P||Cmax

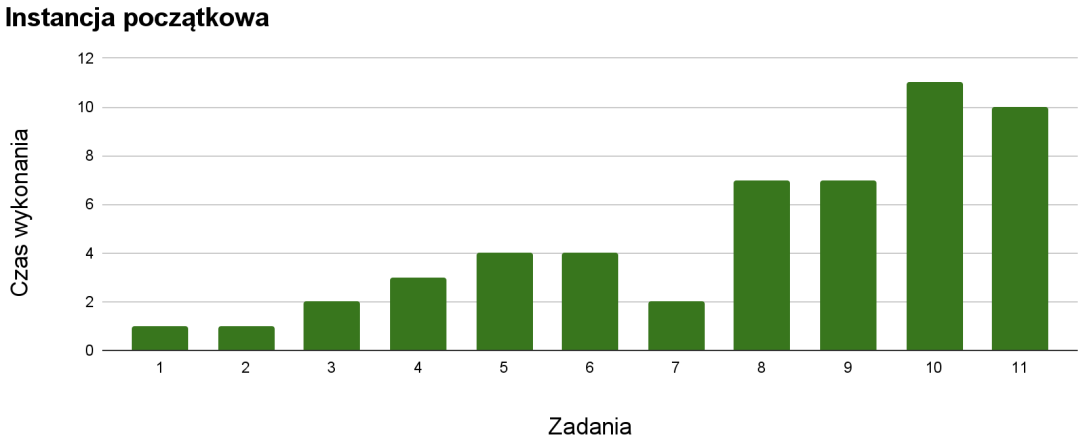
Algorytm genetyczny

1. Inicjalizacja

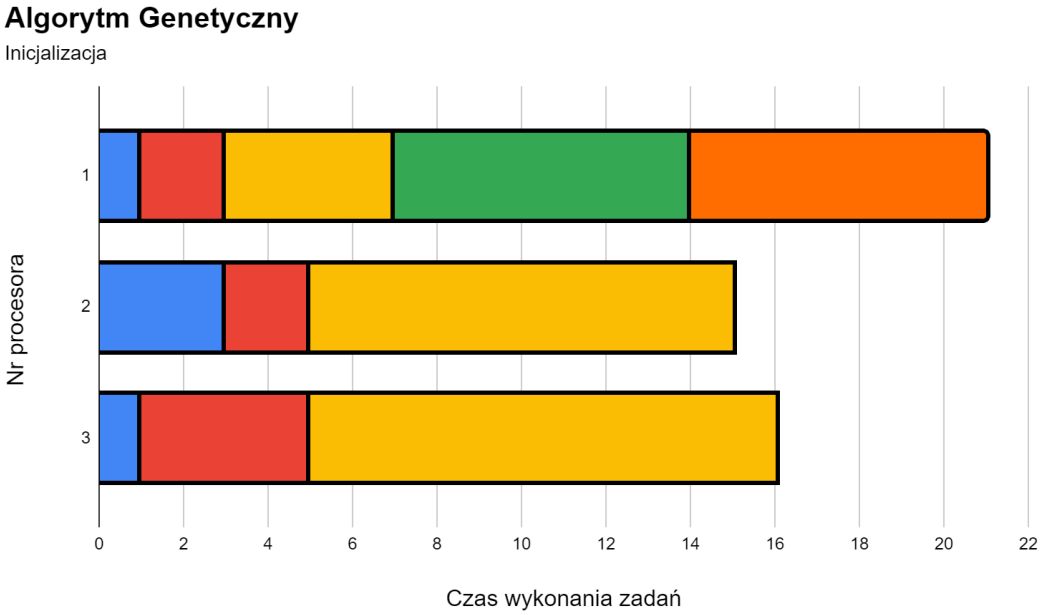
W problemie P||Cmax, zadaniem jest uszeregowanie n zadań na m procesorach w taki sposób, aby czas zakończenia wszystkich zadań na wszystkich procesorach był jak najkrótszy.

Dla naszej instancji początkowej mamy do dyspozycji:

$$\text{Liczba procesorów } (m) \ - \ 3$$
$$\text{Liczba zadań } (n) \ - \ 11$$



Rysunek 1. Wszystkie zadania i odpowiadający im czas wykonania.



Rysunek 2. Graficzne przedstawienie uszeregowania zadań przez algorytm zachłanny.

W dalszych etapach działania algorytmu genetycznego dla problemu P||Cmax, dane są przekształcane na pośredni sposób reprezentowania rozwiązań (ang. *indirect representation of solutions*). W takim podejściu, dane występują w chromosomie w taki sposób, że zadanie wskazane przez pozycje genu jest przypisane do procesora wyznaczonego poprzez odpowiedni allel, jak pokazano na poniższym rysunku.

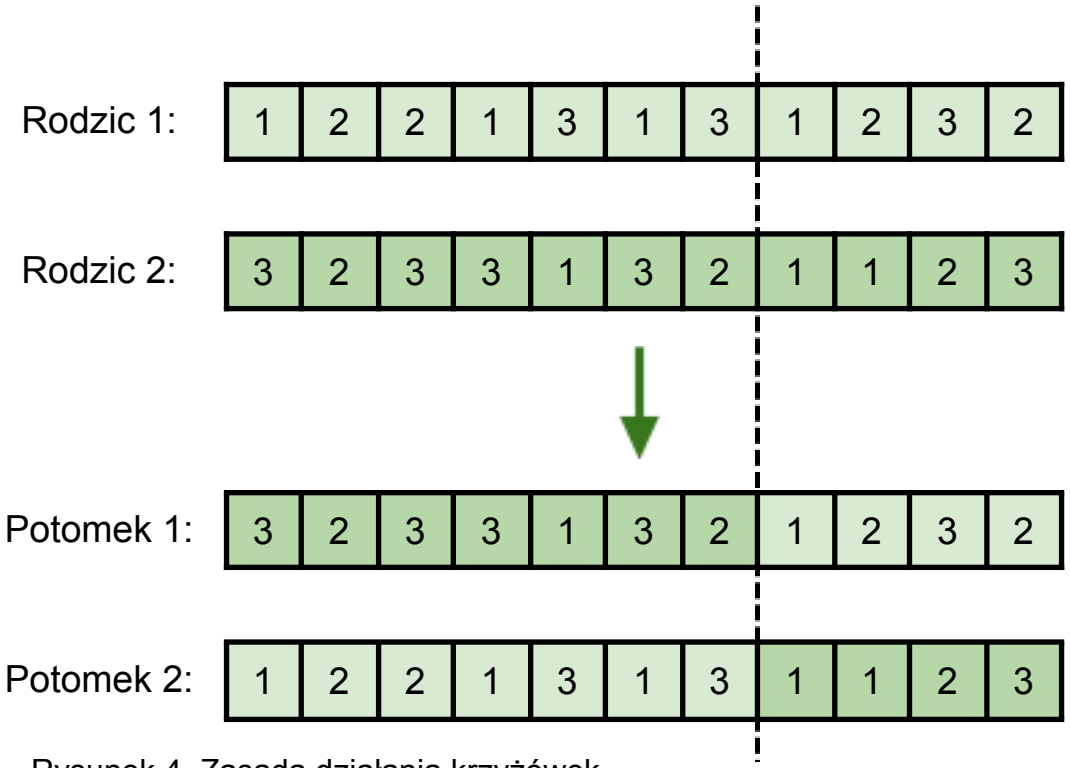
procesory →	3	1	1	2	1	3	2	1	1	3	2
zadania →	1	2	3	4	5	6	7	8	9	10	11

Rysunek 3. Pośredni sposób reprezentowania rozwiązań.

Dzięki takiemu przedstawieniu rozwiązań, możemy stworzyć dekoder, który po otrzymaniu danych zwróci nam rozwiązanie. Chromosom podaje instrukcje jak dekoder ma zbudować przyporządkowanie odpowiednich zadań do konkretnych procesorów.

2. Opis Algorytmu

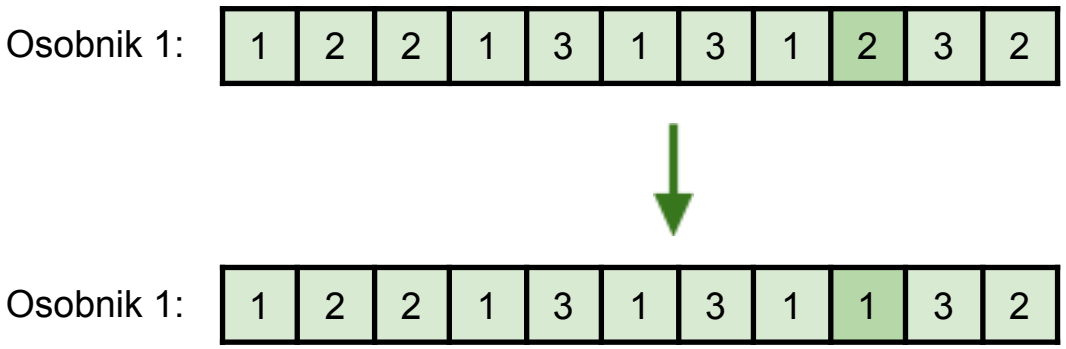
Algorytm genetyczny to rodzaj metaheurystyki, która wykorzystuje procesy genetyczne zachodzące w przyrodzie, do znajdowania najlepszych rozwiązań w zadaniach optymalizacyjnych. Na początku algorytmu losowana jest populacja początkowa, w której genotyp każdego osobnika zapisany jest za pomocą ciągu genów. Za pomocą funkcji przystosowania, wybierane są osobniki z najlepszymi cechami. 90% z nich poddane zostaje krzyżowaniu. Polega ono na wymieszaniu genotypów dwóch osobników (rodziców) i powstaniu dwóch zupełnie nowych jednostek (potomków).



Rysunek 4. Zasada działania krzyżówek.

Aby zachować taką samą liczbę osobników w każdym pokoleniu, 10% organizmów o najlepszych wynikach zostaje przeniesionych do kolejnej generacji, zapewniając tym samym brak możliwości pogorszenia się wyniku.

Dodatkowo, 40% z osobników, które przeszły proces reprodukcji poddawane jest mutacji. Polega ona na losowym zmienieniu jednego genu w genotypie organizmu.



Rysunek 5. Zasada działania mutacji.

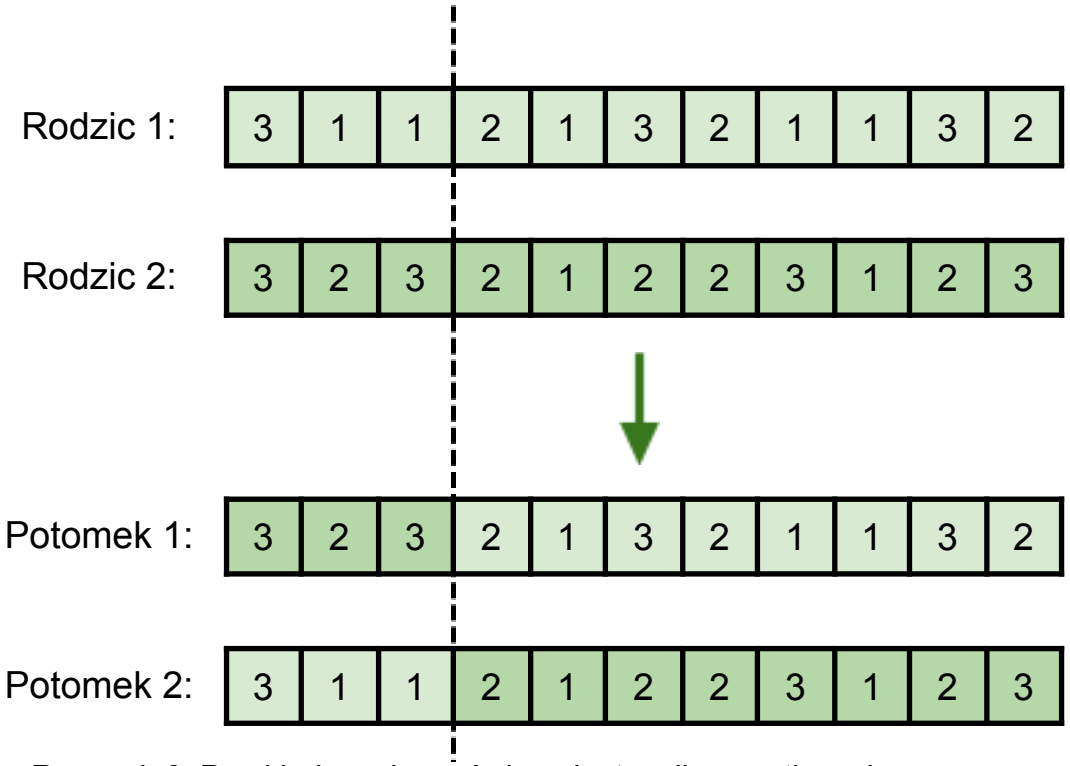
Po zakończonych procesach krzyżowania i mutacji następuje ponowna selekcja w celu ustalenia przydatności organizmów w kolejnych pokoleniach. Algorytm zapętla się w tym momencie powtarzając taką samą listę instrukcji aż do osiągnięcia pożądanego celu lub skończenia się czasu przeznaczonego jego wykonanie.

3. Pseudokod

```
wylosuj_populacje_poczatkowa()
while ilosc_iteracji:
   oblicz_przydatnosc_osobnikow()
   nowe_pokolenie = []
   for 90% najlepszych:
      krzyzuj()
      dopisz_do_nowego_pokolenia()
   for 40% z krzyżowanych osobnikow:
      mutuj()
      dopisz_do_nowego_pokolenia()
   for 10% najlepszych:
      dopisz_do_nowego_pokolenia()
```

4. Przykład obrazujący działanie

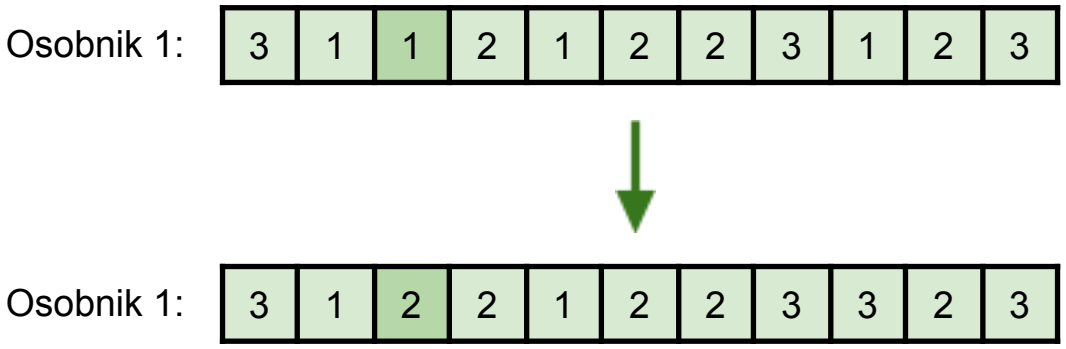
Krzyżówka w początkowej instancji prezentuje się następująco:



Rysunek 6. Przykładowa krzyżówka z instancji początkowej.

W wyniku krzyżówki, z dwóch osobników o wyniku 21 i 21 otrzymano Potomka 1 o wyniku 20 oraz Potomka 2 o wyniku 20. Dzięki prostej wymianie trzech pierwszych genów, udało się zmniejszyć czas wykonania wszystkich zadań.

Mutacja wygląda następująco:

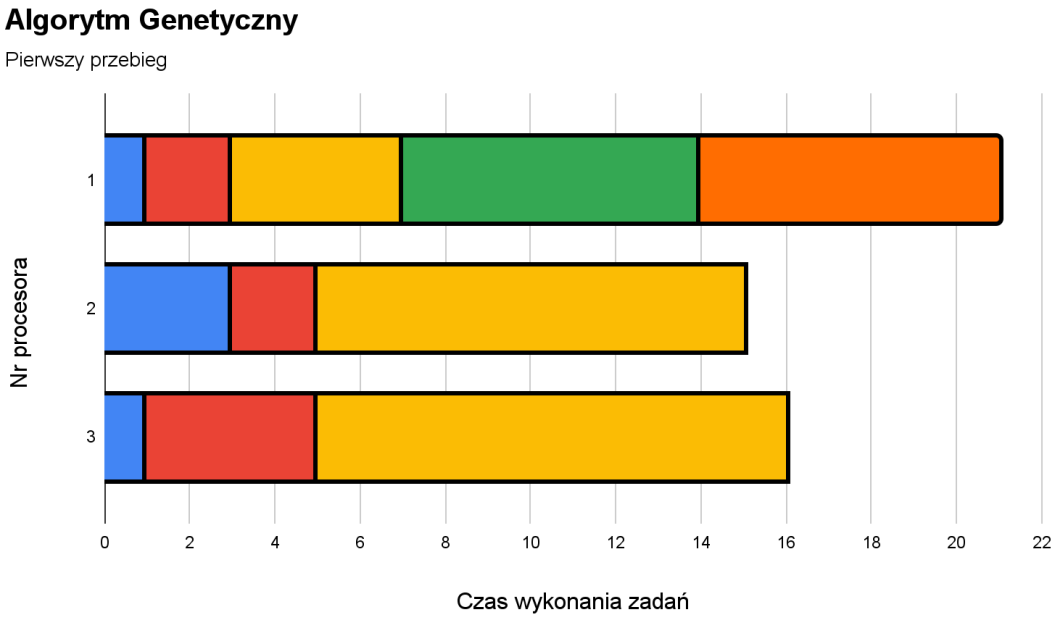


Rysunek 7. Przykładowa mutacja z instancji początkowej.

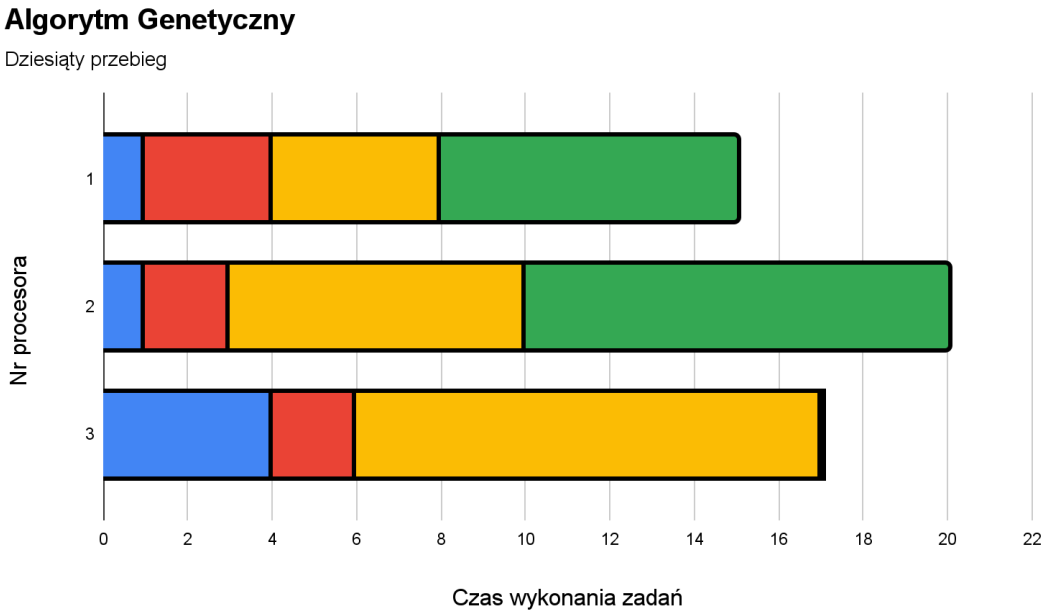
W wyniku mutacji, z genotypu osiągającego wynik 21, otrzymano wynik 19 poprzez zmianę przypisania zadania nr 3 z procesora nr 1 na procesor nr 2.

W wyniku licznych iteracji algorytmu można zaobserwować poprawę wyniku dla całego pokolenia.

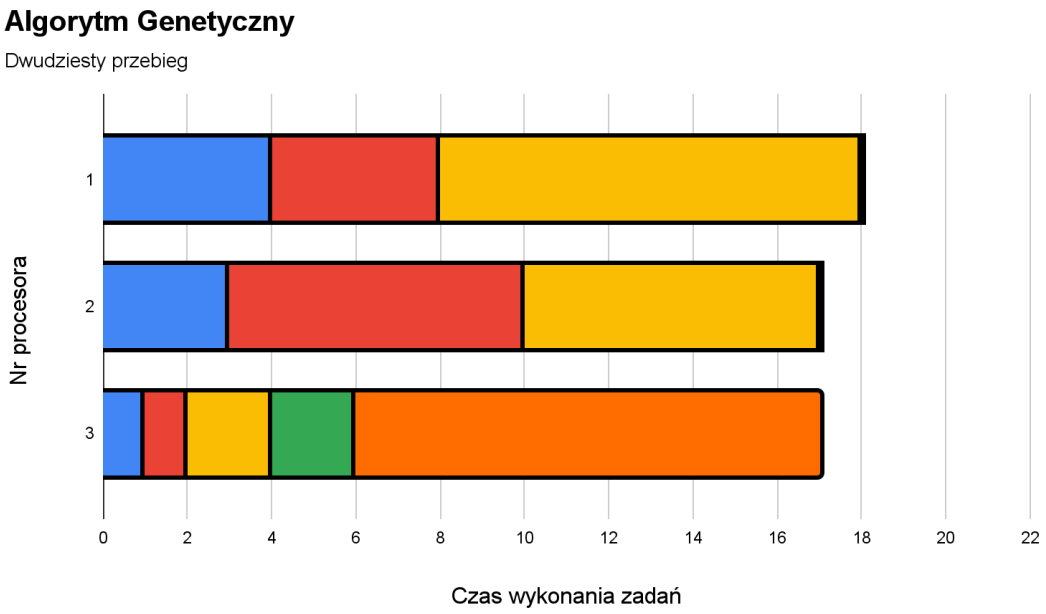
Poniżej przedstawiono rozkład zadań na procesorach dla najlepszego osobnika w odpowiednio pierwszym, dziesiątym i dwudziestym przebiegu rozpatrywanej instancji.



Rysunek 8. Rozkład zadań na procesorach w pierwszym pokoleniu.



Rysunek 9. Rozkład zadań na procesorach w dziesiątym pokoleniu.



Rysunek 10. Rozkład zadań na procesorach w dwudziestym pokoleniu.

5. Finalizacja

Ostateczny wynik algorytmu, został przedstawiony na rysunku poniżej.

procesory →	3	3	3	2	1	1	3	2	2	3	1
zadania →	1	2	3	4	5	6	7	8	9	10	11

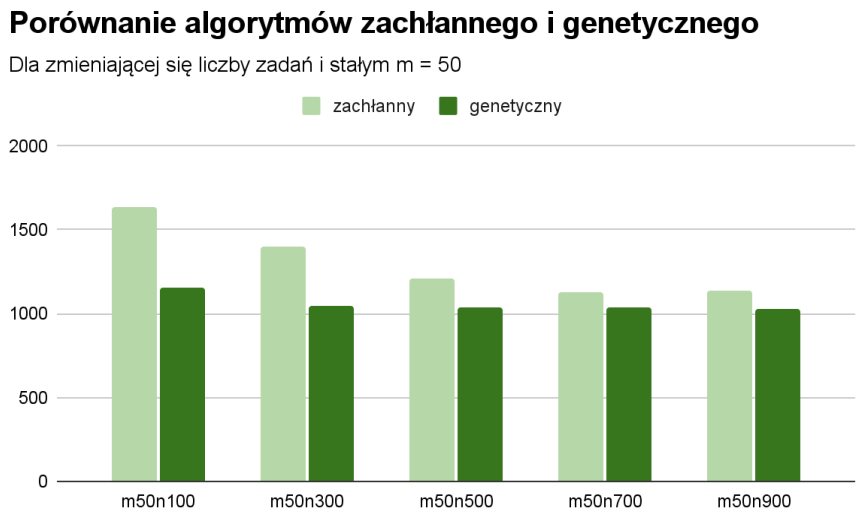
Rysunek 11. Ostateczny wynik algorytmu.

Można z niego odczytać, które zadanie zostało przypisane któremu procesorowi. Wynikiem, nazywa się jednak czas wykonania wszystkich zadań, więc aby go obliczyć, trzeba zsumować wszystkie zadania na poszczególnych procesorach i znaleźć z nich maksimum. W przykładzie powyżej, otrzymamy czas 18 jednostek na procesorze nr 1, czas 17 jednostek na procesorze nr 2 oraz czas 17 jednostek na procesorze nr 3. Oznacza to, że naszym ostatecznym wynikiem jest liczba 18, bo po takim czasie wykonane zostaną wszystkie procesy.

Wykresy

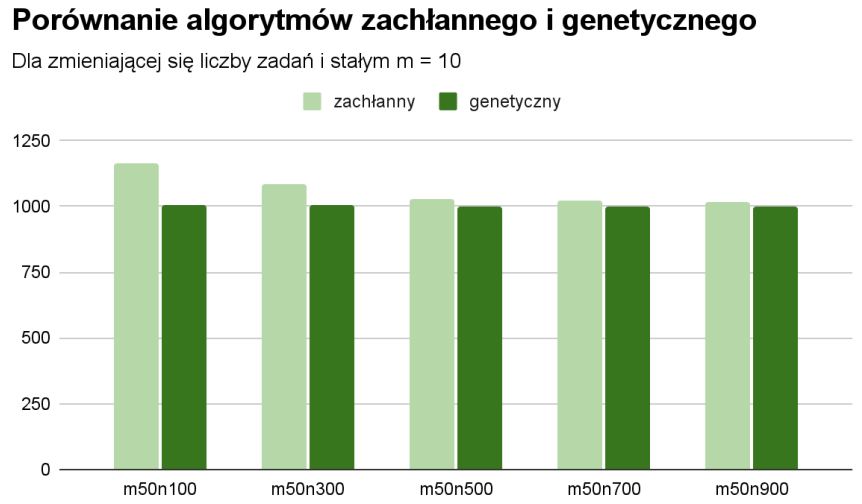
1. Porównanie algorytmu zachłannego i algorytmu genetycznego.

Poniższe wykresy pokazują porównanie pomiędzy wynikami algorytmu zachłannego oraz algorytmu genetycznego dla takich samych instancji wygenerowanych losowo. Pierwszy wykres przedstawia wyniki dla pięciu instancji ze stałą liczbą procesorów równą 50 oraz stałą wartością optymalną równą 1000. Liczba zadań zmienia się co 200 od 100 do 900.



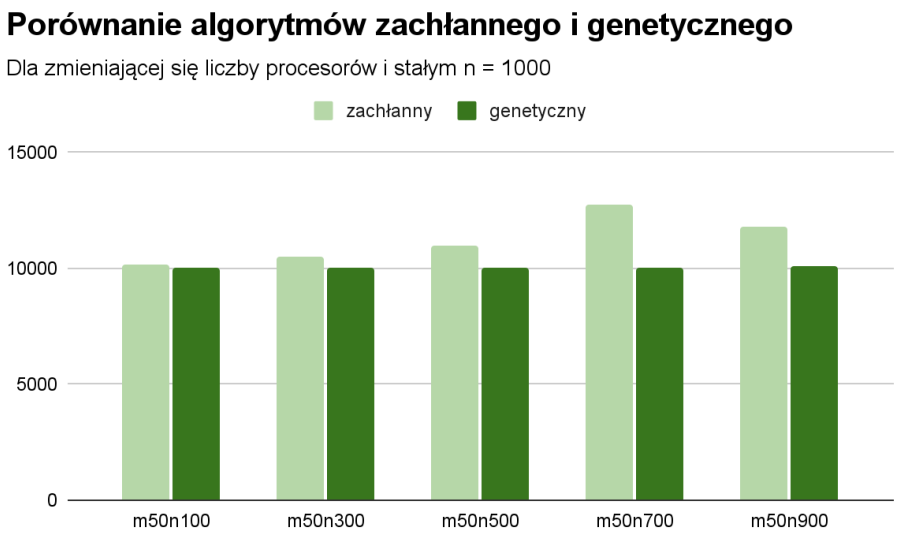
Rysunek 12. Porównanie algorytmów zachłannego i genetycznego.

Na drugim wykresie sytuacja jest bardzo podobna. Liczba zadań zmienia się w zakresie od 100 do 900 co 200, a wartość optymalna jest równa 1000, ale tym razem liczba procesorów wynosi 50.



Rysunek 13. Porównanie algorytmów zachłannego i genetycznego.

Na trzecim wykresie przedstawiono wyniki dla stałej liczby zadań równej 1000 oraz stałym optimum równym 10000. Zmienia się co 10 liczba procesorów, która początkowo wynosi 10.

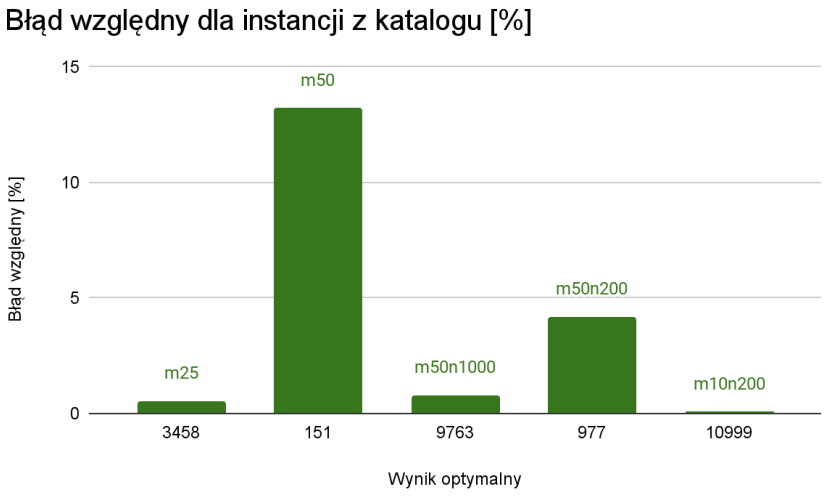


Rysunek 14. Porównanie algorytmów zachłannego i genetycznego.

2. Błąd bezwzględny algorytmu genetycznego w stosunku do wartości optymalnej.

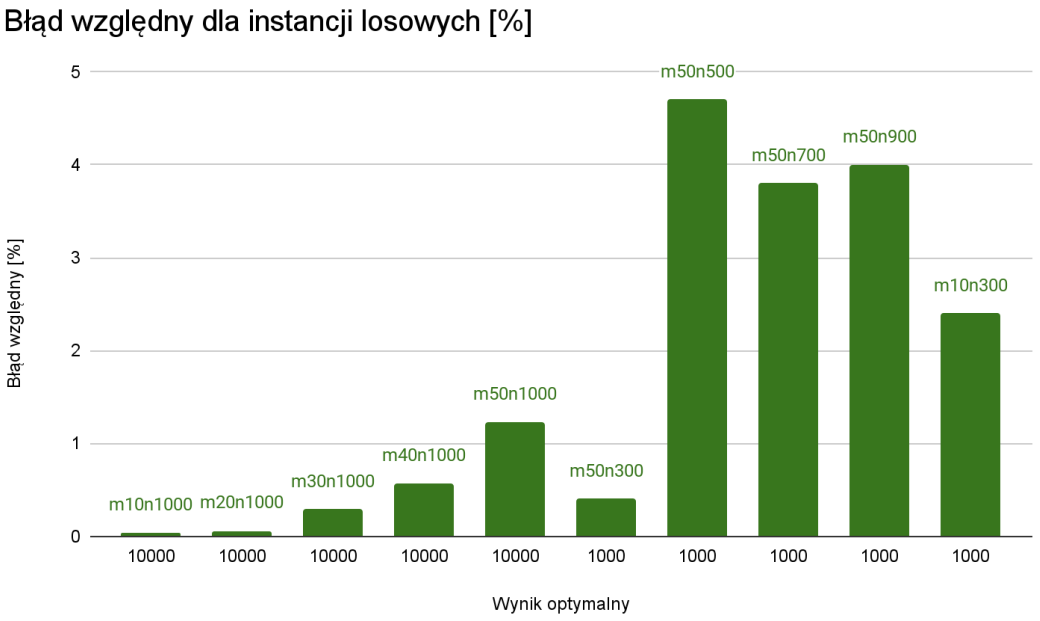
Znając wartości optymalne instancji, można obliczyć błąd względny algorytmu. Aby to zrobić należy podzielić różnicę pomiędzy wynikiem algorytmu a optimum przez optimum.

Poniższy wykres przedstawia błędy względne w instancjach omawianych na laboratoriach.



Rysunek 15. Błąd względny dla instancji z katalogu.

Poniższy wykres przedstawia błędy względne dla wygenerowanych losowo instancji ze znanym optimum.



Rysunek 16. Błąd względny dla instancji wygenerowanych losowo.

3. Wyniki w instancjach z katalogu instancji.

Poniższa tabela przedstawia najlepsze wyniki algorytmu genetycznego dla instancji zadanych na laboratoriach.

Instancja	Wynik
m25	3477
m50	171
m50n1000	9841
m50n200	1018
m10n200	11008