

Problem harmonogramowania zadań w permutacyjnym systemie przepływowym

Przedmiot: Inteligencja Obliczeniowa

Wydział: Wydział Zarządzania

Kierunek: Informatyka i Ekonometria, 3 rok, V semestr

Autorzy:

Marcin Skałka   
Tomasz Solarski  
Mateusz Mulka   
Jakub Laszczyk

[Tabu Search 4](#_Toc1054475972)

[Część teoretyczna: 4](#_Toc2098361697)

[Dobór metody doboru sąsiedztwa 5](#_Toc1769365849)

[Zestaw danych Dane\_PFSP\_100\_10.xlsx 5](#_Toc1225189366)

[Zestaw danych Dane\_PFSP\_200\_10.xlsx 5](#_Toc1419621281)

[Zestaw danych Dane\_PFSP\_50\_20.xlsx 5](#_Toc262013248)

[Dobór liczby iteracji oraz wstępny dobór długości listy tabu 6](#_Toc136374965)

[Zestaw danych Dane\_PFSP\_100\_10.xlsx 6](#_Toc414922776)

[Zestaw danych Dane\_PFSP\_200\_10.xlsx 6](#_Toc1458080630)

[Zestaw danych Dane\_PFSP\_50\_20.xlsx 7](#_Toc1044084423)

[Dokładny dobór długości listy tabu 7](#_Toc1492393918)

[Zestaw danych Dane\_PFSP\_100\_10.xlsx 7](#_Toc648766750)

[Zestaw danych Dane\_PFSP\_200\_10.xlsx 7](#_Toc594198742)

[Zestaw danych Dane\_PFSP\_50\_20.xlsx 8](#_Toc1336152074)

[Wielokrotne powtórzenie algorytmu dla dobranych parametrów 8](#_Toc2029960258)

[Wielokrotne powtórzenie algorytmu dla dobranych parametrów 8](#_Toc1249602935)

[Simulated Annealing 9](#_Toc1729250019)

[Część teoretyczna: 9](#_Toc378982469)

[Dobór metody doboru sąsiedztwa 10](#_Toc1135582365)

[Zestaw danych Dane\_PFSP\_100\_10.xlsx 10](#_Toc364673193)

[Zestaw danych Dane\_PFSP\_200\_10.xlsx 11](#_Toc1853396289)

[Zestaw danych Dane\_PFSP\_50\_20.xlsx 11](#_Toc46613252)

[Dobór metody redukcji temperatury 11](#_Toc1884327100)

[Zestaw danych Dane\_PFSP\_100\_10.xlsx 11](#_Toc1134171024)

[Zestaw danych Dane\_PFSP\_200\_10.xlsx 12](#_Toc443177247)

[Zestaw danych Dane\_PFSP\_50\_20.xlsx 12](#_Toc1502701276)

[Dobór temperatury początkowej 12](#_Toc294505773)

[Zestaw danych Dane\_PFSP\_100\_10.xlsx 12](#_Toc2116850390)

[Zestaw danych Dane\_PFSP\_200\_10.xlsx 13](#_Toc977108474)

[Zestaw danych Dane\_PFSP\_50\_20.xlsx 13](#_Toc1124188947)

[Dobór parametru alfa – szybkości redukcji temperatury 13](#_Toc1214568356)

[Zestaw danych Dane\_PFSP\_100\_10.xlsx 14](#_Toc854491194)

[Zestaw danych Dane\_PFSP\_200\_10.xlsx 14](#_Toc2076164599)

[Zestaw danych Dane\_PFSP\_50\_20.xlsx 14](#_Toc754817487)

[Dobór liczby iteracji 15](#_Toc758224426)

[Zestaw danych Dane\_PFSP\_100\_10.xlsx 15](#_Toc1887993604)

[Zestaw danych Dane\_PFSP\_200\_10.xlsx 15](#_Toc1553475640)

[Zestaw danych Dane\_PFSP\_50\_20.xlsx 16](#_Toc258948236)

[Analiza wpływu jakości rozwiązania początkowego na wielkość uzyskanego rozwiązania 16](#_Toc1349468462)

[Zestaw danych Dane\_PFSP\_100\_10.xlsx 16](#_Toc1823753452)

[Zestaw danych Dane\_PFSP\_200\_10.xlsx 17](#_Toc310341763)

[Zestaw danych Dane\_PFSP\_50\_20.xlsx 17](#_Toc814584785)

[Wielokrotne powtórzenie algorytmu dla dobranych parametrów 17](#_Toc1136590798)

[Algorytm Genetyczny 18](#_Toc299975754)

[Wstęp teoretyczny 18](#_Toc1529074066)

[Działanie algorytmu 18](#_Toc564154695)

[Analiza doboru sąsiedztwa oraz wpływ początkowej wartości funkcji celu na finalny wynik 19](#_Toc2072968929)

[Analiza wpływu ilości generacji 22](#_Toc1718047310)

[Analiza wpływu metody doboru rodziców 24](#_Toc1846398859)

[Analiza wpływu szansy na mutację 24](#_Toc781727514)

[Analiza wpływu rodzaju funkcji crossover – rodzaju krzyżówki 26](#_Toc791141779)

[Analiza wpływu maksymalnej szerokości przedziałów bocznych w metodzie PMX 26](#_Toc1169898801)

[Przeprowadzenie finalnych obliczeń 29](#_Toc1655080715)

[Podsumowanie 30](#_Toc1419929458)

[Algorytm wspinaczki (HILL CLIMBING) 32](#_Toc1017247932)

[Wprowadzenie 32](#_Toc205405771)

[Zasada działania: 32](#_Toc24589381)

[Optymalizacja parametrów 32](#_Toc1134963859)

[Opis Parametrów: 32](#_Toc684836200)

[Zestaw Danych PFSP\_50\_20.xlsx 33](#_Toc593325075)

[Zestaw Danych PFSP\_100\_10.xlsx 36](#_Toc195756529)

[Zestaw Danych PFSP\_200\_10.xlsx 41](#_Toc1316726190)

[Poszukiwanie najlepszego wyniku po optymalizacji parametrów 43](#_Toc2119861856)

[Najlepszy wynik osiągnięty dla 50\_20: 43](#_Toc133232442)

[Najlepszy wynik osiągnięty dla 100\_10: 43](#_Toc341070)

[Najlepszy wynik osiągnięty dla 200\_10: 43](#_Toc73280294)

[Algorytm NEH 44](#_Toc1602775846)

[Wstęp teoretyczny: 44](#_Toc272924446)

[Działanie algorytmu: 44](#_Toc1298717323)

[Wyniki otrzymane dla algorytmu NEH: 44](#_Toc619467865)

[EXCEL SOLVER 45](#_Toc1228183782)

[Zestaw Danych PFSP\_50\_20.xlsx 45](#_Toc1799098272)

[Zestaw Danych PFSP\_100\_10.xlsx 45](#_Toc1491026329)

[Zestaw Danych PFSP\_200\_10.xlsx 45](#_Toc1424436721)

# Tabu Search

## Część teoretyczna:

Przeszukiwanie z tabu jest metahuerystyką do rozwiązywania problemów optymalizacyjnych, opartą na iteracyjnym przeszukiwaniu przestrzeni rozwiązań, wykorzystując sąsiedztwo pewnych elementów tej przestrzeni oraz zapamiętując przy tym przeszukiwaniu ostatnie ruchy, dopóki nie zostanie spełniony warunek końcowy. W roku 1986 Fred Glover wydał pierwszą pracę na temat Tabu Search.

Ogólna idea:

Wykonane ruchy np. zamiana danego miasta z innym (metoda swap) są zapisywane na **liście tabu**. Obecność danego ruchu na liście tabu jest tymczasowa – jeżeli wykonamy dany ruch to ten ruch zapisywany jest na liście tabu i będzie tam przez określoną liczbę iteracji k jaką ustali użytkownik. Jeżeli dany ruch jest na liście tabu, oznacza to, że tego ruchu nie możemy wykonać przez kolejnych k iteracji – chyba, że ruch spełnia **kryterium aspiracji**. Celem listy tabu jest wykluczenie prawdopodobieńst zapętleń przy przeszukiwaniu i zmuszenie algorytmu do przeszukiwania nowych regionów przestrzeni przeszukiwań. Szybkość algorytmu nie jest rewelacyjna, ponieważ zamieniamy dane rozwiązanie z najlepszym jego sąsiadem, bez względu na to czy sąsiad jest od niego lepszy czy gorszy. Pozytywnym aspektem tego podejścia jest jednak możliwość szybszego wyjścia z regionu przyciągania ekstremum lokalnego, które niekoniecznie musi być globalnie optymalne (w szczególności, może być czasami bardzo odległe od optymalnego).   
Zazwyczaj **warunkiem końcowym przeszukiwania** jest:  
- zadana liczba iteracji  
- brak poprawy rozwiązania przez określoną liczbę iteracji z rzędu  
- osiągnięcie określonej wartość funkcji celu.

Najbardziej czasochłonnym ruchem tego algorytmu jest przegląd całego sąsiedztwa w każdej z iteracji.  
W naszym przypadku wykorzystujemy dwie struktury sąsiedztwa:  
- swap(i,j) – zamiana miejscami i-tego elementu z j-tym  
- invert(i,j) – odwrócenie kolejności w podciągu zaczynającym się na i-tej pozycji i kończącym się na j-tej pozycji

Kryterium aspiracji:  
Spełnienie kryterium aspiracji pozwala na złamanie tabu, czyli wykonanie ruchu, pomimo że jego atrybuty znajdują się na liście ruchów zabronionych. Najprostszym i najczęściej stosowanym kryterium aspiracji jest zezwolenie na wykonanie ruchu, jeżeli prowadzi on bezpośrednio do uzyskania najlepszego znanego globalnie rozwiązania.

Wykorzystanie pamięci krótkotrwałej i długoterminowej:  
- krótkotrwałej – do zapamiętywania ostatnich wykonanych ruchów i nadpisywania zmian w kolejnych iteracjach algorytmu  
- długotrwałej – do zapamiętywania najbardziej atrakcyjnych elementów przestrzeni przeszukiwań

Algorytm w postaci listy kroków:

1. Wygeneruj losowe rozwiązanie, ustal długość listy tabu
2. Przez określoną liczbę iteracji powtarzaj:
   1. Wygeneruj wszystkich sąsiadów dla danego rozwiązania
   2. KRYTERIUM ASPIRACJI jeżeli ten ruch jest na liście tabu, ale jego wartość jest najlepsza globalnie to wybierz tego sąsiada
   3. Wybierz najlepszego sąsiada jeżeli ten ruch nie jest na liście tabu. Jeżeli jest to wybierz kolejnego najlepszego sąsiada, którego nie ma na liście tabu
   4. Jeżeli wartość funkcji celu dla wybranego sąsiada jest najlepsza dotychczas to uaktualnij najlepsze rozwiązanie i wartość jego funkcji celu.
   5. Zaktualizuj listę tabu
3. Zwróć najlepsze rozwiązanie wraz z jego wartością funkcji celu

## Dobór metody doboru sąsiedztwa

Na początku sprawdziliśmy dla której metody doboru sąsiedztwa, algorytm zwraca lepsze wyniki, dla poszczególnych zestawów danych. Dla każdego rodzaju danych wywołaliśmy algorytm 6 razy dla danego kryterium stopu, którym u nas była liczba iteracji równa 120, długość listy tabu ustawiliśmy na 10. Zrobiliśmy to w celu wyznaczenia średniej wartości funkcji celu dla każdego z typu sąsiedztwa. Wyniki zostały zaprezentowane poniżej dla poszczególnych zestawów danych. Dokonując tych pomiarów, zbieraliśmy także informacje o najlepszym wyniku funkcji celu w całym algorytmie.

### Zestaw danych Dane\_PFSP\_100\_10.xlsx

|  |  |  |
| --- | --- | --- |
| rodzaj sąsiedztwa | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| swap | 5584,67 | 5569 |
| invert | 5792,67 | 5714 |

### Zestaw danych Dane\_PFSP\_200\_10.xlsx

|  |  |  |
| --- | --- | --- |
| rodzaj sąsiedztwa | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| swap | 10654,67 | 10625 |
| invert | 10902,67 | 10810 |

### Zestaw danych Dane\_PFSP\_50\_20.xlsx

|  |  |  |
| --- | --- | --- |
| rodzaj sąsiedztwa | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| swap | 3897,67 | 3873 |
| invert | 4018,00 | 3983 |

Podsumowując, dla każdego zestawu danych, lepszą metodą doboru sąsiedztwa jest metoda swap. Zwróciła ona znacznie lepsze wyniki. Zdecydowaliśmy, że dalsze obliczenia przeprowadzimy wykorzystując tylko metodę swap do tworzenia sąsiedztwa.

Podsumowując analizę dotychczasowych parametrów, dla poszczególnych danych wiemy już, że:

* Dane\_PFSP\_100\_10.xlsx: sąsiedztwo: swap
* Dane\_PFSP\_200\_10.xlsx: sąsiedztwo: swap
* Dane\_PFSP\_50\_20.xlsx: sąsiedztwo: swap

Dotychczasowe najlepsze rozwiązania:

* Dane\_PFSP\_100\_10.xlsx: wartość funkcji celu: 5569
* Dane\_PFSP\_200\_10.xlsx: wartość funkcji celu: 10625
* Dane\_PFSP\_50\_20.xlsx: wartość funkcji celu: 3873

## Dobór liczby iteracji oraz wstępny dobór długości listy tabu

Teraz przetestujemy jaką liczbę iteracji wybrać oraz wstępnie dobierzemy długość listy tabu, aby w późniejszych wywołaniach algorytmu dobrać jeszcze bardzie optymalną długość listy tabu. W tym celu wywołamy algorytm 4 razy dla każdej kombinacji tych parametrów i obliczymy średnią wartość funkcji celu oraz najlepszą uzyskaną.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| liczba iteracji | 10 | 20 | 30 | 50 | 75 | 100 |
| długość listy tabu | 4 | 10 | 16 |  |  |  |

Wyniki zaprezentujemy w posortowanej tabeli względem średniej wartości funkcji celu, uwzględnimy tylko 5 najlepszych pozycji.

### Zestaw danych Dane\_PFSP\_100\_10.xlsx

|  |  |  |  |
| --- | --- | --- | --- |
| liczba iteracji | długość listy tabu | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| 100 | 16 | 5590 | 5588 |
| 75 | 10 | 5594,5 | 5589 |
| 75 | 16 | 5598,5 | 5580 |
| 100 | 10 | 5621 | 5569 |
| 50 | 4 | 5626,5 | 5607 |

W tym przypadku, wybieramy piątą kombinację z tabeli. Wartości pomiędzy kombinacjami w pierwszej piątce nie różnią się znacząco, lecz czas ich wykonywania różni się bardzo, dzięki wyborze 50 iteracji zaoszczędzimy sporo czasu.

### Zestaw danych Dane\_PFSP\_200\_10.xlsx

W przypadku danych dla 200 zadań algorytm liczył się bardzo, bardzo długo w porównaniu do innych algorytmów, a zwracał wyniki nie takie jakich byśmy się spodziewali po tak ogromnie długim czasie działania algorytmu. Zdecydowaliśmy się przerwać działanie algorytmu dla wywołań dla 75 i 100 iteracji. Uznaliśmy, że czas poświęcony na czekanie aż algorytm zakończy działanie jest bezsensowne i możemy użyć innych algorytmów, które w o wiele szybszym tempie zwrócą nam satysfakcjonujący nas wynik np. symulowane wyżarzanie. Nie zakończymy analizy tego zbioru danych na tym co napisaliśmy powyżej, ale wybierzemy mniejszą liczbę iteracji dla tego przypadku danych i sprawdzimy, czy porównywalnie dobre wyniki z innymi algorytmami i w podobnym czasie zwróci nam Tabu Search.

|  |  |  |  |
| --- | --- | --- | --- |
| liczba iteracji | długość listy tabu | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| 50 | 4 | 10637.0 | 10633 |
| 50 | 10 | 10672.5 | 10668 |
| 30 | 4 | 10808.5 | 10796 |
| 30 | 16 | 10809.5 | 10789 |
| 30 | 10 | 10849.0 | 10815 |
| 20 | 10 | 10883.0 | 10873 |

Zdecydowaliśmy się wybrać 20 iteracji oraz długość listy tabu równą 10.

### Zestaw danych Dane\_PFSP\_50\_20.xlsx

|  |  |  |  |
| --- | --- | --- | --- |
| liczba iteracji | długość listy tabu | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| 50 | 16 | 3832,5 | 3835 |
| 100 | 10 | 3847,5 | 3836 |
| 75 | 16 | 3860,5 | 3848 |
| 30 | 10 | 3875 | 3844 |
| 75 | 4 | 3883 | 3855 |

Dla tego zbioru danych do następnych obliczeń wybieramy pierwszą kombinację z tabeli. Dla tej kombinacji średnia wartość f. celu oraz najlepsza uzyskana wartość f. celu były najlepsze.

Podsumowując analizę dotychczasowych parametrów, dla poszczególnych danych wiemy już, że:

* Dane\_PFSP\_100\_10.xlsx: sąsiedztwo: swap, liczba iteracji: 50
* Dane\_PFSP\_200\_10.xlsx: sąsiedztwo: swap, liczba iteracji: 20
* Dane\_PFSP\_50\_20.xlsx: sąsiedztwo: swap, liczba iteracji: 50

Dotychczasowe najlepsze rozwiązania:

* Dane\_PFSP\_100\_10.xlsx: wartość funkcji celu: 5569
* Dane\_PFSP\_200\_10.xlsx: wartość funkcji celu: 10625
* Dane\_PFSP\_50\_20.xlsx: wartość funkcji celu: 3835

## Dokładny dobór długości listy tabu

Teraz włączymy algorytm, kilka razy, dla danej liczby iteracji oraz dla trzech wartości parametrów oscylujących wokół dobranej w poprzednim punkcie długości listy tabu.

### Zestaw danych Dane\_PFSP\_100\_10.xlsx

Dla tego zestawu danych długość listy tabu jaką zbadamy to: 2, 5, 8.

|  |  |  |
| --- | --- | --- |
| długość listy tabu | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| 2 | 5632,67 | 5603 |
| 5 | 5620,00 | 5591 |
| 8 | 5635,00 | 5612 |

### Zestaw danych Dane\_PFSP\_200\_10.xlsx

Dla tego zestawu danych długość listy tabu jaką zbadamy to: 6, 10, 14

|  |  |  |
| --- | --- | --- |
| długość listy tabu | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| 6 | 11018,33 | 10991 |
| 10 | 10918,00 | 10877 |
| 14 | 10946,67 | 10903 |

### Zestaw danych Dane\_PFSP\_50\_20.xlsx

Dla tego zestawu danych długość listy tabu jaką zbadamy to: 12, 16 oraz 20.

|  |  |  |
| --- | --- | --- |
| długość listy tabu | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| 12 | 3895,67 | 3884 |
| 16 | 3893,67 | 3865 |
| 20 | 3866,00 | 3833 |

Podsumowując analizę dotychczasowych parametrów, dla poszczególnych danych wiemy już, że:

* Dane\_PFSP\_100\_10.xlsx: sąsiedztwo: swap, liczba iteracji: 50, długość listy tabu: 5
* Dane\_PFSP\_200\_10.xlsx: sąsiedztwo: swap, liczba iteracji: 20, długość listy tabu: 10
* Dane\_PFSP\_50\_20.xlsx: sąsiedztwo: swap, liczba iteracji: 50, długość listy tabu: 20

Dotychczasowe najlepsze rozwiązania:

* Dane\_PFSP\_100\_10.xlsx: wartość funkcji celu: 5569
* Dane\_PFSP\_200\_10.xlsx: wartość funkcji celu: 10625
* Dane\_PFSP\_50\_20.xlsx: wartość funkcji celu: 3833

## Wielokrotne powtórzenie algorytmu dla dobranych parametrów

Teraz włączymy wielokrotnie algorytmy w celu uzyskania jak najniższej wartości funkcji celu. Dodaliśmy także warunek, dla liczby iteracji bez poprawy, jeżeli liczba iteracji bez poprawy będzie większa niż 11 dla danych z liczbą iteracji równą 50 to przerywamy algorytm. Tak samo dla danych z 20 iteracjami, lecz przerywamy algorytm po 7 iteracjach bez poprawy. Podsumowanie wyników znajduje się poniżej:

Podsumowując analizę dotychczasowych parametrów, dla poszczególnych danych wiemy już, że:

* Dane\_PFSP\_100\_10.xlsx: sąsiedztwo: swap, liczba iteracji: 50, długość listy tabu: 5
* Dane\_PFSP\_200\_10.xlsx: sąsiedztwo: swap, liczba iteracji: 20, długość listy tabu: 10
* Dane\_PFSP\_50\_20.xlsx: sąsiedztwo: swap, liczba iteracji: 50, długość listy tabu: 20

Dotychczasowe najlepsze rozwiązania:

* Dane\_PFSP\_100\_10.xlsx: wartość funkcji celu: 5569
* Dane\_PFSP\_200\_10.xlsx: wartość funkcji celu: 10625
* Dane\_PFSP\_50\_20.xlsx: wartość funkcji celu: 3804

## Wielokrotne powtórzenie algorytmu dla dobranych parametrów

Teraz włączymy wielokrotnie algorytmy w celu uzyskania jak najniższej wartości funkcji celu. Ze względu na czas wykonywania algorytmu dodaliśmy także warunek liczby iteracji bez poprawy. Limit iteracji bez poprawy był u nas równy liczba iteracji/2, po tylu iteracjach bez poprawy algorytm przerywał swoje działanie. Podsumowanie wyników znajduje się poniżej:

Podsumowując analizę dotychczasowych parametrów, dla poszczególnych danych wiemy już, że:

* Dane\_PFSP\_100\_10.xlsx: sąsiedztwo: swap, liczba iteracji: 50, długość listy tabu: 5
* Dane\_PFSP\_200\_10.xlsx: sąsiedztwo: swap, liczba iteracji: 20, długość listy tabu: 10
* Dane\_PFSP\_50\_20.xlsx: sąsiedztwo: swap, liczba iteracji: 50, długość listy tabu: 20

Takie wyniki mieliśmy na początku(takie wysokie wyniki początkowe są spowodowane tym, że włączyliśmy algorytm na bardzo dużą liczbę iteracji przy doborze metody sąsiedztwa):

* Dane\_PFSP\_100\_10.xlsx: wartość funkcji celu: 5569
* Dane\_PFSP\_200\_10.xlsx: wartość funkcji celu: 10625
* Dane\_PFSP\_50\_20.xlsx: wartość funkcji celu: 3873

Niektóre z nich udało nam się zmniejszyć poprzez wybór odpowiednich parametrów:

* Dane\_PFSP\_100\_10.xlsx: wartość funkcji celu: 5569
* Dane\_PFSP\_200\_10.xlsx: wartość funkcji celu: 10625
* Dane\_PFSP\_50\_20.xlsx: wartość funkcji celu: 3804

W algorytmach optymalizacji bardzo ważny jest czas, więc dobierając parametry powinniśmy zwrócić uwagę na wynik algorytmu oraz jego czas. Jeżeli chcemy uzyskać dobre wyniki algorytmem Tabu Search to musimy wykonywać go długo, przez dużą liczbę iteracji. Dla danych z 200 zadaniami zdecydowaliśmy się wybrać małą liczbę iteracji, ponieważ z większą liczbą iteracji algorytm liczył się bardzo długo. W tym wypadku algorytm zwracał w miarę dobre rozwiązania, lecz na pewno dałby lepsze przy większej liczbie iteracji, ale raczej nie jest to opłacalne względem długości liczenia. Długość listy tabu nie powinna być zbyt mała oraz za duża względem liczby iteracji. Zbyt mała, ponieważ wtedy algorytm jest bardziej intensyfikacyjny, tzn. że może krążyć cały czas wokół jednego miejsca oraz za duża, ponieważ wtedy algorytm będzie bardziej w stylu dywersyfikacji tzn. będzie go to odpychało od miejsc gdzie już był, będzie nastawiony na eksplorację nowych regionów. W naszym przypadku o wiele bardziej sprawdziło się sąsiedztwo tworzone metodą swap.

# Simulated Annealing

## Część teoretyczna:

Symulowane wyżarzanie jest metaheurystyką do rozwiązywania problemów optymalizacyjnych. Algorytm unika wad algorytmów lokalnego przeszukiwania poprzez ograniczone akceptowanie pogorszeń funkcji celu. Cechą charakterystyczną tej metody jest występowanie parametru sterującego zwanego temperaturą, który maleje w trakcie wykonywania algorytmu. Im wyższą wartość ma ten parametr, tym bardziej chaotyczne mogą być zmiany. Podejście to jest inspirowane zjawiskami obserwowanymi w metalurgii – im większa temperatura metalu, tym bardziej jest on plastyczny. (wyżarzanie – powolne zmniejszanie temperatury do chwili, w której cząsteczki ułożą się wzajemnie i osiągną temperaturę zerową). Algorytm pochodzi od algorytmu Monopolis. Algorytm nie przeszukuje całego sąsiedztwa dla danego rozwiązania. Kolejne rozwiązania przyjmuje na podstawie kryterium akceptacji.

**Kryterium akceptacji:**  
Rozwiązania: i – aktualne oraz j – proponowane  
Koszty (wartość funkcji celu): f(i), f(j)  
Temperatura: T  
Kryterium akceptacji określa czy proponowane rozwiązanie jest akceptowane:  
- Jeżeli f(i) – f(j) jest większe lub równe 0 to przyjmujemy proponowane rozwiązanie z prawdopodobieństwem równym 1.  
- Jeżeli f(i) – f(j) jest mniejsze od 0 to przyjmujemy proponowane rozwiązanie z prawdopodobieństwem równym *e(f(i))−f(j))/T*

**Sposób chłodzenia** określa:  
- początkowa wartości temperatury – powinna być odpowiednio wysoka by zapewnić akceptację wszystkich przejść, im większa temperatura tym bardziej chaotyczne są wybory  
- funkcja redukcji temperatury – stosujemy redukcję geometryczną, ustalamy parametr alfa bliski 1, ponieważ wtedy będzie chętnie wchodził w polepszenia i pogorszenia, a wraz ze zmniejszaniem temperatury algorytm będzie wybierał częściej te lepsze rozwiązania, funkcja redukcji temperatury ma więc postać: alfa\*T, można także zastosować powolny spadek temperatury, przedstawiony wzorem: T/(1 + alfa \* T)  
- końcowa wartość temperatury – algorytm może działać, aż do określonej końcowej temperatury lub przez określoną liczbę iteracji lub przez określoną liczbę iteracji bez poprawy

Tak samo jak w algorytmie Tabu Search trzeba rejestrować najlepsze dotychczasowe rozwiązanie w jakim się było, ponieważ algorytm przyjmuje także rozwiązania pogorszające.

Algorytm w postaci listy kroków:

1. Wygeneruj losowe rozwiązanie, ustal początkową wartość temperatury
2. Przez określoną liczbę iteracji powtarzaj:
   1. Wybierz losowe rozwiązanie j znajdujące się w pobliżu dotychczasowego rozwiązania i.
   2. Jeżeli nowe rozwiązanie jest lepsze to je przyjmij, jeżeli jest gorsze wyznacz prawdopodobieństwo akceptacji *e(f(i))−f(j))/T* . Wylosuj liczbę z przedziału [0,1], jeżeli jest ona mniejsza od prawdopodobieństwa akceptacji to przyjmij to rozwiązanie, w przeciwnym wypadku pozostań przy aktualnym rozwiązaniu.
   3. Jeżeli wartość funkcji celu dla wybranego rozwiązania jest najlepsza dotychczas to uaktualnij najlepsze rozwiązanie i wartość jego funkcji celu.
   4. Zmniejsz temperaturę.
3. Zwróć najlepsze rozwiązanie wraz z jego wartością funkcji celu.

## Dobór metody doboru sąsiedztwa

Dla algorytmu SA, także na początku sprawdziliśmy dla której metody doboru sąsiedztwa, algorytm zwraca lepsze wyniki, dla poszczególnych zestawów danych. Dla każdego rodzaju danych wywołaliśmy algorytm 6 razy dla danego kryterium stopu, którym u nas była liczba iteracji równa 10000, temperatura początkowa wynosiła 5000 oraz parametr alfa, który ustala szybkość spadku temperatury wyniósł 0,96. Zrobiliśmy to w celu wyznaczenia średniej wartości funkcji celu dla każdego z typu sąsiedztwa. Wyniki zostały zaprezentowane poniżej dla poszczególnych zestawów danych. Dokonując tych pomiarów, zbieraliśmy także informacje o najlepszym wyniku funkcji celu w całym algorytmie.

## Zestaw danych Dane\_PFSP\_100\_10.xlsx

|  |  |  |
| --- | --- | --- |
| rodzaj sąsiedztwa | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| swap | 5782,75 | 5732 |
| invert | 5802,00 | 5788 |

## Zestaw danych Dane\_PFSP\_200\_10.xlsx

|  |  |  |
| --- | --- | --- |
| rodzaj sąsiedztwa | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| swap | 10936,50 | 10884 |
| invert | 10897,00 | 10861 |

## Zestaw danych Dane\_PFSP\_50\_20.xlsx

|  |  |  |
| --- | --- | --- |
| rodzaj sąsiedztwa | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| swap | 4017,50 | 3984 |
| invert | 3990,25 | 3935 |

Podsumowując, w tym przypadku, dla danych ze stoma zadaniami oraz dziesięcioma maszynami, lepsza okazała się metoda swap. Dla pozostałych zestawów danych, lepsza okazała się metoda invert. Jak można zauważyć nie tylko średnie były lepsze dla tych metod, ale także najlepsza uzyskana wartość została uzyskana metodami, które wykazały także lepszą średnią. Więc w przyszłych obliczeniach dla Dane\_PFSP\_100\_10.xlsx użyjemy metody swap, a dla Dane\_PFSP\_200\_10.xlsx oraz Dane\_PFSP\_50\_20.xlsx metody invert.

Podsumowując analizę dotychczasowych parametrów, dla poszczególnych danych wiemy już, że:

* Dane\_PFSP\_100\_10.xlsx: sąsiedztwo: swap
* Dane\_PFSP\_200\_10.xlsx: sąsiedztwo: invert
* Dane\_PFSP\_50\_20.xlsx: sąsiedztwo: invert

Dotychczasowe najlepsze rozwiązania:

* Dane\_PFSP\_100\_10.xlsx: wartość funkcji celu: 5732
* Dane\_PFSP\_200\_10.xlsx: wartość funkcji celu: 10861
* Dane\_PFSP\_50\_20.xlsx: wartość funkcji celu: 3935

## Dobór metody redukcji temperatury

W tym przypadku sprawdzimy, czy wybrać geometryczną redukcję temperatury określoną wzorem: alfa\*T, czy wybrać powolny spadek temperatury określony wzorem: T/(1 + alfa \* T). Tak samo jak w powyższych przypadkach: liczba iteracji równa 10000, temperatura początkowa równa 5000 oraz parametr alfa równy 0,96.

### Zestaw danych Dane\_PFSP\_100\_10.xlsx

|  |  |
| --- | --- |
| metoda redukcji temperatury | średnia wartość f. celu |
| geometric | 5758,75 |
| slow | 5764,25 |

### Zestaw danych Dane\_PFSP\_200\_10.xlsx

|  |  |
| --- | --- |
| metoda redukcji temperatury | średnia wartość f. celu |
| geometric | 10906,5 |
| slow | 10947,5 |

### Zestaw danych Dane\_PFSP\_50\_20.xlsx

|  |  |
| --- | --- |
| metoda redukcji temperatury | średnia wartość f. celu |
| geometric | 4020,5 |
| slow | 3989,25 |

Podsumowując analizę dotychczasowych parametrów, dla poszczególnych danych wiemy już, że:

* Dane\_PFSP\_100\_10.xlsx: sąsiedztwo: swap, metoda redukcji temperatury: geometric
* Dane\_PFSP\_200\_10.xlsx: sąsiedztwo: invert, metoda redukcji temperatury: geometric
* Dane\_PFSP\_50\_20.xlsx: sąsiedztwo: invert, metoda redukcji temperatury: slow

Dotychczasowe najlepsze rozwiązania:

* Dane\_PFSP\_100\_10.xlsx: wartość funkcji celu: 5675
* Dane\_PFSP\_200\_10.xlsx: wartość funkcji celu: 10861
* Dane\_PFSP\_50\_20.xlsx: wartość funkcji celu: 3873

## Dobór temperatury początkowej

Teraz chcemy sprawdzić jaka wartość temperatury początkowej da nam najlepszą wartość funkcji celu. W tym celu dla naszego kryterium stopu, którym jest liczba iteracji włączyliśmy algorytm 7 razy, w celu obliczenia średniej dla każdej z temperatur początkowych, aby wybrać tą, dla której wartość funkcji celu jest najwyższa. Tutaj także, zapisujemy najniższe wartości funkcji celu dla każdej temperatury początkowe jakie sprawdzamy, zaprezentowane są w tabelce poniżej:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| temperatura początkowa | 500 | 1000 | 3000 | 5000 | 7000 | 9000 | 12000 |

### Zestaw danych Dane\_PFSP\_100\_10.xlsx

|  |  |  |
| --- | --- | --- |
| temperatura początkowa | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| 500 | 5791,2 | 5730 |
| 1000 | 5771,2 | 5732 |
| 3000 | 5744,4 | 5679 |
| 5000 | 5797 | 5766 |
| 7000 | 5774,2 | 5712 |
| 9000 | 5742,2 | 5675 |
| 12000 | 5799,4 | 5746 |

### Zestaw danych Dane\_PFSP\_200\_10.xlsx

|  |  |  |
| --- | --- | --- |
| temperatura początkowa | średnia wartość f, celu | najlepsza uzyskana wartość f. celu |
| 500 | 10952,2 | 10884 |
| 1000 | 10935,2 | 10896 |
| 3000 | 10924,2 | 10869 |
| 5000 | 10912 | 10874 |
| 7000 | 10974,2 | 10849 |
| 9000 | 10901,6 | 10860 |
| 12000 | 10963,4 | 10899 |

### Zestaw danych Dane\_PFSP\_50\_20.xlsx

|  |  |  |
| --- | --- | --- |
| temperatura początkowa | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| 500 | 4025 | 4008 |
| 1000 | 4006,6 | 3925 |
| 3000 | 4004,4 | 3955 |
| 5000 | 4073,2 | 3944 |
| 7000 | 3978,2 | 3873 |
| 9000 | 3972,2 | 3944 |
| 12000 | 4028,4 | 3964 |

Dla danych, gdzie maszyn jest 10, najlepszą temperaturą początkową jest 9000, a dla danych gdzie maszyn jest 20 i zadań mniej, bo 50, najlepsza temperatura początkowa wynosi 7000.

Podsumowując analizę dotychczasowych parametrów, dla poszczególnych danych wiemy już, że:

* Dane\_PFSP\_100\_10.xlsx: sąsiedztwo: swap, metoda redukcji temperatury: geometric, temperatura początkowa: 9000
* Dane\_PFSP\_200\_10.xlsx: sąsiedztwo: invert, metoda redukcji temperatury: geometric, temperatura początkowa: 9000
* Dane\_PFSP\_50\_20.xlsx: sąsiedztwo: invert, metoda redukcji temperatury: slow, temperatura początkowa: 7000

Dotychczasowe najlepsze rozwiązania:

* Dane\_PFSP\_100\_10.xlsx: wartość funkcji celu: 5675
* Dane\_PFSP\_200\_10.xlsx: wartość funkcji celu: 10860
* Dane\_PFSP\_50\_20.xlsx: wartość funkcji celu: 3873

## Dobór parametru alfa – szybkości redukcji temperatury

Sprawdzimy jaką wartość powinien mieć parametr alfa, aby wartość funkcji celu była jak najniższa.

Przetestujemy następujące wartości parametru alfa:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| alfa | 0,99 | 0,95 | 0,9 | 0,85 | 0,8 | 0,75 |

### Zestaw danych Dane\_PFSP\_100\_10.xlsx

|  |  |  |
| --- | --- | --- |
| alfa | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| 0,99 | 5753,4 | 5695 |
| 0,95 | 5767,4 | 5689 |
| 0,9 | 5731,8 | 5691 |
| 0,85 | 5774 | 5740 |
| 0,8 | 5744,6 | 5709 |
| 0,75 | 5744,6 | 5680 |

W tym wypadku, najlepsza średnia wartość f. celu nie pokrywa się z najlepszą uzyskaną wartością f. celu. Do dalszych obliczeń zdecydowaliśmy się wybrać parametr alfa równy 0,9, ponieważ średnia wartość f. celu może lepiej świadczyć, że dla przyszłych obliczeń częściej będzie można dostrzec lepsze wartości funkcji celu, a przy okazji najlepsze wartości f. celu dla parametrów alfa 0,9 oraz 0,75 nie różnią się znacząco.

### Zestaw danych Dane\_PFSP\_200\_10.xlsx

|  |  |  |
| --- | --- | --- |
| alfa | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| 0,99 | 10957,4 | 10873 |
| 0,95 | 10925,8 | 10849 |
| 0,9 | 10954,6 | 10915 |
| 0,85 | 10900,6 | 10822 |
| 0,8 | 10923,8 | 10832 |
| 0,75 | 10937,2 | 10818 |

W tym wypadku także kierujemy się średnią wartością funkcji celu i wybieramy parametr alfa = 0,85.

### Zestaw danych Dane\_PFSP\_50\_20.xlsx

|  |  |  |
| --- | --- | --- |
| alfa | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| 0,99 | 4012,8 | 3862 |
| 0,95 | 4054,8 | 3930 |
| 0,9 | 3992,6 | 3927 |
| 0,85 | 4026,4 | 3981 |
| 0,8 | 3991,8 | 3858 |
| 0,75 | 4040,8 | 3983 |

W tym przypadku wybieramy parametr alfa = 0,8.

Podsumowując analizę dotychczasowych parametrów, dla poszczególnych danych wiemy już, że:

* Dane\_PFSP\_100\_10.xlsx: sąsiedztwo: swap, metoda redukcji temperatury: geometric, temperatura początkowa: 9000, parametr alfa: 0,9,
* Dane\_PFSP\_200\_10.xlsx: sąsiedztwo: invert, metoda redukcji temperatury: geometric, temperatura początkowa: 9000, parametr alfa: 0,85
* Dane\_PFSP\_50\_20.xlsx: sąsiedztwo: invert, metoda redukcji temperatury: slow, temperatura początkowa: 7000, parametr alfa: 0,8

Dotychczasowe najlepsze rozwiązania:

* Dane\_PFSP\_100\_10.xlsx: wartość funkcji celu: 5675
* Dane\_PFSP\_200\_10.xlsx: wartość funkcji celu: 10818
* Dane\_PFSP\_50\_20.xlsx: wartość funkcji celu: 3858

## Dobór liczby iteracji

Chcemy dobrać jak najbardziej optymalną liczbę iteracji, tak aby wyniki były jak najlepsze oraz czas poświęcony na działanie algorytmu był jak najkrótszy.

Przetestujemy następujące wartości dla liczby iteracji:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| liczba iteracji | 2000 | 5000 | 10000 | 16000 | 22000 | 29000 |

W tym celu włączyliśmy algorytm kilka razy dla każdej z wartości iteracji, aby wyznaczyć średnią uzyskanych wyników dla danej liczby iteracji. Wyniki zostały zaprezentowane w tabelkach poniżej:

### Zestaw danych Dane\_PFSP\_100\_10.xlsx

|  |  |  |
| --- | --- | --- |
| liczba iteracji | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| 22000 | 5688.6 | 5664 |
| 29000 | 5707.2 | 5660 |
| 16000 | 5729.8 | 5692 |
| 5000 | 5745.6 | 5693 |
| 10000 | 5754.6 | 5696 |
| 2000 | 5891.6 | 5832 |

Wybraliśmy liczbę iteracji równą 22000. Średnia wartość f. celu dla takiej liczby iteracji była o około 30 niższa niż dla 2900.

### Zestaw danych Dane\_PFSP\_200\_10.xlsx

|  |  |  |
| --- | --- | --- |
| liczba iteracji | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| 29000 | 10771.0 | 10743 |
| 22000 | 10812.6 | 10747 |
| 16000 | 10877.4 | 10800 |
| 10000 | 10909.6 | 10833 |
| 5000 | 11014.2 | 10933 |
| 2000 | 11175.6 | 11093 |

Wybraliśmy liczbę iteracji równą 22000.

### Zestaw danych Dane\_PFSP\_50\_20.xlsx

|  |  |  |
| --- | --- | --- |
| liczba iteracji | średnia wartość f. celu | najlepsza uzyskana wartość f. celu |
| 10000 | 3959.4 | 3889 |
| 29000 | 3992.2 | 3895 |
| 16000 | 4007.4 | 3961 |
| 22000 | 4011.6 | 3944 |
| 5000 | 4046.2 | 4005 |
| 2000 | 4103.6 | 4070 |

Wybraliśmy liczbę iteracji równą 10000. Co zaskakujące uzyskała najlepsze rezultaty dla średniej jak i najlepszej wartości.

Podsumowując, zwykle większa liczba iteracji zwraca lepsze wyniki, ale nie zdarza się to zawsze. Czasem przy dobrze dobranych pozostałych parametrach takich jak temperatura początkowa, czy szybkość wyżarzania, mniejsza liczba iteracji także spisuje się świetnie.

Podsumowując analizę dotychczasowych parametrów, dla poszczególnych danych wiemy już, że:

* Dane\_PFSP\_100\_10.xlsx: sąsiedztwo: swap, metoda redukcji temperatury: geometric, temperatura początkowa: 9000, parametr alfa: 0,9, liczba iteracji: 22000.
* Dane\_PFSP\_200\_10.xlsx: sąsiedztwo: invert, metoda redukcji temperatury: geometric, temperatura początkowa: 9000, parametr alfa: 0,85, liczba iteracji: 22000.
* Dane\_PFSP\_50\_20.xlsx: sąsiedztwo: invert, metoda redukcji temperatury: slow, temperatura początkowa: 7000, parametr alfa: 0,8, liczba iteracji: 10000.

Dotychczasowe najlepsze rozwiązania:

* Dane\_PFSP\_100\_10.xlsx: wartość funkcji celu: 5660
* Dane\_PFSP\_200\_10.xlsx: wartość funkcji celu: 10743
* Dane\_PFSP\_50\_20.xlsx: wartość funkcji celu: 3858

## Analiza wpływu jakości rozwiązania początkowego na wielkość uzyskanego rozwiązania

W tym celu podczas dobierania najlepszej wartości parametru alfa, zapisywaliśmy także wartość rozwiązania początkowego oraz końcowego. Do zestawienia dołączyliśmy tabelki dla tych parametrów alfa, które dały największą sumę różnic pomiędzy początkowym a końcowym rozwiązaniem.

### Zestaw danych Dane\_PFSP\_100\_10.xlsx

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| alfa | | 0.85 |  | alfa | | 0.90 |
| F0 | Fbest | różnica |  | F0 | Fbest | różnica |
| 6554 | 5780 | 774 |  | 6401 | 5735 | 666 |
| 6948 | 5814 | 1134 |  | 6758 | 5691 | 1067 |
| 6907 | 5740 | 1167 |  | 6654 | 5719 | 935 |
| 6900 | 5743 | 1157 |  | 7026 | 5716 | 1310 |
| 6704 | 5793 | 911 |  | 6690 | 5798 | 892 |
| suma | | 5143 |  | suma | | 4870 |

### Zestaw danych Dane\_PFSP\_200\_10.xlsx

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| alfa | | 0.85 |  | alfa | | 0.95 |
| F0 | Fbest | różnica |  | F0 | Fbest | różnica |
| 12159 | 10822 | 1337 |  | 12829 | 10952 | 1877 |
| 12308 | 10834 | 1474 |  | 12694 | 10849 | 1845 |
| 12359 | 10891 | 1468 |  | 12027 | 10987 | 1040 |
| 12387 | 10907 | 1480 |  | 11977 | 10960 | 1017 |
| 12429 | 11049 | 1380 |  | 12056 | 10881 | 1175 |
| suma | | 7139 |  | suma | | 6954 |

### Zestaw danych Dane\_PFSP\_50\_20.xlsx

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| alfa | | 0.75 |  | alfa | | 0.99 |
| F0 | Fbest | różnica |  | F0 | Fbest | różnica |
| 4694 | 4086 | 608 |  | 4752 | 4004 | 748 |
| 4790 | 4018 | 772 |  | 4670 | 3935 | 735 |
| 4742 | 4055 | 687 |  | 4721 | 4055 | 666 |
| 4627 | 3983 | 644 |  | 4625 | 4208 | 417 |
| 4909 | 4062 | 847 |  | 4694 | 3862 | 832 |
| suma | | 3558 |  | suma | | 3398 |

Podsumowując, wartość rozwiązania początkowego nie ma wpływu na wartość rozwiązania końcowego. Jest to także niezależne od szybkości redukcji temperatury. Jak można zauważyć dla różnych parametrów alfa, wartości rozwiązania początkowego mogły być wysokie i schodziły do niskich wartości na koniec, ale także zdarzały się przypadki, gdy wartość na początku już była niska i podczas wyżarzania nie spadała bardzo znacząco. Można także zauważyć, że im wartość rozwiązania początkowego jest wyższa, tym różnica pomiędzy wartością początkową a końcową jest wyższa.

## Wielokrotne powtórzenie algorytmu dla dobranych parametrów

Teraz włączymy wielokrotnie algorytmy w celu uzyskania jak najniższej wartości funkcji celu. Ze względu na czas wykonywania algorytmu dodaliśmy także warunek liczby iteracji bez poprawy. Limit iteracji bez poprawy był u nas równy liczba iteracji/3, po tylu iteracjach bez poprawy algorytm przerywał swoje działanie. Podsumowanie wyników znajduje się poniżej:

Podsumowując analizę dotychczasowych parametrów, dla poszczególnych danych wiemy już, że:

* Dane\_PFSP\_100\_10.xlsx: sąsiedztwo: swap, metoda redukcji temperatury: geometric, temperatura początkowa: 9000, parametr alfa: 0,9, liczba iteracji: 22000.
* Dane\_PFSP\_200\_10.xlsx: sąsiedztwo: invert, metoda redukcji temperatury: geometric, temperatura początkowa: 9000, parametr alfa: 0,85, liczba iteracji: 22000.
* Dane\_PFSP\_50\_20.xlsx: sąsiedztwo: invert, metoda redukcji temperatury: slow, temperatura początkowa: 7000, parametr alfa: 0,8, liczba iteracji: 10000.

Takie wyniki mieliśmy na początku:

* Dane\_PFSP\_100\_10.xlsx: wartość funkcji celu: 5732
* Dane\_PFSP\_200\_10.xlsx: wartość funkcji celu: 10861
* Dane\_PFSP\_50\_20.xlsx: wartość funkcji celu: 3935

Udało nam się je zmniejszyć poprzez wybór odpowiednich parametrów:

* Dane\_PFSP\_100\_10.xlsx: wartość funkcji celu: 5606
* Dane\_PFSP\_200\_10.xlsx: wartość funkcji celu: 10633
* Dane\_PFSP\_50\_20.xlsx: wartość funkcji celu: 3806

W przypadku algorytmu symulowanego wyżarzania, staraliśmy się dobrać parametry tak aby algorytm był najbardziej optymalny i zwracał najlepsze wyniki funkcji celu. Symulowane wyżarzanie jest szybkim algorytmem, w dobrym czasie zwraca bardzo dobre wyniki, np. w porównaniu do Tabu Search. Szybkość spadku temperatury, czyli parametr alfa, był dla każdego z typu danych mniejszy bądź równy 0,9. Im więcej zadań do uszeregowania, tym powinniśmy ustawić więcej iteracji, aby wynik był lepszy, lecz wtedy długość obliczeń wzrasta.

# Algorytm Genetyczny

## Wstęp teoretyczny

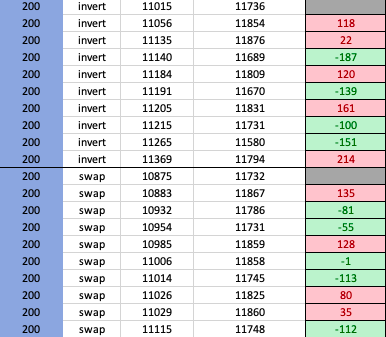
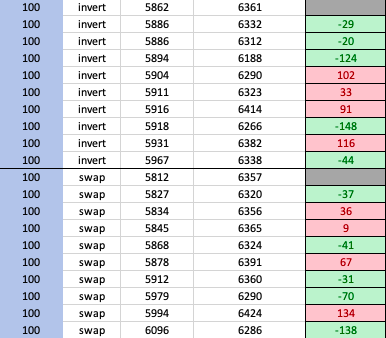
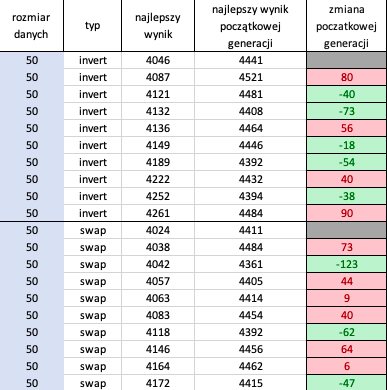
Jest to rodzaj heurystyki szukającej wystarczająco dobrego rozwiązania pośród przestrzeni potencjalnych rozwiązań. Nazwa algorytmów genetycznych nie została ustanowiona bez powodu. Algorytmy te w pewien sposób naśladują mechanizmy ewolucyjne, dobierając rodziców i tworząc z nich dzieci, przy jednoczesnym zachowaniu pewnych fragmentów, krzyżowaniu cech dwójki rodziców, a także dokonując mutacje.

## Działanie algorytmu

W pierwszej iteracji startowa populacja zostaje wylosowana, jest to ciąg kolejności zadań bez powtórzeń, której wielkość uzależniona jest od rozmiaru danych. Z populacji startowej wybierani są rodzice, w przypadku tego projektu zdecydowaliśmy się na użycie metod tournament oraz turnication. Metoda tournament polega na losowaniu dwójki osobników z bazowej populacji i wyborze lepszego osobnika jako rodzica (dla którego trasa jest krótsza). Schemat powtarzany dwukrotnie dla każdego dziecka, a następnie powtarzany jest aż do osiągnięcia oczekiwanej ilości rodziców. Na tak przygotowanej populacji rodziców przeprowadzana jest krzyżówka (crossover), w naszym przypadku korzystamy z metod PMX oraz OX, w obu metodach losowane są fragmenty rodziców, który przepisywane są do dzieci, a następnie reszta dziecka uzupełniania jest po kolei (zaczynając od prawej strony tego fragmentu), z drugiego rodzica z uwzględnieniem zasady, że wartości z poszczególnych komórek nie mogą się powtarzać. Po dokonaniu krzyżówki otrzymujemy populacje dzieci, populacja ta powinna dawać nie gorsze wyniki niż populacja rodziców, a w pożądanym przypadku będzie dawać wyniki lepsze. Na tym etapie musimy jeszcze przeprowadzić mutację. Jednym z parametrów funkcji jest szansa na mutację. Mutacje użyte w tym algorytmie są takie same jak w poprzednich algorytmach, używamy metod invert oraz swap. Mutacja rozwiązuje bardzo istotny problem, bez niej istnieje duże prawdopodobieństwo zamknięcia się naszych wyników w znacznie ograniczonym podzbiorze rozwiązań, w przypadku natrafienia na ekstremum lokalne albo na przykład, gdy osobniki populacji startowej są stosunkowo podobne do siebie. Po przejściu pełnej iteracji algorytmu populacja dzieci przekazywana jest do kolejnej iteracji, jako populacja rodziców. Algorytm powtarzany jest do uzyskania oczekiwanej ilości populacji lub do momentu, w którym przez 10 populacji pod rząd najlepszy wynik dla danej populacji się nie poprawi. Algorytm klasyfikuje rozwiązania na podstawie funkcji celu, liczy ona łączny czas wykonania wszystkich zadań na wszystkich maszynach, z uwzględnieniem faktu, że aby rozpocząć wykonywanie zadania na danej maszynie, maszyna ta musi być wolna, to znaczy poprzednie zadanie musi zostać ukończone, lub maszyna ta musi być użyta po raz pierwszy w iteracji.

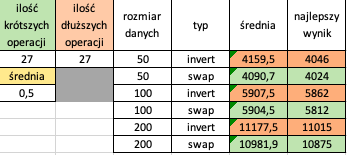
## Analiza doboru sąsiedztwa oraz wpływ początkowej wartości funkcji celu na finalny wynik

Pierwszym parametrem który analizowaliśmy jest metoda doboru sąsiedztwa. Tak jak wspomnieliśmy we wstępie, używane przez nas metody to invert oraz swap. Algorytm uruchomiliśmy wielokrotnie dla każdej z metod oraz dla każdego zbioru danych, wyniki prezentujemy w poniższych tabelach . Kolorami zielonymi oraz czerwonymi oznaczyliśmy zmianę wartości w kolumnie “najlepszy wynik początkowej generacji”, wartość dodatnia oznacza zwiększenie funkcji celi o daną wartość, na przykład wartość równa 80 oznacza, że kolejny wynik był dłuższy o 80, a zatem jest on gorszy niż wynik powyżej niego. Dane podzieliliśmy na grupy zawierające poszczególne rozmiary danych oraz metody doboru sąsiedztwa, wewnątrz każdej grupy wyniki posegregowaliśmy rosnąco według najlepszego wyniku.



Na tym etapie nasze najlepsze wyniki to :   
dane 50 : 4024 , dane 100 : 5812 , dane 200 : 10875

Do wykonania analizy stworzyliśmy dodatkową tabelę, dodatkowo zdecydowaliśmy się na przeanalizowanie wpływu najlepszej wartości funkcji celu dla populacji początkowej, na finalny wynik.

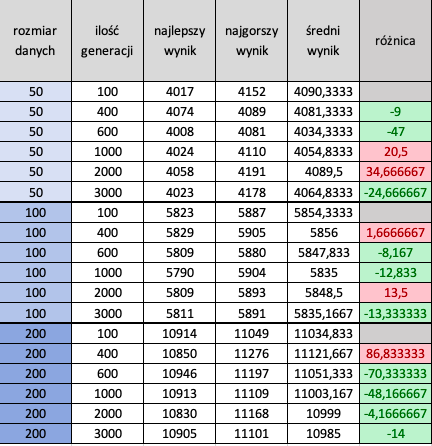


Na tym etapie możemy wyciągnąć wnioski odnośnie obu analizowanych parametrów.

1. Po posegregowaniu danych od najmniejszej do największej wartości funkcji celu (w finalnym wyniku, to znaczy najlepszym dla danego uruchomienia algorytmu) okazało się, że poprawa i pogorszenie najlepszego wyniku dla populacji początkowej występują tyle samo razy, średnia zmiana wynosi 0,5 , oznacza to że dla kolejnych, coraz to gorszych wyników końcowych, wynik początkowy pogarsza się średnio o 0,5. Na podstawie tych wartości wyciągamy wnioski, że wpływ najlepszej wartości funkcji celu dla populacji początkowej na finalny wynik jest prawie niezauważalny, uznajemy go więc za nieistotny.
2. Zarówno w przypadku średnich jak i najlepszych wyników metoda doboru sąsiedztwa swap zwracała lepsze wyniki niż metoda invert, we wszystkich przypadkach. Z tego powodu w późniejszych analizach i iteracjach odrzucamy metodę invert i przyjmujemy metodę swap.

## Analiza wpływu ilości generacji

W celu przeanalizowania wpływu ilości generacji (pokoleń) na wynik finalny, algorytm uruchomiliśmy wielokrotnie dla każdego rodzaju danych oraz dla następujących ilości generacji : 100, 400, 600, 1000, 2000, 3000 . Wyniki prezentujemy w tabeli poniżej.



Podczas przeprowadzania analizy skupiliśmy się głównie na wyniki średnie, ponieważ po przeprowadzeniu całej analizy algorytm uruchomiliśmy jeszcze ponad 100 razy dla każdego rozmiaru danych, więc mała wartość średnia zwiększała szane na uzyskanie najlepszego wyniku. Kolumna różnica przedstawia różnicę wyników średnich dla każdego kolejnego rodzaju parametru w porównaniu z poprzednią wartością.

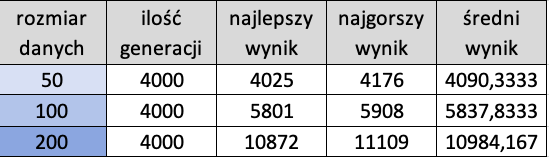
Podsumowanie wyników przedstawia poniższa tabela :



Na podstawie obu tabel możemy wyciągnąć następujące wnioski:

Mimo pojedynczych odstępstw od tej reguły, możemy zaobserwować, że minimalne, a przede wszystkim średnie wyniki wychodzą najmniejsze dla 3000 iteracji w przypadku każdego zestawu danych. W ogólnym ujęciu zwiększenie ilości generacji powinno poprawić, a przynajmniej nie powinno znacznie pogorszyć średnich wyników (należy pamiętać, że w tym algorytmie występuje element losowy), z tego powodu uruchomiliśmy również algorytm dla 4000 generacji, jednak zarówno w przypadku wartości minimalnej jak i wartości średniej nie zauważyliśmy różnicy w stosunku do 3000 generacji, w związku z czym w projekcie przyjęliśmy ilość generacji równą 3000.

Wyniki dla 4000 generacji przedstawiamy w tabeli poniżej :

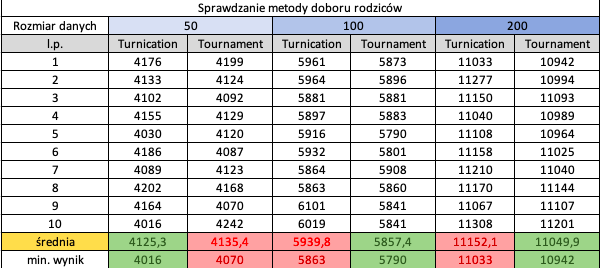


Podczas przeprowadzania analiz następnych parametrów przyjmujemy ilość generacji równą 200 w celu zmniejszenia czasu obliczenia poszczególnej iteracji, co pozwoliło nam na przeprowadzenie większej ilości obliczeń, a w rezultacie skutkowało dokładniejszą analizą wpływu tych parametrów. Po przeprowadzeniu analiz wszystkich parametrów uruchomiliśmy wielokrotnie kod (ponad 100 razy) dla 3000 generacji.

Na tym etapie nasze najlepsze wyniki to :   
dane 50 : 4008 , dane 100 : 5790 , dane 200 : 10830

## Analiza wpływu metody doboru rodziców

W projekcie analizowaliśmy wpływ dwóch metod doboru rodziców - tournament selection oraz turnication selection, dokładnie tak samo jak w poprzednim projekcie. Dla każdego wariantu oraz zbioru danych uruchomiliśmy algorytm 10 razy, wyniki przedstawiamy w poniższych tabelach:





Zarówno w przypadku minimalnych wartości funkcji celu, jak i wyników średnich dla danych metod doboru rodziców, Tournament selection okazało się lepsze od Turnication selection w przypadku każdego zestawu danych. Wynik ten uwarunkowany jest faktem, że w przypadku Turnication selection algorytm podczas wybierania rodziców nie sprawdza, który z kandydatów jest lepszy, a jedynie ich losuje. W pozostałej części proejktu przyjmujemy więc metodę tournament jako naszą metodę doboru rodziców.

Na tym etapie nasze najlepsze wyniki to :   
dane 50 : 4008 , dane 100 : 5790 , dane 200 : 10830

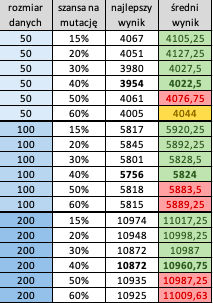
## Analiza wpływu szansy na mutację

Jednym z analizowanych przez nas parametrów jest szansa na przeprowadzenie mutacji.

Analizę tą przeprowadziliśmy dla wartości 15%, 20%, 30%, 40%, 50%, 60%.

Algorytmy puściliśmy wielokrotnie. Przypadkiem puściliśmy algorytm dla 3000 generacji zamiast dla 200, a zatem wyniki mogą być tu bardziej zbliżone do finalnych wyników niż to będzie miało miejsce w przypadku analizy kolejnych parametrów.

Zbyt mała szansa na mutację może narazić nas na utknięcie w ekstremum lokalnym funkcji celu, z drugiej strony zbyt duża szansa może powodować za dużą zmienną populacji, tan element losowy może negatywnie wpływać na ewolucję kolejnych generacji.

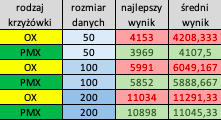


Jak możemy zaobserwować, dla wszystkich rozmiarów danych zarówno najniższy jak i najniższy średni wynik występują przy szansie na mutację równą 40%, dla wartości mniejszych możemy zaobserwować poprawę wyników wraz ze wzrostem tej szansy, a dla wartości wyższych pogorszenie się wyników wrac ze wzrostem szansy na mutację, jedyne odstępstwo od tej reguły zaobserwowaliśmy w przypadku danych 50 i szansy na mutację równej 60%, jednak wyniki były wtedy wciąż gorsze niż dla 40%. Z tego powodu do dalszych analiz przyjmujemy szansę na mutację na poziomie 40%.

Na tym etapie nasze najlepsze wyniki to :   
dane 50 : 3954, dane 100 : 5756, dane 200 : 10830

## Analiza wpływu rodzaju funkcji crossover – rodzaju krzyżówki

Zgodnie z opisem we wstępie, analizujemy wpływ funkcji PMX oraz funkcji OX. Wyniki przedstawiamy w tabeli poniżej :

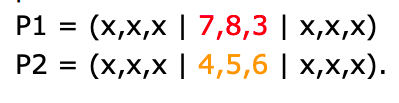


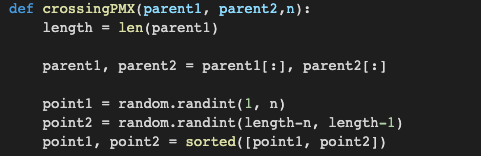
Jak możemy zaobserwować, dla wszystkich zestawów danych metoda PMX okazała się lepsza od OX zarówno w przypadku najlepszego jak i średniego wyniku.

Na tym etapie nasze najlepsze wyniki to :   
dane 50 : 3954, dane 100 : 5756, dane 200 : 10830

## Analiza wpływu maksymalnej szerokości przedziałów bocznych w metodzie PMX

Zgodnie z opisem zawartym w prezentacji w metodzie PMX wybieramy punkty na podstawie których przeprowadzana jest krzyżówka.

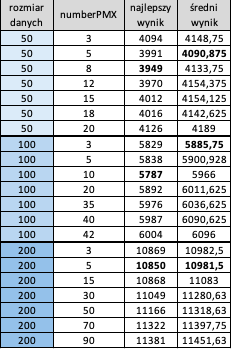




W przypadku naszego algorytmu punkty te są losowane. Analizowany przez nas parametr, oznaczany później jako numberPMX, oznaczony jest w powyższym fragmencie kodu poprzez literę n , oznacza on maksymalną możliwą szerokość przedziałów bocznych, a więc indeks punktów przecięcia genu. Punkty te losowane są z przedziałów [1, n] oraz [length-n, length-1] , włącznie z granicami przedziałów, gdzie length oznacza długość danego osobnika populacji, a więc wielkość danych.

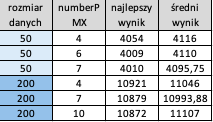
Z tego powodu zdecydowaliśmy się na przeanalizowanie różnych wartości tego parametru zależnie od rozmiaru danych, gdyż numberPMX = 10 dla danych 50 jest równoznaczne numberPMX = 2- dla danych 100, zależnie od rozmiaru danych proporcje mogą potencjalnie mieć różny wpływ na wynik.

Wyniki obliczeń przedstawiamy w poniższej tabeli :



W przypadku danych o rozmiarach 50 i 200 zdecydowaliśmy się na dodatkowe przeanalizowanie kilku wartości tego parametru, wartości które są w okolicach najlepszych średnich wyników z powyższej tabeli . Natomiast dla danych o rozmiarze 100 widzimy zauważalnie najlepszą średnią dla parametru numberPMX = 3, a następnie pogarszanie się średniej wraz ze wzrostem tego parametru. Stwierdziliśmy, że analizowanie wartości mniejszych niż 5 nie ma sensu i powodowałoby przeprowadzenie bardzo znikomej krzyżówki.

Dodatkowe obliczenia przedstawiamy w poniższej tabeli:

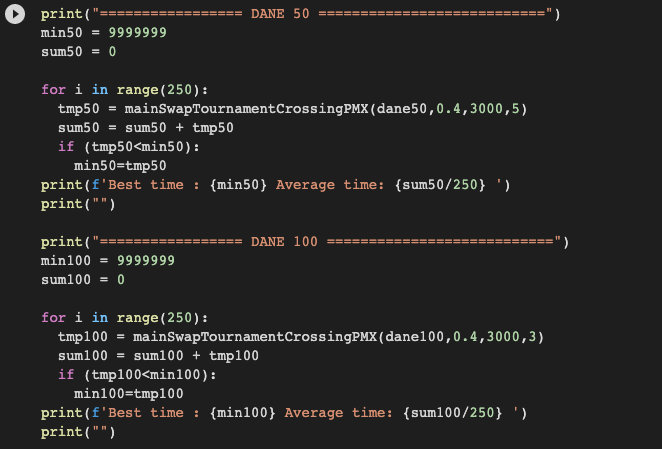


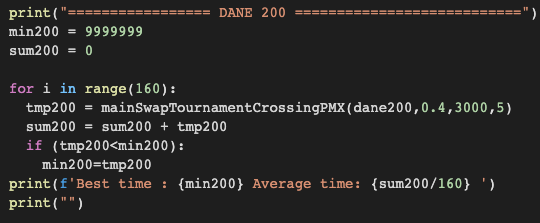
Jak możemy zaobserwować, najlepsze wyniki uzyskujemy dla parametru numberPMX równego 5 dla danych o rozmiarze 50 oraz 200, a także dla wartości 3 w przypadku danych o rozmiarze 100 .

Na tym etapie nasze najlepsze wyniki to :   
dane 50 : 3949, dane 100 : 5756, dane 200 : 10830

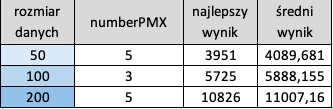
## Przeprowadzenie finalnych obliczeń

Na tym etapie znamy już wszystkie parametry funkcji i ich wpływ na finalny wynik. Kod który uruchamialiśmy wyglądał w następujący sposób ( jest to przykład tylko dwóch z kilku uruchomionych komórek, komórki uruchamialiśmy dla różnej liczności wykonania pętli )





Wielokrotnie uruchomiliśmy każdy z algorytmów, a ich wyniki ponownie umieściliśmy w tabeli poniżej :



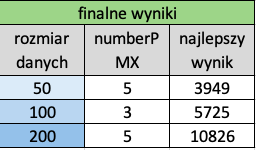
Porównując te wyniki z wcześniej uzyskanymi najlepszymi wynikami okazuje się, że udało nam się poprawić wyniki dla danych o rozmiarze 100 i 200 , natomiast w przypadku danych o rozmiarze 50 w dalszym stopniu najlepszym wynikiem jest wynik 3949.

Nasze najlepsze wyniki to :   
dane 50 : 3949, dane 100 : 5725, dane 200 : 10826

## Podsumowanie

**Zestawienie wyników**

Zebranie finalnych wyników prezentuje się w następujący sposób :



Wyniki uzyskiwane są dla następujących ciągów zadań :

Dane 50, wynik 3949:

[24, 4, 39, 22, 5, 28, 12, 8, 15, 19, 32, 36, 45, 16, 11, 21, 20, 48, 46, 9, 23, 6, 18, 29, 37, 50, 30, 1, 25, 49, 27, 47, 2, 44, 3, 41, 31, 42, 38, 26, 14, 43, 7, 35, 33, 17, 10, 13, 40, 34]

Dane 100, wynik 5725:

[33, 65, 83, 79, 13, 4, 56, 35, 1, 45, 16, 27, 51, 80, 54, 97, 73, 61, 48, 18, 92, 75, 42, 5, 85, 60, 58, 94, 76, 30, 87, 66, 74, 39, 68, 72, 82, 96, 14, 91, 57, 12, 47, 21, 9, 22, 100, 63, 36, 99, 77, 84, 88, 23, 19, 34, 11, 28, 81, 8, 53, 2, 7, 89, 95, 40, 70, 10, 52, 86, 55, 93, 43, 46, 62, 24, 71, 15, 50, 38, 67, 44, 69, 6, 78, 41, 37, 20, 3, 17, 29, 98, 90, 64, 26, 31, 59, 25, 49, 32]

Dane 200, wynik 10826:

[58, 148, 37, 35, 92, 22, 166, 110, 173, 158, 49, 83, 181, 150, 25, 134, 152, 7, 99, 163, 20, 46, 176, 80, 29, 59, 21, 178, 184, 88, 109, 182, 188, 167, 24, 123, 154, 131, 1, 165, 191, 168, 112, 27, 56, 95, 175, 192, 43, 15, 45, 147, 117, 93, 139, 32, 146, 84, 179, 3, 2, 133, 193, 85, 164, 19, 75, 155, 86, 97, 129, 89, 145, 52, 171, 138, 51, 198, 48, 8, 57, 54, 77, 140, 170, 68, 44, 120, 18, 28, 108, 9, 72, 62, 156, 87, 82, 26, 122, 69, 114, 71, 125, 103, 124, 144, 115, 31, 41, 142, 121, 190, 73, 30, 81, 162, 136, 177, 5, 11, 65, 128, 197, 12, 47, 143, 135, 174, 186, 61, 160, 60, 132, 185, 149, 96, 17, 14, 38, 16, 157, 116, 101, 36, 102, 187, 151, 34, 67, 119, 13, 169, 50, 199, 159, 161, 74, 66, 130, 172, 113, 70, 78, 4, 195, 98, 55, 141, 118, 76, 189, 153, 40, 53, 126, 183, 42, 39, 33, 137, 105, 6, 111, 90, 104, 23, 91, 100, 194, 64, 107, 106, 127, 94, 200, 180, 10, 196, 63, 79]

**Wpływ parametrów na wynik**

W tym projekcie w przypadku algorytmu genetycznego analizowaliśmy wpływ 7 parametrów na kocowy wynik. Parametrami tymi były :

* Najlepsza trasa dla początkowej populacji
* Metoda doboru sąsiedztwa - sposób mutacji
* Ilość populacji
* Metoda doboru rodziców
* Szansa na przeprowadzenie mutacji
* Rodzaj krzyżówki - funkcja crossover
* Maksymalna szerokość bocznych przedziałów podczas krzyżówki

Jak się okazało najlepsza trasa dla początkowej populacji nie ma większego wpływu na finalny wynik, natomiast w przypadku pozostałych parametrów następujące wartości okazały się optymalne :

* Sposób mutacji: SWAP
* Ilość populacji: 3000
* Metoda doboru rodziców: tournament selection
* Szansa na przeprowadzenie mutacji: 40%
* Rodzaj krzyżówki: Partially mapped crossover (PMX)
* Maksymalna szerokość bocznych przedzialów: 5 dla danych 50 i 200, 3 dla danych 100

# Algorytm wspinaczki (HILL CLIMBING)

## Wprowadzenie

Algorytm wspinaczkowy, jedna iteracja zwraca pełny wynik rozwiązania z rozwiązania który jest w sąsiedztwie do poprzedniego rozwiązania. Wynik gorszy jest odrzucany a wynik lepszy poddawany jest algorytmowi od początku aż do momenty gdy wyniki będzie lokalnie najlepszy. Wadą algorytmu jest wysoka zależność pomiędzy rozwiązaniem startowym a wynikiem, należy więc wiele razy modyfikować rozwiązanie początkowe w celu znalezienia najlepszego rozwiązania. Do wada należy również nieodporność algorytmu na maksima lokalne których nie da się już poprawić a mogą nie być globalnie optymalne. Zaletami algorytmu jest szybkie działanie.

## Zasada działania:

1. Losujemy rozwiązanie początkowe i ustalmy jej wynik (czas realizacji zadań).
2. Tworzymy listę sąsiadów dla aktualnego rozwiązania, (sposób tworzenia sąsiedztwa może być parametrem algorytmu)
3. Sprawdzamy czy lista sąsiadów nie jest pusta jeśli tak to kończymy algorytm rozwiązanie zwracając aktualny wynik jako maksimum lokalne. W przypadku gdy lista nie jest pusta przechodzimy do kroku następnego.
4. Losujemy jednego z sąsiadów. Sprawdzamy dla niego wynik, jeśli wynik okazał się lepszy, ustalamy sąsiada jako aktualne rozwiązanie i wracamy do punktu 2.
5. Jeśli wynik jest gorszy usuwamy rozwiązanie z listy sąsiadów i wracamy do punktu 3.

Dany algorytm należy przeprowadzić wiele razy losując za każdym razem nowe rozwiązanie początkowe.

## Optymalizacja parametrów

### Opis Parametrów:

**Liczba iteracji wewnątrz algorytmu –** parametr ten ustawiamy aby ograniczyć ilość przejść między rozwiązaniami w metodzie (1 iteracja to przejście z 1 rozwiązania do „lepszego” sąsiada jako nowego aktualnego rozwiązania) – w skrypcie nazwa zmiennej to ihc\_itteration

**Wartość startowa –** Wartość ta określa maksymalną długość trasy dla trasy początkowej, która wybierana jest losowo, algorytm nie przejdzie do kolejnego kroku dopóki wylosowana trasa początkowa nie osiągnie wystarczająca małej trasy.

**Rodzaj sąsiedztwa:**

**Insert –** w ten rodzaj sąsiedztwo wchodzą wszystkie rozwiązania które od aktualnego rozwiązania różnią się 1 indeksem który przełożony jest pomiędzy inne 2 rożne indeksy, pozostała część rozwiązania musi być identyczna [1,2,3,4] -> [2,**1**,3,4],[2,3,**1**,4],[2,3,4,**1**][1,3,**2**,4][1,3,4,**2**][**3,**1,2,4]…..

**SWAP –** tak jak przy Tabu Search

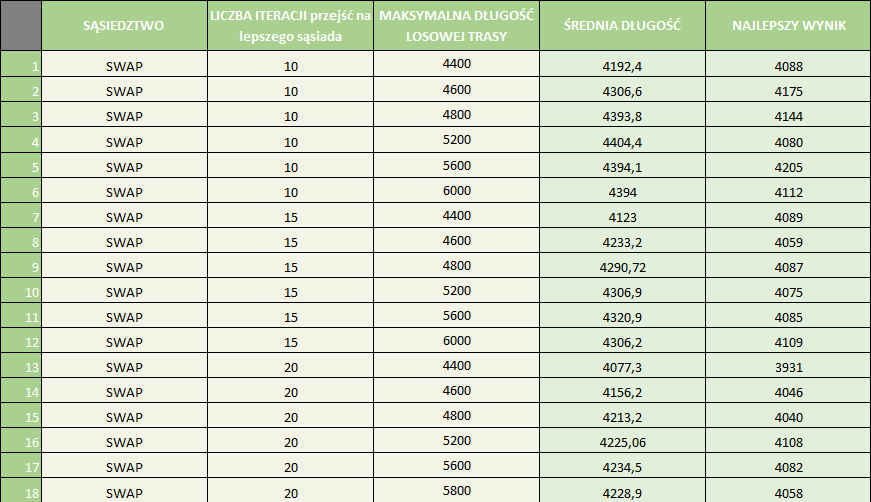
**INVERT –** tak jak przy Tabu Search

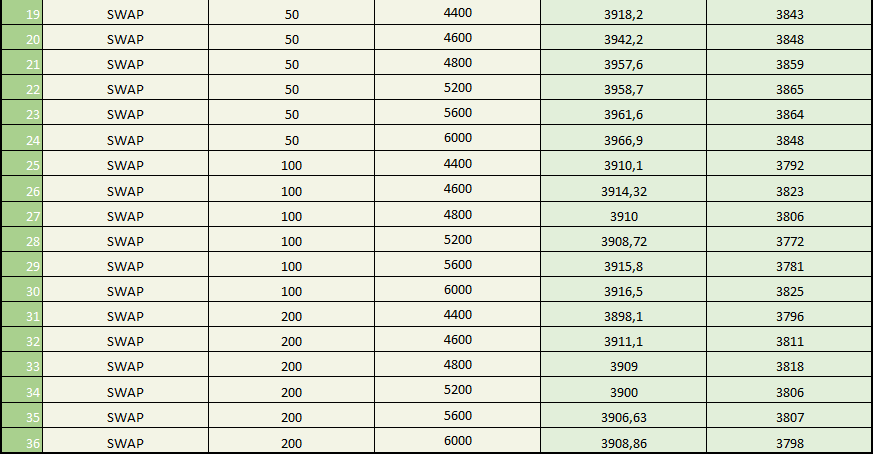
**Losowanie miasta startowego –** Losowanie miasta startowego wykonuje się zawsze na samym początku algorytmu więc im więcej razy chcemy wykonać algorytm w celu poszukiwania jak najlepszego wyniku, tym większy musimy ustawić ten parametr - większe znaczeni ma sprawdzenie powyższych 3 parametrów gdyż tą wartość ustalimy na jak najwyższą

### Zestaw Danych PFSP\_50\_20.xlsx

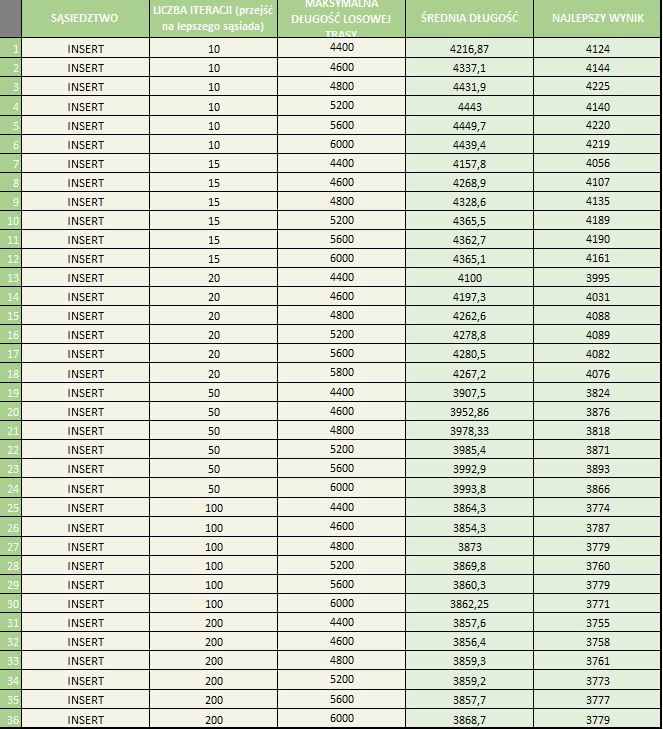
Dla każdej kombinacji zostało wykonane 100 iteracji algorytmu ( 1 iteracja = podanie jednego minimum lokalnego zaczynając 1rozwiązanie początkowe )

SWAP:

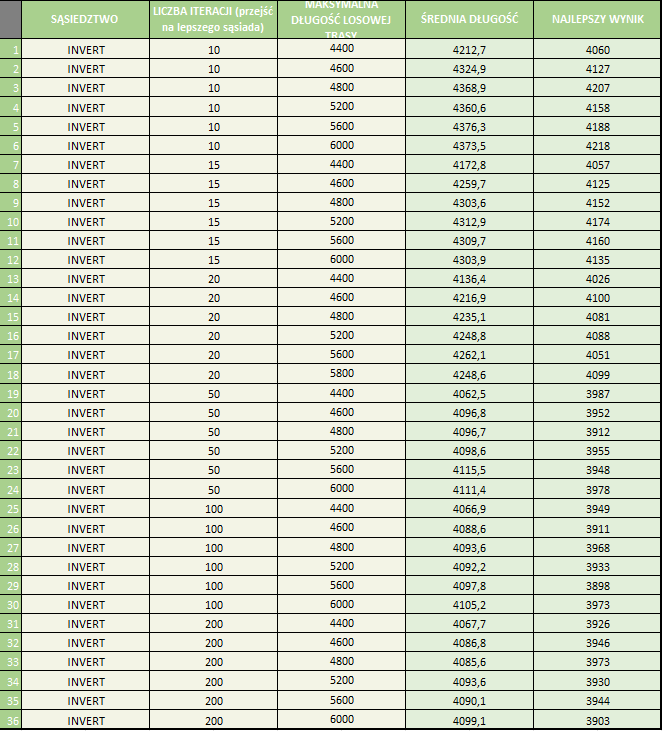




INSERT:



INVERT:



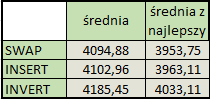
**Najlepszy wynik** : 3760 Najlepsza trasa : [24, 4, 28, 45, 25, 8, 46, 10, 11, 2, 31, 23, 16, 27, 49, 5, 15, 9, 39, 17, 30, 20, 32, 33, 29, 43, 37, 44, 1, 21, 3, 22, 50, 34, 48, 47, 19, 6, 36, 42, 26, 38, 14, 35, 41, 13, 12, 7, 18, 40]

Najlepszy rezultat dały parametry:

* Sąsiedztwo: **INSERT**
* iterację wewnątrz algorytmu: 1**00**
* start distacne: **5200**

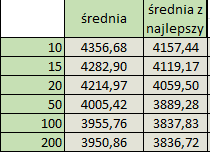
Przeanalizujmy jeszcze średnie dla danych parametrów:

**Sąsiedztwo:**



Widzimy bardzo podobne wyniki dla sąsiedztwa swap i insert, jednak wyniki invert są zdecydowanie gorsze od 2 poprzednich. W przypadku gdzie daliśmy możliwość większej iteracji wewnątrz algorytmu lepsze wyniki prezentował insert gdy mamy na nie duże ogranicznia lepiej wypada swap.

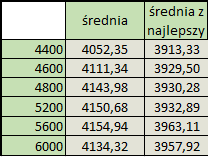
**Iteracje wewnątrz algorytmu:**



W przypadku liczb iteracji wewnątrz algorytmu najlepszym wyborem wydaje się zwiększanie do 200 iteracji, wartości dla 100 i 200 są bardzo zbliżone, jednak przy założymy na ten moment że jest sens zwiększanie tego parametru do 200.

**Maksymalny wynik początkowy losowy:**

Widać jak najlepiej ustawić najniższą możliwą( przy wartości niższej niż 4400 konieczność restartowania kernela bez zwrócenia wyniku) maksymalną wartość wyniku początkowego losowego.



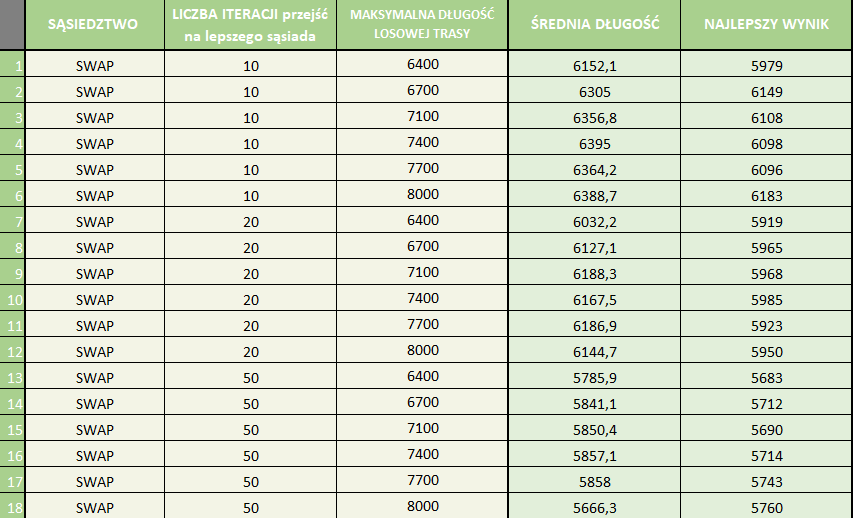
Analizując średnie oraz najlepsze wyniki dla danych parametrów, dalsze poszukiwania najlepszego rozwiązania należy poszukiwać ustawiając parametry na:

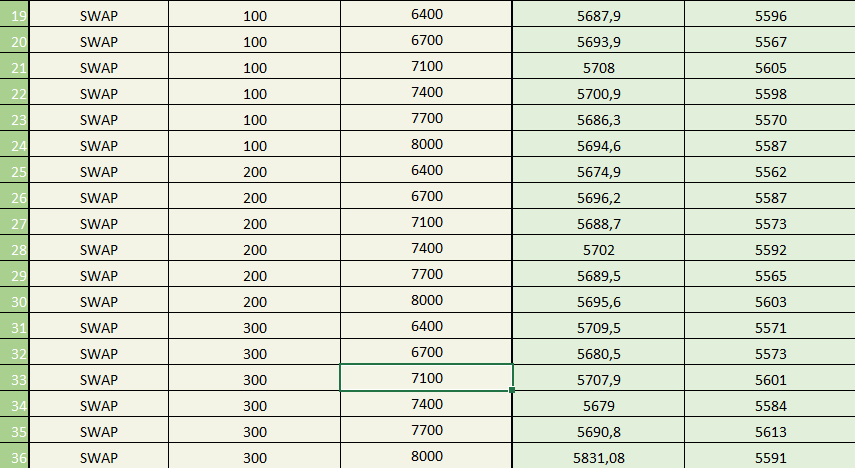
* Sąsiedztwo: INSERT
* Iterację wewnątrz algorytmu : 200
* Start distance max: 4400

### Zestaw Danych PFSP\_100\_10.xlsx

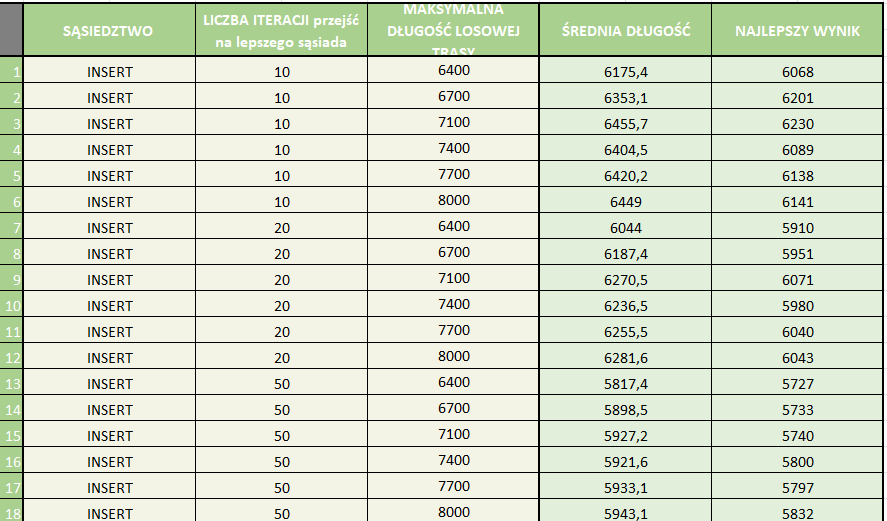
Dla każdej kombinacji zostało wykonane 50 iteracji algorytmu ( 1 iteracja = podanie jednego minimum lokalnego zaczynając 1rozwiązanie początkowe )

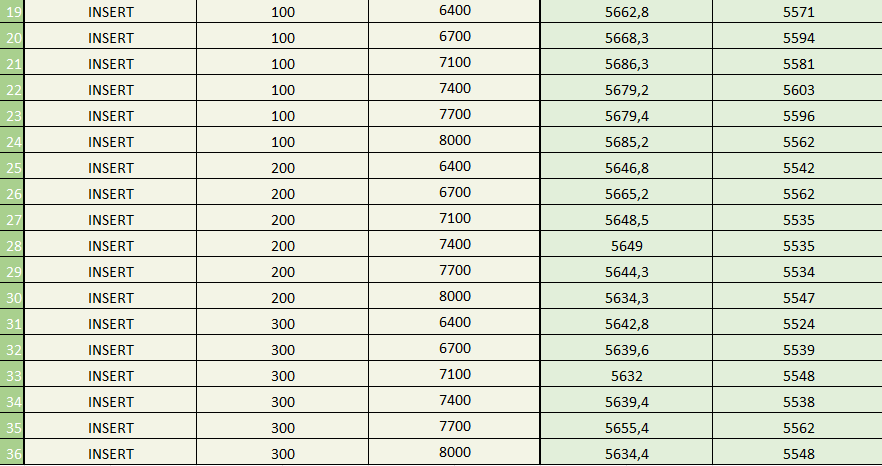
**SWAP :**



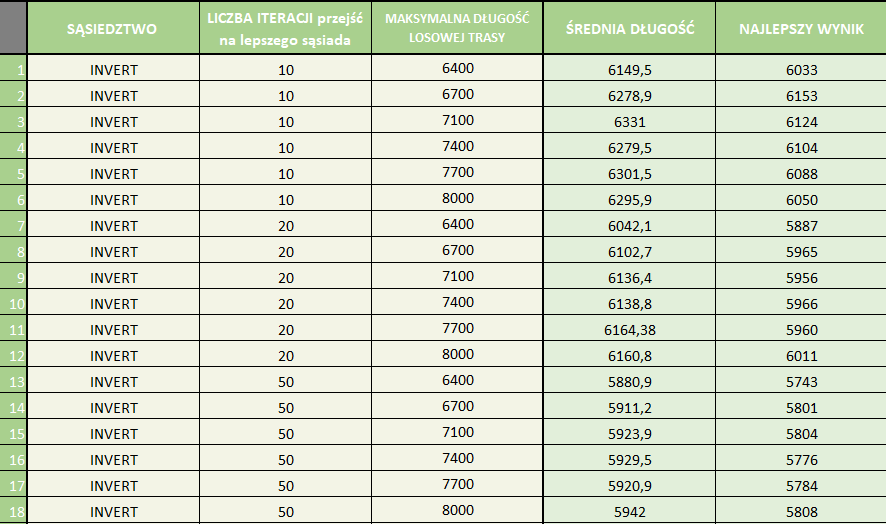


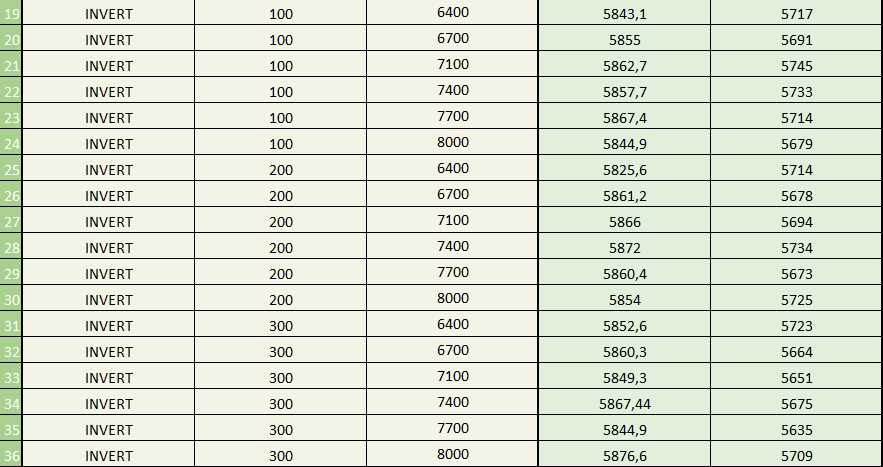
**INSERT:**





**INVERT:**





Najlepszy rezultat dały parametry:

* Sąsiedztwo: **INSERT**
* iterację wewnątrz algorytmu: **300**
* start distacne: **6400**

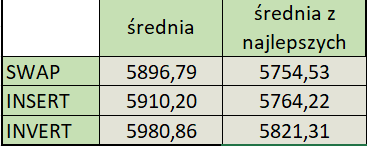
**Wynik: 5524**

Najlepsza trasa : [79, 83, 65, 75, 56, 97, 87, 44, 4, 90, 27, 23, 16, 74, 91, 36, 73, 68, 81, 86, 9, 66, 10, 25, 18, 85, 42, 1, 57, 22, 2, 49, 39, 98, 33, 100, 76, 99, 58, 63, 48, 41, 24, 55, 19, 46, 14, 35, 28, 72, 7, 21, 61, 6, 8, 88, 38, 54, 15, 92, 5, 45, 89, 96, 51, 71, 60, 50, 70, 67, 20, 59, 47, 37, 32, 94, 93, 30, 53, 17, 80, 84, 62, 78, 64, 95, 34, 12, 11, 31, 40, 77, 3, 69, 13, 43, 52, 29, 82, 26]

Przeanalizujmy jeszcze średnie dla danych parametrów:

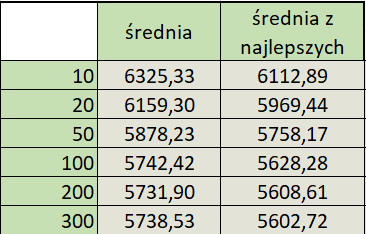
**Sąsiedztwo:**

Widzimy bardzo podobne wyniki dla sąsiedztwa swap i insert, jednak wyniki invert są zdecydowanie gorsze od 2 poprzednich. Jednak tak jak w przypadku dla 50 danych wyniki w momencie ustawienia iteracji na wyższy poziom lepsze osiągaja sąsiedztwo insert.



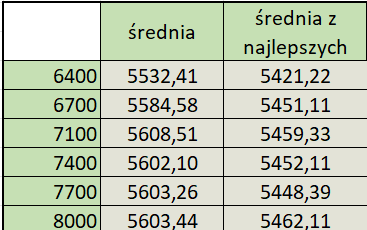
**Iteracje wewnątrz algorytmu:**

W przypadku liczb iteracji wewnątrz algorytmu najlepszym wyborem wydaje się zwiększanie do 200 iteracji, ponieważ wartości dla 200 i 300 są bardzo zbliżone (wśród najlepszych wyników przy 300 iteracjach i tak nie przekroczona 200 iteracji)



**Maksymalny wynik początkowy losowy:**

Widać jak najlepiej ustawić najniższą możliwą( przy wartości niżeszej niż 6400 konieczność restartowania kernela bez zwrócenia wyniku) maksymalną wartość wyniku początkowego losowego.



Najlepszym więc wyborem wydaje się uruchomienia większej ilości prób dla parametrów:

* Sąsiedztwo: INSERT
* Iterację wewnątrz algorytmu : 200
* Start distance max: 6400

I dla takich parametrów zostaną wykonane dalsze iteracje w celu znalezienia najlepszego wyniku.

### Zestaw Danych PFSP\_200\_10.xlsx

Dla każdej kombinacji zostało wykonane 20 iteracji algorytmu ( 1 iteracja = podanie jednego minimum lokalnego zaczynając 1rozwiązanie początkowe )

W przypadku danych dla 200, do optymalizacji 1 parametru zostały wybrane wartości stałe dla pozostałych parametrów, podane zostały one w szarej ramce gdzie:

Kn – rodzaj sąsiedztwa

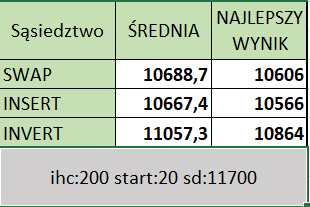
Start – liczba iteracji algorytmu

Sd – maksymalny wynik rozwiązania losowego

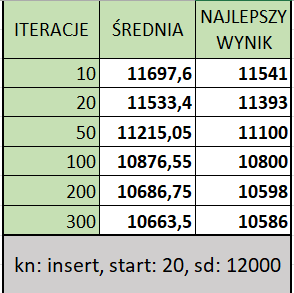
Ihc – iteracje wewnątrz algorytmu.

Parametry stałe zostały dobrane analizując wyniki dla 50 i 100 zadań.

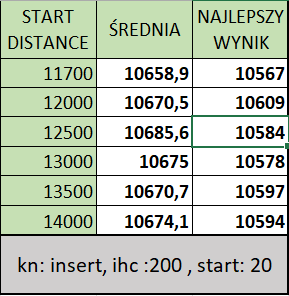
**Sąsiedztwo:**



**Ilość iteracji wewnątrz algorytmu:**



**Maksymalny dystans startowy:**



Na tym etapie najlepszy wynik został osiągniety:

Najlepszy wynik : **10567**

Najlepsza trasa : [22, 97, 184, 144, 1, 150, 135, 146, 173, 179, 183, 71, 65, 143, 28, 84, 68, 77, 27, 16, 166, 171, 185, 21, 96, 81, 20, 117, 89, 110, 83, 200, 145, 159, 181, 163, 94, 141, 33, 193, 57, 56, 35, 24, 131, 70, 5, 174, 90, 25, 76, 45, 100, 198, 113, 72, 48, 191, 139, 156, 88, 103, 133, 59, 34, 112, 151, 168, 175, 7, 50, 162, 137, 52, 31, 102, 119, 37, 114, 93, 95, 130, 170, 32, 74, 158, 73, 40, 54, 8, 58, 15, 155, 178, 66, 176, 91, 29, 172, 43, 126, 111, 42, 80, 125, 3, 192, 46, 19, 23, 61, 6, 148, 116, 165, 11, 49, 164, 86, 147, 17, 194, 140, 36, 169, 92, 47, 167, 197, 9, 128, 154, 4, 195, 152, 138, 78, 160, 122, 99, 2, 104, 62, 101, 161, 12, 153, 123, 149, 132, 98, 199, 124, 85, 63, 51, 41, 188, 60, 182, 107, 38, 157, 134, 187, 108, 13, 120, 87, 180, 109, 67, 189, 105, 55, 44, 82, 190, 196, 127, 106, 79, 30, 115, 177, 26, 14, 142, 118, 136, 129, 75, 69, 53, 121, 18, 39, 186, 10, 64]

Analizując średnie oraz najlepsze wyniki dla danych parametrów, dalsze poszukiwania najlepszego rozwiązania należy poszukiwać ustawiając parametry na:

* Sd – 11700
* Iteracje – 300
* Sąsiedztwo - insert

Parametr jakim jest liczba iteracji całego algorytmu nie była optymalizowana, ponieważ każda kolejna taka iteracja to możliwość uruchomienia całego algorytmu jeszcze raz z nowym miastem początkowym więc najlepiej maksymalizować daną zmienną w celu jak największej ilości ekstremum lokalnych aby spośród jak największej puli można było wybrać najlepsze rozwiązanie.

## Poszukiwanie najlepszego wyniku po optymalizacji parametrów

Wykonamy jak najwięcej iteracji algorytmu, skupiając się już tylko na zoptymalizowanych wcześniej parametrów.

### Najlepszy wynik osiągnięty dla 50\_20:

Najlepszym wynikiem okazał się wynik z optymlizacjii wyników:

**Najlepszy wynik** : 3760 Najlepsza trasa : [24, 4, 28, 45, 25, 8, 46, 10, 11, 2, 31, 23, 16, 27, 49, 5, 15, 9, 39, 17, 30, 20, 32, 33, 29, 43, 37, 44, 1, 21, 3, 22, 50, 34, 48, 47, 19, 6, 36, 42, 26, 38, 14, 35, 41, 13, 12, 7, 18, 40]

Najlepszy rezultat dały parametry:

* Sąsiedztwo: **INSERT**
* iterację wewnątrz algorytmu: 1**00**
* start distacne: **5200**

### Najlepszy wynik osiągnięty dla 100\_10:

Najlepszy rezultat dały parametry:

* Sąsiedztwo: **INSERT**
* iterację wewnątrz algorytmu: **300**
* start distacne: **6400**

**Wynik: 5524**

Najlepsza trasa : [79, 83, 65, 75, 56, 97, 87, 44, 4, 90, 27, 23, 16, 74, 91, 36, 73, 68, 81, 86, 9, 66, 10, 25, 18, 85, 42, 1, 57, 22, 2, 49, 39, 98, 33, 100, 76, 99, 58, 63, 48, 41, 24, 55, 19, 46, 14, 35, 28, 72, 7, 21, 61, 6, 8, 88, 38, 54, 15, 92, 5, 45, 89, 96, 51, 71, 60, 50, 70, 67, 20, 59, 47, 37, 32, 94, 93, 30, 53, 17, 80, 84, 62, 78, 64, 95, 34, 12, 11, 31, 40, 77, 3, 69, 13, 43, 52, 29, 82, 26]

### Najlepszy wynik osiągnięty dla 200\_10:

Najlepszy wynik : **10567**

Najlepsza trasa : [22, 97, 184, 144, 1, 150, 135, 146, 173, 179, 183, 71, 65, 143, 28, 84, 68, 77, 27, 16, 166, 171, 185, 21, 96, 81, 20, 117, 89, 110, 83, 200, 145, 159, 181, 163, 94, 141, 33, 193, 57, 56, 35, 24, 131, 70, 5, 174, 90, 25, 76, 45, 100, 198, 113, 72, 48, 191, 139, 156, 88, 103, 133, 59, 34, 112, 151, 168, 175, 7, 50, 162, 137, 52, 31, 102, 119, 37, 114, 93, 95, 130, 170, 32, 74, 158, 73, 40, 54, 8, 58, 15, 155, 178, 66, 176, 91, 29, 172, 43, 126, 111, 42, 80, 125, 3, 192, 46, 19, 23, 61, 6, 148, 116, 165, 11, 49, 164, 86, 147, 17, 194, 140, 36, 169, 92, 47, 167, 197, 9, 128, 154, 4, 195, 152, 138, 78, 160, 122, 99, 2, 104, 62, 101, 161, 12, 153, 123, 149, 132, 98, 199, 124, 85, 63, 51, 41, 188, 60, 182, 107, 38, 157, 134, 187, 108, 13, 120, 87, 180, 109, 67, 189, 105, 55, 44, 82, 190, 196, 127, 106, 79, 30, 115, 177, 26, 14, 142, 118, 136, 129, 75, 69, 53, 121, 18, 39, 186, 10, 64]

Zwiększanie iteracji algorytmów nie przyniosło poprawy rezultatu w żadne z 3 wersji danych, wyniki były bardzo zbliżone do najlepszych wyników lecz nie okazały się lepsze.

# Algorytm NEH

## Wstęp teoretyczny:

Algorytm NEH służy do rozwiazywania problemu harmonogramowania zadań. Za zadanie ma znalezienie optymalnego harmonogramu dla danego zestawu danych. Jest algorytmem deterministycznym, daje on zawsze taki sam wynik dla tego samego zestawu danych. Z uwagi na sposób działania algorytmu nie zostały zastosowane żadne dodatkowe parametry.

## Działanie algorytmu:

1. Sortowanie zadań względem sumy wartości operacji(od największej do najmniejszej)
2. Sprawdzamy które ułożenie pierwszych dwóch zadań da nam krótszy czas. (1,2 czy 2,1)
3. Wybieramy ułożenie z krótszym czasem i dodajemy do niego kolejne zadanie w każdym z możliwych miejsc(dla 2,1 mamy ułożenie 3,2,1 ; 2,3,1 ; 2,1,3)
4. Wybieramy ponownie najlepsze ułożenie.
5. Kroki 3 i 4 powtarzamy tyle razy ile mamy zadań.
6. Wybieramy najlepsze rozwiązanie i liczymy dla niego czas.

## Wyniki otrzymane dla algorytmu NEH:

|  |  |  |  |
| --- | --- | --- | --- |
| Rozmiar danych | 50x20 | 100x10 | 200x10 |
| Wynik | 3890 | 5669 | 10751 |
| Solver | 3892 | 5719 | 11095 |

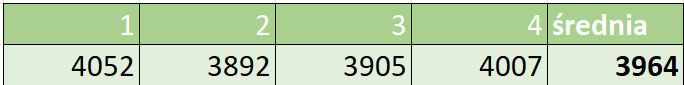
Wyniki otrzymane w skutek działania algoytmu neh w każdym zestawie danych osiągały lepsze rezultaty od metody solvera. Przy najmniejszym zestawie danych różnica jest minimalna jednak przy większych zestawach różnica się powiększa.

# EXCEL SOLVER

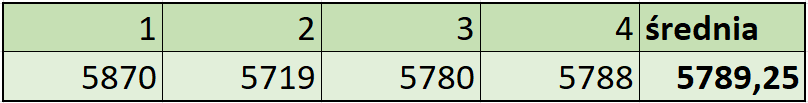
Dla każdego zestawu danych wygenerowano 4 różne rozwiązania za pomocą dodatku Solver w Excel:

Do każdego rozwiązania zostało na nowo wylosowane rozwiązanie początkowe.

### Zestaw Danych PFSP\_50\_20.xlsx



### Zestaw Danych PFSP\_100\_10.xlsx



### Zestaw Danych PFSP\_200\_10.xlsx

