano: algorytm *greedy* / *random*,  
Algorytm *greedy* polega na wybieraniu lokalnie najkrótszej ścieżki. Nie zwraca rozwiązania optymalnego, jednak w stosunkowo krótkim czasie uzyskujemy dzięki niemu rozwiązanie znacznie lepsze od losowego.
- selekcja – zastosowano: metodę rankingową / turniejową,  
Metoda rankingowa polega na wyborze zadanej liczby najlepszych osobników. Metoda turniejowa polega na wylosowaniu pary osobników, z których lepszy zostaje wybierany do krzyżowania. Turniej trwa do momentu stworzenia wszystkich par.
- krzyżowanie – zastosowano: algorytm *Order Crossover (OX)*,  
Algorytm *OX* buduje potomstwo przez wybór podciągu z rozwiązania i zachowanie względnej kolejności wierzchołków z drugiego rodzica.
- mutacja – zastosowano: inwersję / 2-zamianę,  
2-zamiana polega na wylosowaniu dwóch wierzchołków i zamianie ich miejscami. Metodę inwersji zaczynamy podobnie od wylosowania dwóch wierzchołków. Następnie odwracamy kolejność łuku zaczynającego się w jednym wylosowanym wierzchołku, a kończącego się w drugim.
- sukcesja (stworzenie nowej populacji) – zastosowano: metoda rankingowa  
Z dotychczasowej populacji oraz nowo stworzonych potomków do nowej populacji zostają wybierane najlepsze osobniki.

## 2. Algorytm



Rysunek 2: Schemat blokowy przedstawiający działanie algorytmu genetycznego.

### 3. Dane testowe

Do sprawdzenia poprawności działania algorytmu oraz do dostrojenia (określenia wartości parametrów algorytmu) wybrano następujący zestaw instancji:

ftv70.atsp

lin318.tsp

u1432.tsp

Źródło: [TSPLIB \(uni-heidelberg.de\)](http://tsplib.uni-heidelberg.de)

Do wykonania badań wybrano następujący zestaw instancji:

ftv70.atsp

ftv170.atsp

lin318.tsp

pcb442.tsp

u724.tsp

pr1002.tsp

pcb1173.tsp

pr2392.tsp

Źródło: [TSPLIB \(uni-heidelberg.de\)](http://tsplib.uni-heidelberg.de)

#### 4. Procedura badawcza

Procedura badawcza polegała na uruchomieniu programu (napisanego w języku *python*) sterowanego plikiem inicjującym *.INI* (format pliku: nazwa instancji, liczba wykonań rozwiązania, nazwa pliku wyjściowego, najlepsze znane rozwiązanie (*Rysunek 3*)).

```
[FILE]
file1 = ftv170.atsp
file2 = lin318.tsp

[ITERATOR]
i1 = 10
i2 = 10

[OUTPUT]
file1 = GA.xlsx

[OPTIMAL_VALUE]
opt1 = 2755
opt2 = 42029
```

Rysunek 3: Przykładowa zawartość pliku inicjalizującego *.INI*

Każda z instancji rozwiązywana była zadaną liczbę razy, np. *lin318.tsp* wykonana została 10 razy. Do pliku wyjściowego *GA.xlsx* zapisywany był czas wykonania, otrzymane rozwiązanie (koszt ścieżki) oraz jego błąd procentowy. Czas wykonywania algorytmu został zmierzony przy pomocy funkcji *perf\_counter()* z biblioteki *time*. Plik wyjściowy zapisywany był w formacie *xlsx*. Na *Rysunku 4* przedstawiono fragment zawartości przykładowego pliku wyjściowego.

	A	B	C	D	E	F
1	Nazwa instancji		Liczba powtórzeń		Wartość optymalna	
2	ftv170.atsp		3		2755	
3						
4	Czas wykonywania [s]:		Uzyskany koszt:		Błąd [%]:	
5	0,456962		3499		27,00544	
6	0,439751		3493		26,78766	
7	0,436858		3511		27,44102	

Rysunek 4: fragment pliku wyjściowego w formacie *.xlsx*

Wyniki opracowane zostały w programie MS Excel.

## 5. Wyniki

❖ Badanie wpływu prawdopodobieństwa krzyżowania  $p_k$  na czas wykonywania algorytmu.

Założenia:

Warunek stopu - osiągnięcie wymaganego progu błędu względnego:

$$\delta = \frac{\Delta x}{x} \cdot 100\% = \frac{|x - x_0|}{x} \cdot 100\%$$

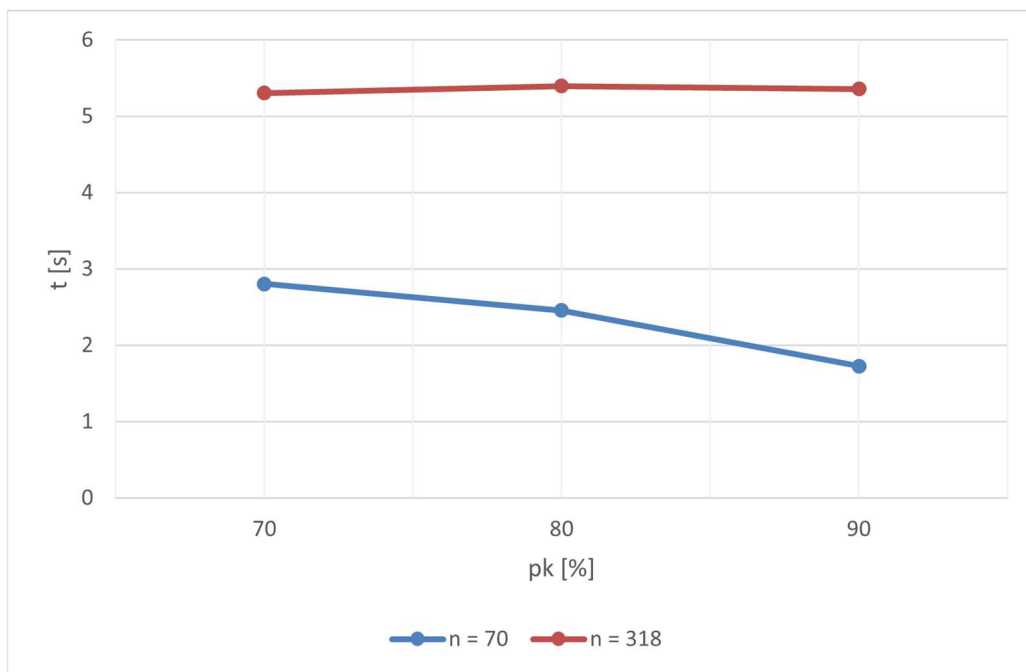
Wielkość instancji $n$	70	318	1432
Próg błędu $\delta$	5%	50%	150%

Tabela 1: Progi błędu rozwiązania, zastosowane do badania wpływu  $p_k$ ,  $p_m$  oraz wielkości populacji na czas wykonywania algorytmu.

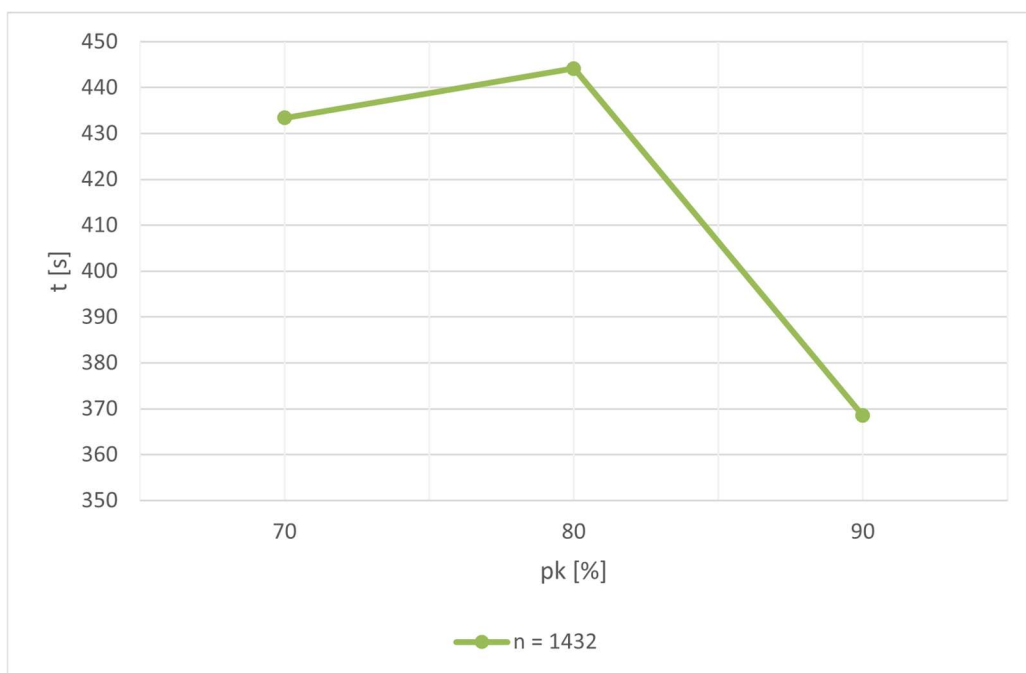
- Populacja początkowa – algorytm *greedy*
- Prawdopodobieństwo mutacji – 0.01
- Metoda krzyżowania – OX
- Wielkość populacji – 50
- Metoda selekcji – rankingowa
- Metoda mutacji – inwersja
- Metoda sukcesji – ranking

Testy zostały przeprowadzone dla trzech wartości prawdopodobieństwa  $p_k$ : 70%, 80% i 90%. Dla instancji  $n = 318$  zmiana prawdopodobieństwa krzyżowania nie wpłynęła znacząco na czas wykonywania algorytmu  $t$  (Rysunek 5). Natomiast można zauważyć, że dla pozostałych badanych instancji (Rysunek 5 i 6)  $t$  malało ze wzrostem  $p_k$ . Z tego powodu wartość  $p_k = 90\%$  została uznana za najlepszą i wykorzystana w późniejszych badaniach.





Rysunek 5: Wykres zależności czasu wykonywania algorytmu od wielkości prawdopodobieństwa krzyżowania dla instancji  $n = 70$  i  $n = 318$ .



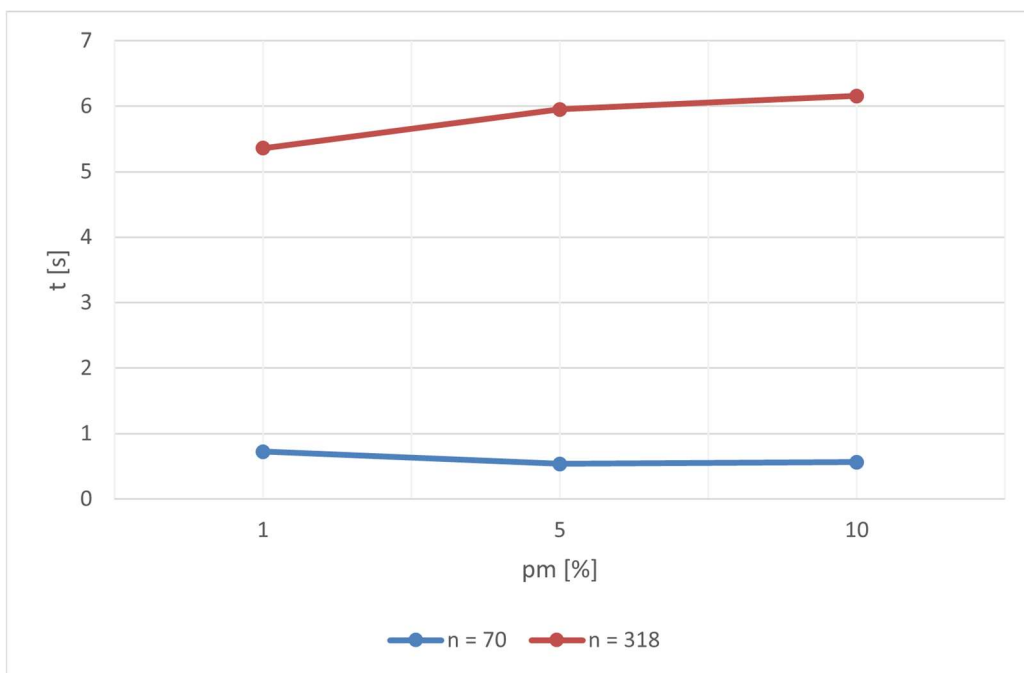
Rysunek 6: Wykres zależności czasu wykonywania algorytmu od wielkości prawdopodobieństwa krzyżowania dla instancji  $n = 1432$ .

❖ Badanie wpływu prawdopodobieństwa mutacji  $pm$  na czas wykonywania algorytmu.

Założenia:

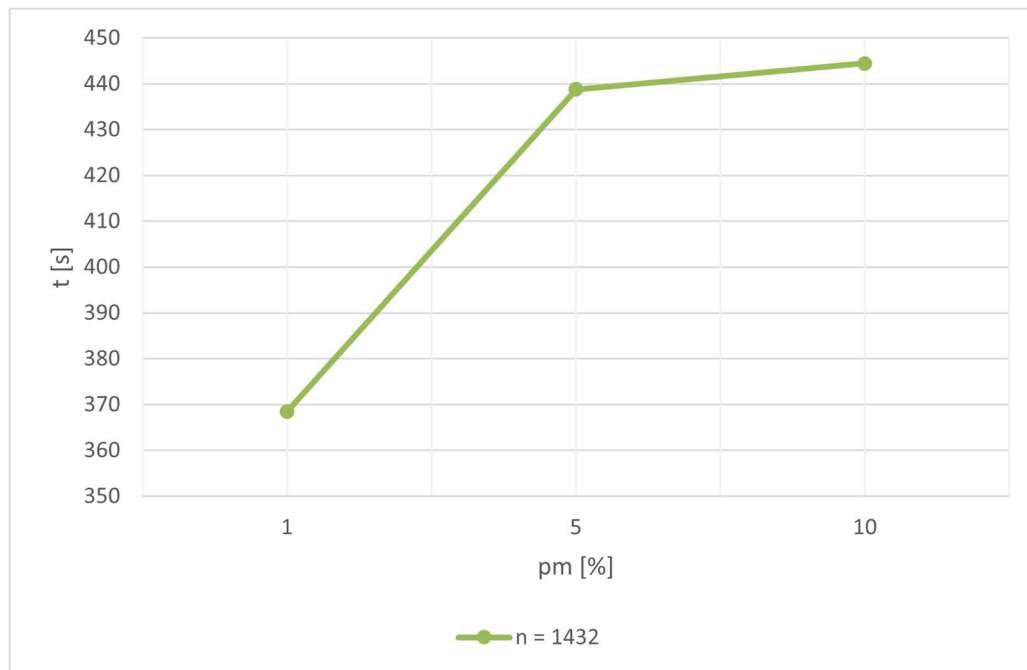
- Populacja początkowa – algorytm *greedy*
- Prawdopodobieństwo krzyżowania – 0.9
- Metoda krzyżowania – *OX*
- Wielkość populacji – 50
- Metoda selekcji – rankingowa
- Metoda mutacji – inwersja
- Metoda sukcesji – ranking

Testy zostały przeprowadzone dla trzech wartości prawdopodobieństwa mutacji  $pm$ : 1%, 5% oraz 10%. Z Rysunku 7 i 8 wynika, że czas wykonywania algorytmu wzrastał wraz ze wzrostem wartości  $pm$ . Najlepsze wyniki zostały uzyskane dla  $pm = 1\%$ , dlatego ta wartość została wykorzystana w późniejszych badaniach.



Rysunek 7: Wykres zależności czasu wykonywania algorytmu od wielkości prawdopodobieństwa mutacji

dla instancji  $n = 70$  i  $n = 318$ .



Rysunek 8: Wykres zależności czasu wykonywania algorytmu od wielkości prawdopodobieństwa mutacji

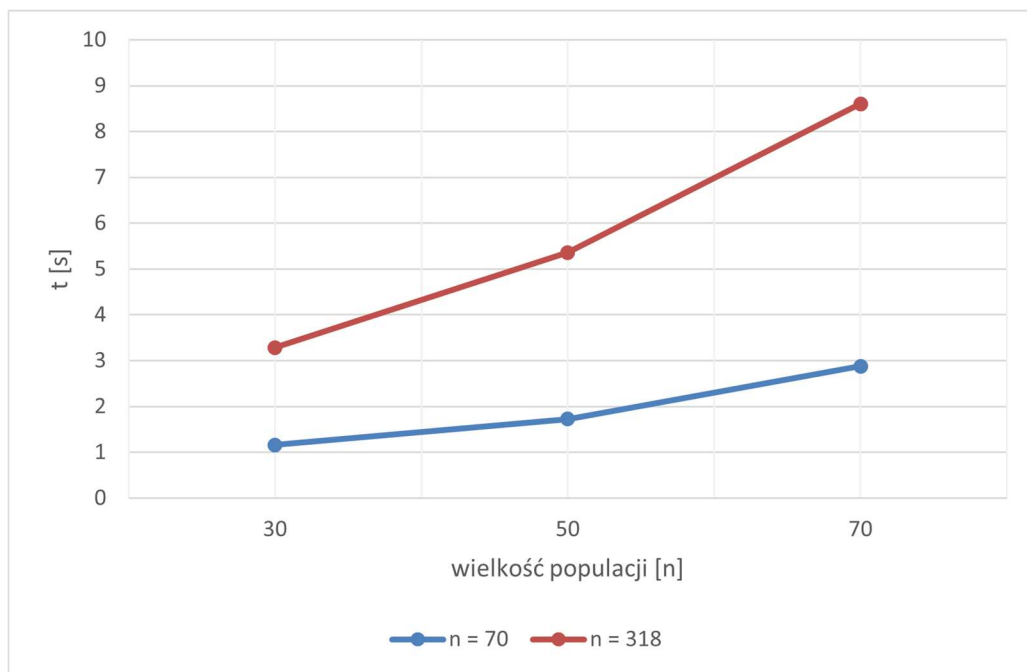
dla instancji  $n = 1432$ .

❖ Badanie wpływu wielkości populacji na czas wykonywania algorytmu.

Założenia:

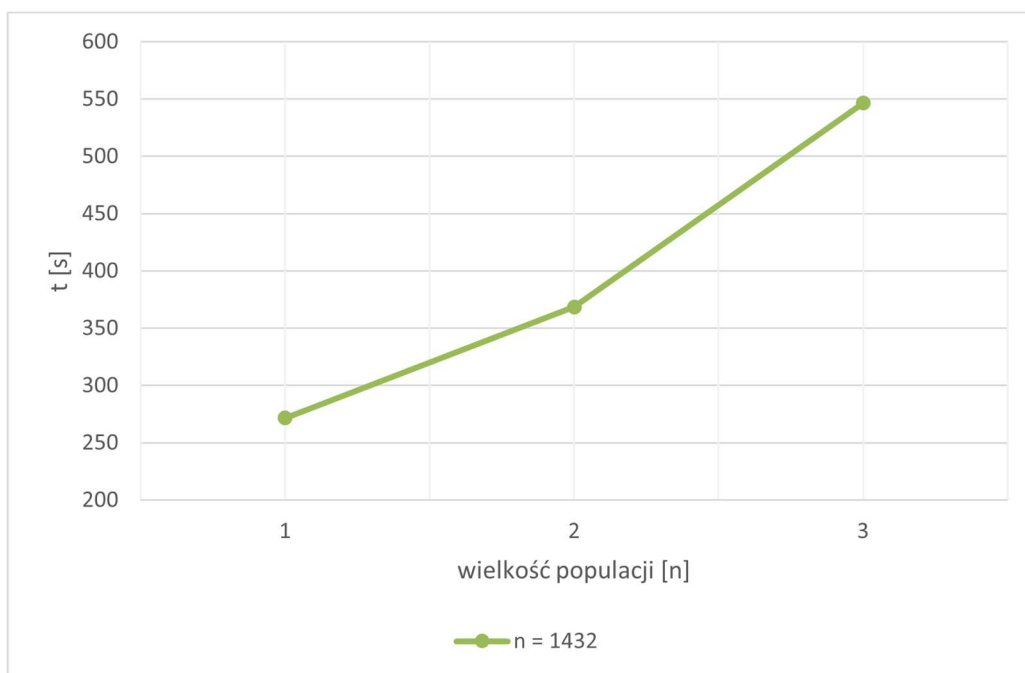
- Populacja początkowa – algorytm *greedy*
- Prawdopodobieństwo krzyżowania – 0.9
- Prawdopodobieństwo mutacji – 0.01
- Metoda krzyżowania – OX
- Metoda selekcji – rankingowa
- Metoda mutacji – inwersja
- Metoda sukcesji – ranking

Testy zostały przeprowadzone dla trzech wielkości populacji: 30, 50 oraz 70. Z Rysunków 9 i 10 można wywnioskować, że najlepszą z testowanych wielkości populacji jest 30, ponieważ czas w jakim uzyskujemy dla niej satysfakcjonujące rozwiązanie jest najmniejszy. Wartość ta została zastosowana w późniejszych badaniach.



Rysunek 9: Wykres zależności czasu wykonywania algorytmu od wielkości populacji

dla instancji  $n = 70$  i  $n = 318$ .



Rysunek 10: Wykres zależności czasu wykonywania algorytmu od wielkości populacji

dla instancji  $n = 1432$ .

❖ Badanie wpływu warunku stopu na czas wykonywania algorytmu.

W ramach badania zostały przetestowane trzy warunki stopu:

- Liczba iteracji bez poprawy wyniku ( $i = 20$ )
- Liczba generacji / iteracji ( $i = 500$ )
- Czas ( $t = 120s$ )

Warunek stopu	Średni czas wykonywania algorytmu $t$ [s]
Brak poprawy wyniku	0,576
Liczba iteracji	0,602
Czas	0,648

Tabela 2: Zależność czasu wykonywania algorytmu od zastosowanego warunku stopu dla instancji  $n = 70$ .

Warunek stopu	Średni czas wykonywania algorytmu $t$ [s]
Brak poprawy wyniku	6,484
Liczba iteracji	6,669
Czas	6,622

Tabela 3: Zależność czasu wykonywania algorytmu od zastosowanego warunku stopu dla instancji  $n = 318$ .

Warunek stopu	Średni czas wykonywania algorytmu $t$ [s]
Brak poprawy wyniku	520,891
Liczba iteracji	529,923
Czas	530,586

Tabela 4: Zależność czasu wykonywania algorytmu od zastosowanego warunku stopu dla instancji  $n = 1432$ .

Analizując Tabele 2-4 można stwierdzić, że najkorzystniejszym warunkiem stopu do przeprowadzania dalszych badań jest brak poprawy wyniku po określonej liczbie iteracji. Czas uzyskiwania rozwiązania z błędem poniżej zadanego progu był najkrótszy dla tej metody.

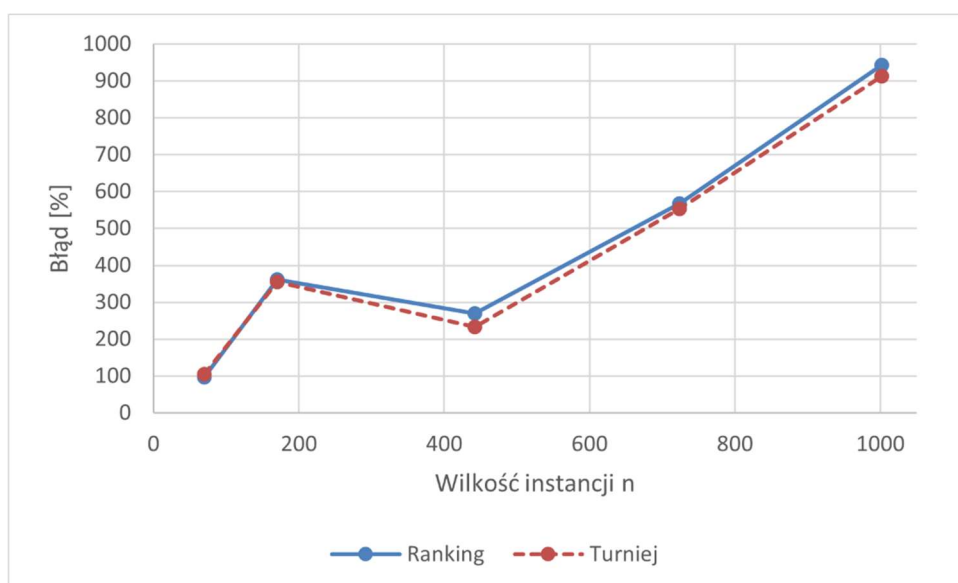
W trakcie przeprowadzania testów można było zauważyć, że w końcowej fazie przeszukiwań efektywność algorytmu spadała – otrzymywane rozwiązania nie poprawiały się. Przerwanie wykonywania algorytmu po zadanej liczbie iteracji bez poprawy wyniku eliminuje dodatkowy czas przeszukiwań spowodowany tym zjawiskiem.

❖ Badanie wpływu metody selekcji na czas i jakość wykonywania algorytmu.

Założenia:

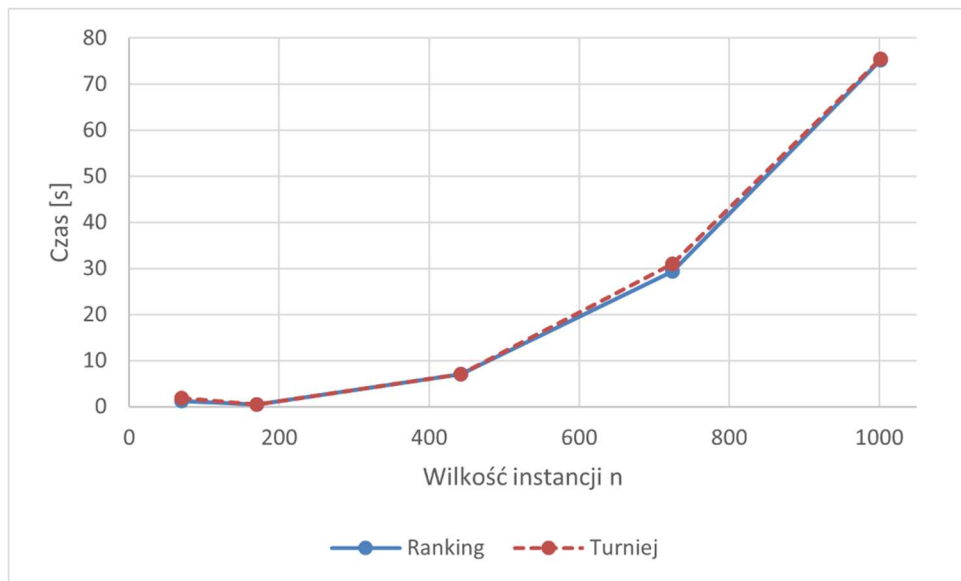
- Populacja początkowa – wybór losowy
- Prawdopodobieństwo krzyżowania – 0.9
- Prawdopodobieństwo mutacji – 0.01
- Metoda krzyżowania – *OX*
- Wielkość populacji – 30
- Metoda mutacji – inwersja
- Metoda sukcesji – ranking

Badane były dwie metody selekcji: rankingowa oraz turniejowa. *Rysunki 11 i 12* przedstawiają porównanie tych metod pod względem wielkości błędu oraz czasu wykonywania algorytmu. Można zauważyć, że obie zwracają bardzo zbliżone wyniki. Mimo to metoda turniejowa dla większości testowanego zakresu wielkości instancji przyjmuje wartości błędu mniejsze od metody rankingowej. Sugeruje to, że może być ona korzystniejsza.



Rysunek 11: Wykres zależności błędów rozwiązania od wielkości instancji dla dwóch metod selekcji

(warunek stopu: czas  $t = 120s$ ).



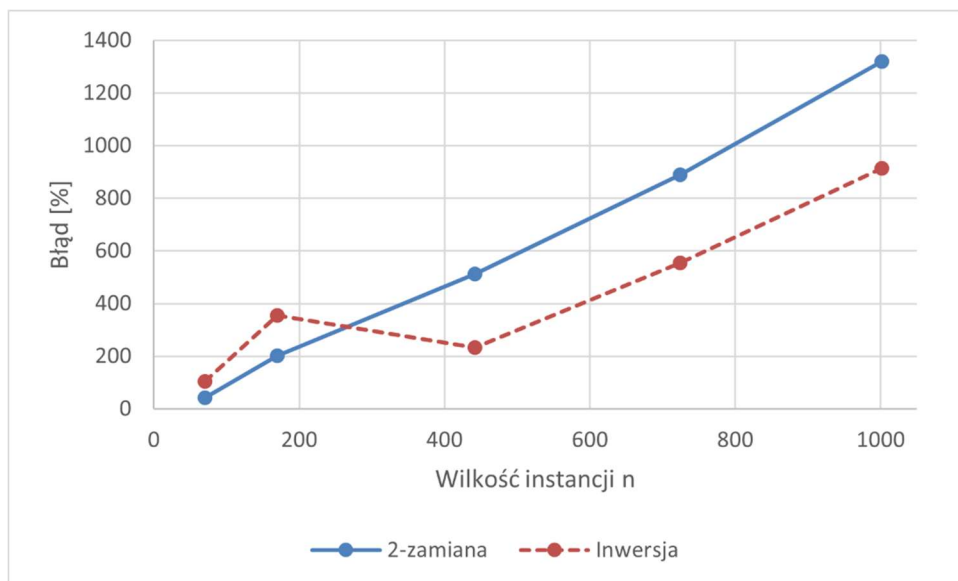
Rysunek 12: Wykres zależności czasu wykonywania algorytmu od wielkości instancji dla dwóch metod selekcji (warunek stopu: błąd  $< \delta$  - Tabela1 ).

❖ Badanie wpływu metody mutacji na czas i jakość wykonywania algorytmu.

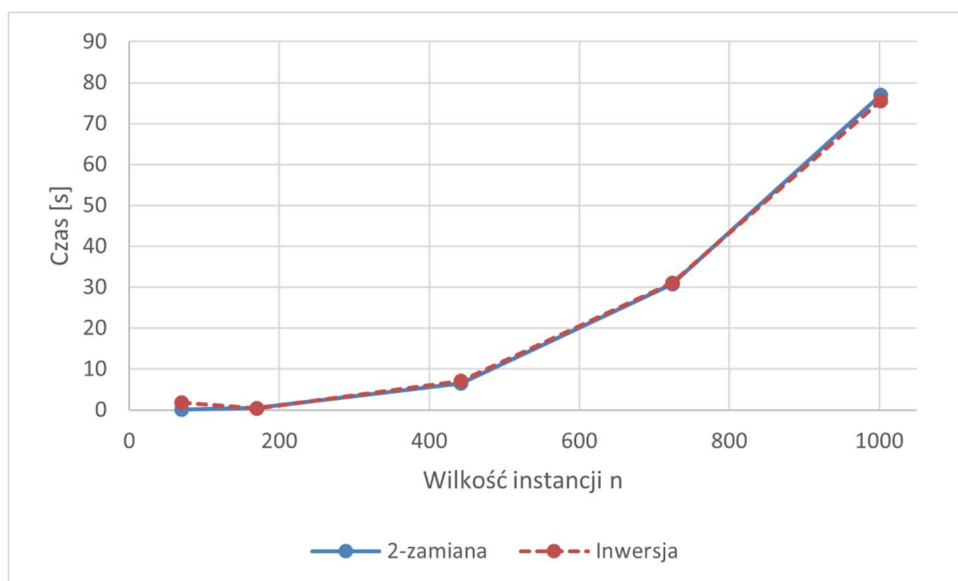
Założenia:

- Populacja początkowa – wybór losowy
- Prawdopodobieństwo krzyżowania – 0.9
- Prawdopodobieństwo mutacji – 0.01
- Metoda krzyżowania – OX
- Wielkość populacji – 30
- Metoda selekcji – turniejowa
- Metoda sukcesji – ranking

Badaniom zostały poddane dwie metody mutacji osobników: metoda 2-zamiany oraz inwersji. Na Rysunku 3 można zauważyć, że błąd uzyskiwanego wyniku dla instancji mniejszych od 170 jest mniejszy przy algorytmie 2-zamiany. Natomiast ze wzrostem wielkości instancji to metoda inwersji zwraca znacząco mniejsze czasy. Z tego powodu przy wyborze metody mutacji powinno się uwzględnić wielkości instancji. Uzyskane wyniki (Rysunek 14) sugerują, że czas wykonywania algorytmu nie zależy od zastosowanego sposobu mutacji.



Rysunek 13: Wykres zależności błędów rozwiązania od wielkości instancji dla dwóch metod mutacji (warunek stopu: czas  $t = 120s$ ).



Rysunek 14: Wykres zależności czasu wykonywania algorytmu od wielkości instancji dla dwóch metod mutacji.. (warunek stopu:  $błąd < \delta$ - Tabela1 ).

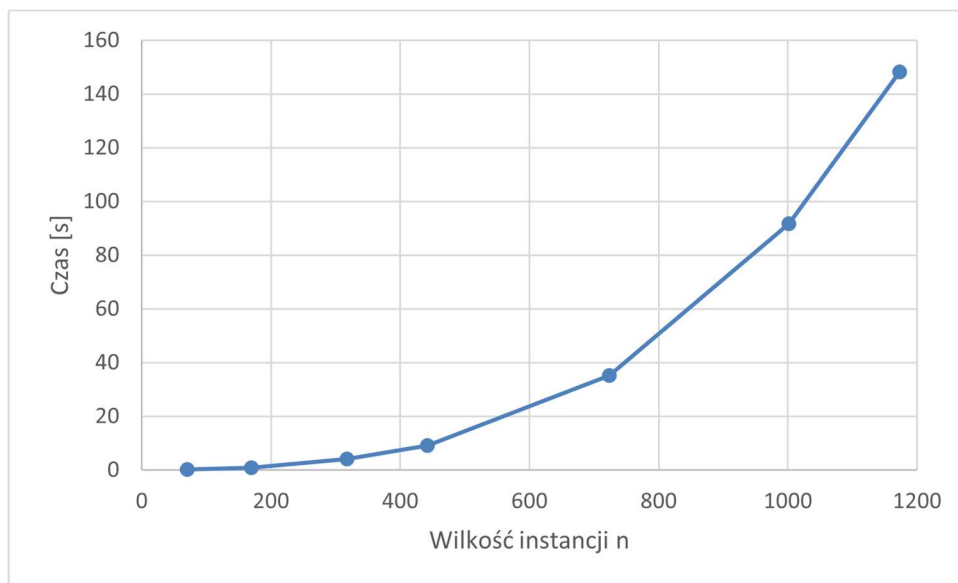


❖ Badania na wybranych instancjach

Ustalone najlepsze parametry (wykorzystane w badaniu):

- Populacja początkowa – algorytm *greedy*
- Prawdopodobieństwo krzyżowania – 0.9
- Prawdopodobieństwo mutacji – 0.01
- Wielkość populacji – 30
- Metoda selekcji – turniejowa
- Metoda mutacji – inwersja
  
- + Metoda krzyżowania – *OX*
- + Metoda sukcesji – ranking

Badanie zostało przeprowadzone dziesięciokrotnie dla każdej instancji, z ograniczeniem do 10 min. Warunkiem stopu było osiągnięcie zadanego progu błędu (*Tabela 1*). Wyniki przedstawione na *Rysunku 15* są wartościami uśrednionymi.



Rysunek 15: Wykres zależności czasu od wielkości instancji, dla wybranych instancji

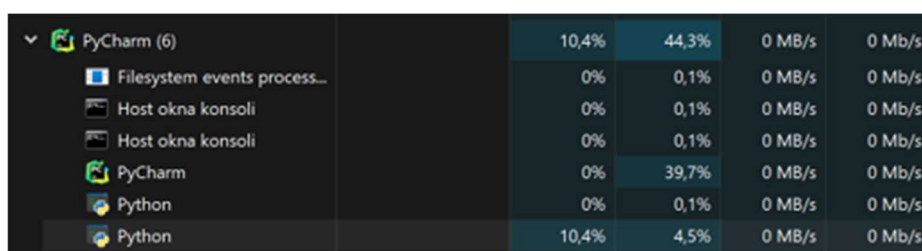
(warunek stopu:  $błqd < \delta$ - Tabela1 ).

Osobno zostało przeprowadzone badanie dla instancji *pr2392.tsp*. Badanie ograniczono do 10 min. Uzyskane wyniki zostały przedstawione na *Rysunku 16*.

	A	B	C	D	E	F
1	Nazwa instancji		Liczba powtórzeń		Wartość optymalna	
2	pr2392.tsp		1		378032	
3						
4	Czas wykonywania [s]:		Uzyskany koszt:		Błąd [%]:	
5	600,0113		5937800		1470,714	

*Rysunek 16: Rezultat badania instancji  $n = 2392$ .*

Zużycie pamięci podczas wykonywania algorytmu zostało pokazane na *Rysunku 17*.

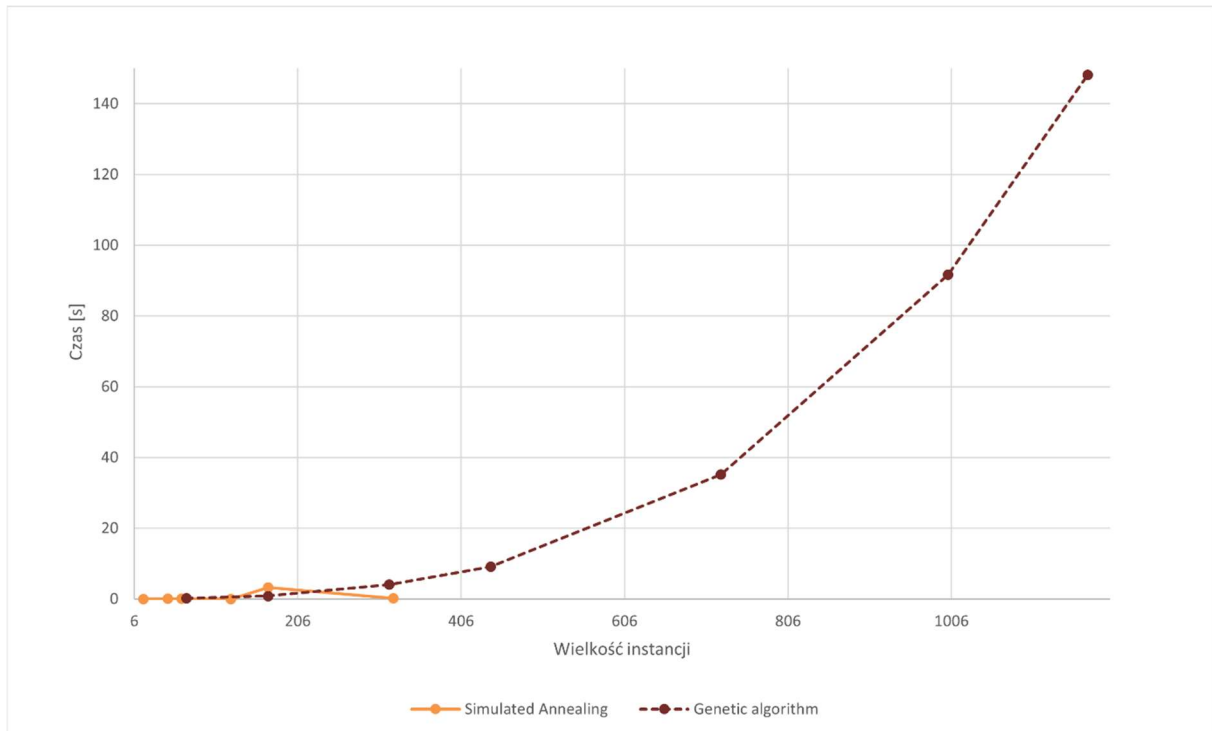


PyCharm (6)	10,4%	44,3%	0 MB/s	0 Mb/s
Filesystem events process...	0%	0,1%	0 MB/s	0 Mb/s
Host okna konsoli	0%	0,1%	0 MB/s	0 Mb/s
Host okna konsoli	0%	0,1%	0 MB/s	0 Mb/s
PyCharm	0%	39,7%	0 MB/s	0 Mb/s
Python	0%	0,1%	0 MB/s	0 Mb/s
Python	10,4%	4,5%	0 MB/s	0 Mb/s

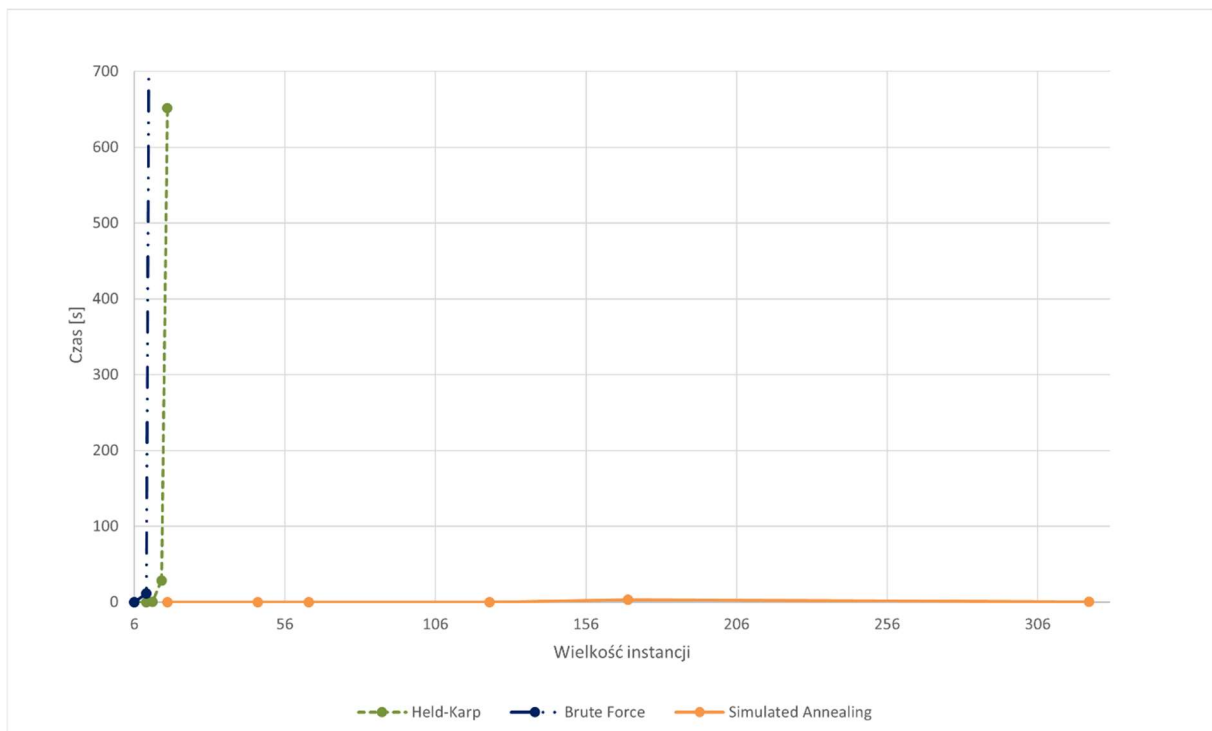
*Rysunek 17: Zużycie pamięci komputera przez algorytm dla instancji  $n = 2392$ .*

Badanie zostało przeprowadzone z zastosowaniem wyboru losowego do rozwiązania początkowego. Z tego powodu uzyskany błąd znacząco przekracza dozwolony próg  $\delta < 150\%$ . Wcześniej przeprowadzone badania z użyciem algorytmu *greedy*, zwracały rozwiązania o błędzie mniejszym od wymaganego progu. Jednak dla większych instancji wygenerowanie początkowego rozwiązania w ten sposób zajmowało znacznie więcej czasu – dochodziło do sytuacji gdy algorytm kończył działanie przed wygenerowaniem rozwiązania początkowego (mijał zadany czas).

## 6. Analiza wyników i wnioski



Rysunek 18: Porównanie czasów wykonywania algorytmów symulowanego wyżarzania (błąd < 40%) oraz algorytmu genetycznego.



Rysunek 19: Porównanie czasów wykonywania wcześniej badanych algorytmów rozwiązujących problem komiwojażera.

Algorytm genetyczny to efektywna technika przeszukiwania rozwiązań, o szerokich możliwościach zastosowania. Jest względnie łatwa w implementacji. Natomiast jej efektywność w dużej mierze zależy od zastosowanych parametrów i metod przeprowadzania kolejnych etapów (selekcji, krzyżowania itd.). Po znalezieniu i zastosowaniu najlepszych parametrów, algorytm znacząco poprawia swoje działanie. Aby uzyskać zadowalające wyniki nie jest potrzebna duża ilość informacji na temat problemu. Pod tym względem algorytm genetyczny ma przewagę nad algorytmem symulowanego wyżarzania. W rozsądnym czasie pozwala na uzyskanie satysfakcjonującego rozwiązania. Jednak należy wziąć pod uwagę, że w rozwiązaniach dopuszczamy błąd, którego wielkość rośnie wraz ze wzrostem instancji. Porównując wybrane algorytmy rozwiązywania problemu komiwojażera (*Rysunek 19 i 20*) można jednoznacznie stwierdzić, że algorytm genetyczny ma przewagę nad pozostałymi pod względem szukania rozwiązania dla dużych instancji.