



**AKADEMIA GÓRNICZO-HUTNICZA  
IM. STANISŁAWA STASZICA  
W KRAKOWIE**

# **Zagadnienie planowania trasy dla dostawcy ciepłego jedzenia**

**Wykonali:**

**Mateusz Łęcznar**

**Tristan Malatyński**

**Patryk Łuczak**

# 1.Opis analizowanego zadania

Zadanie polega na wyszukaniu najlepszej drogi dostawcy przyjmującego i dostarczającego jedzenie z różnych restauracji zapisanych w liście wraz z punktami dostawy.

Problem poddawany analizie należy do dziedziny optymalizacji matematycznej. Zadanie polega na znalezieniu optymalnej trasy dla dostawcy jedzenia, który przyjmuje i dostarcza zamówienia z różnych restauracji w danym regionie jednocześnie. Dana jest lista restauracji oraz punktów do których należy dostarczyć jedzenie.

W zadaniu poszukiwany jest wektor odwiedzanych punktów w macierzy czasu dojazdu, o minimalnej wartości sumy kar za czas realizacji zamówień. Plecak kierowcy ma określoną pojemność, tak więc konieczny jest optymalny wybór odwiedzanych punktów aby nie przekroczyć tego ograniczenia.

## Przyjęte uproszczenia

- Wszystkie zamówienia ważą tyle samo, pozostała pojemność plecaka określona jest jako ilość zamówień, które jeszcze można odebrać
- Lista punktów musi być znana przed rozpoczęciem pracy, nie będzie możliwości dynamicznej zmiany listy punktów(oczywiście można program odpalić jeszcze raz z nową listą punktów
- Czas załadunku/rozładunku nie ma znaczenia
- Plecak dobrze zatrzymuje ciepło więc kolejność dostawy odebranych już zamówień jest nieistotna
- W każdej restauracji może być maksymalnie jedno zamówienie w danym momencie
- Odległości pomiędzy punktami będą liczone w linii prostej

## Ograniczenia

- Pojemność plecaka nie może być przekroczona, wynosi ona 3 zamówienia
- Jeśli czas dojazdu od restauracji do zamawiającego będzie większy niż 40 minut zostanie nałożona ujemna nagroda
- Liczba zamówień do obsłużenia musi być ograniczona, ze względu na czas obliczeń (obliczanie drogi do powiedzmy 50 klientów i 20 restauracji zajmie względnie dużo czasu), a w praktyce i tak dostawca ma ograniczony czas pracy

## 2. Model Matematyczny

Pojedyncze rozwiązanie reprezentuje kolejność odwiedzanych węzłów, restauracje i adresy klientów są reprezentowane przez wierzchołki grafu, krawędzie reprezentują czas potrzebny do przemieszczenia pomiędzy węzłami. Czas do zamawiającego produkt którego jeszcze nie odebrano z restauracji wynosi nieskończoność (nie można do tego wierzchołka się udać)

Funkcja celu -> maksymalizacja zysku za czas realizacji zamówienia

Stan -> wierzchołek grafu

Decyzja -> wybór następnego wierzchołka

$$f(x) = \sum_{i=1}^n (P(\sum_{i=1}^k M(j-1, j)), k < 2n - 1$$

**n** - ilość zamówień=ilość punktów odbioru=ilość doręczeń

**k** - ilość odwiedzonych wierzchołków w drodze od restauracji do zamawiającego z pojedynczym zamówieniem

**x** - kolejność odwiedzania wierzchołków

**M(j-1,j)** - koszt przejazdu z wierzchołka j-1 do wierzchołka j, wyliczany z macierzy kosztów

**P(a)** - wartość kary/nagrody dla czasu realizacji

Macierz kosztów ma postać:

Gdzie wielkie litery symbolizują restauracje, a ich małe odpowiedniki to adresy zamawiających.

From \ To	A	a	B	b	C	c	...
A	$\infty$						
a	$\infty$	$\infty$					
B			$\infty$				
b			$\infty$	$\infty$			
C					$\infty$		
c					$\infty$	$\infty$	
...							

Sąsiedztwo definiujemy jako możliwe scenariusze ruchu z wierzchołka. Zgodnie z parametrami algorytmu wyliczamy zakres przeszukiwanych wierzchołków. Następnie sprawdzamy na podstawie macierzy kosztów, jaka będzie wartość nagrody (patrz tabela nagród) dla sprawdzanego rozwiązania.

Następnie zgodnie z algorytmem, sprawdzamy czy funkcja celu - wielkość nagrody jest większa od obecnej, a jeśli nie to sprawdzamy prawdopodobieństwo przyjęcia rozwiązania (dzielimy wartość poprzednią przez nową, dzięki czemu wiemy jak mocno rozwiązanie się pogorszy), a następnie na podstawie wylosowanej liczby przyjmujemy lub odrzucamy rozwiązanie i przechodzimy do kolejnego etapu.

Tablica nagród:

Tablica działa jako różnica pomiędzy bezpośrednim czasem dostawy (np. z A do a) a aktualnym czasem potrzebnym do realizacji zamówienia. Od czasu dostawy zależy wielkość napiwku. Szybkie dostarczenie gwarantuje otrzymanie napiwku, a spóźnienie powyżej 40 minut powoduje nałożenie ujemnej nagrody.

Czas realizacji zamówienia	ujemna nagroda /nagroda
(0,5)	+40 zł
[5, 8)	+35 zł
[8, 10)	+30 zł
[10,15)	+25 zł
[15,20)	+20 zł
[20,25)	+15 zł
[25,30)	+10 zł
[30,40)	+5zł
[40,+ $\infty$ )	-50zł

# 3.Algorytm

## Opis algorytmu

Do optymalizacji omawianego zagadnienia wykorzystano zaproponowany przez prowadzącego algorytm tabu search. Poniższy pseudokod jest oczywiście zarysem funkcjonalności aplikacji. Dodatkowo oprócz wymienionej funkcji należało zaimplementować funkcje służące do sprawdzenia poprawności rozwiązań, generowania rozwiązania początkowego, generowania rozwiązań pokrewnych, obliczania czasu pracy, obliczania funkcji celu, obliczania nagrody dla danego zamówienia oraz generowania wykresów przebiegu funkcji celu.

## Pseudokod

- max backpack = 3
- lengths tabu
- random solutions
- max\_iterations

// Finding the best solution from all parameters *lengths tabu*

**for** *length\_tabu* **in** lengths\_tabu:

*finish\_road*, *best\_salary\_temp* = **next\_solution\_by\_iter**():

create all necessary elements : cost matrix, possible\_swap, tabu\_table

**for** *i* **in** max\_iterations:

//Finding the best of all possible swap

**for** swap **in** possible\_swap:

calculate\_best\_swap

// comparison with the best solution so far

**if** current\_solution **is** better:

update\_solutions

update\_tabu\_table

update\_possible\_swap

**else:**

update\_tabu\_table

update\_possible\_swap

**return** best\_solution

// comparison with the best solution so far from other parameters

**if** best\_salary < *best\_salary\_temp*

update\_best\_salary\_solution

**return** best salary, best solution

**Przykładowa macierz kosztów ma postać:**

From \ To	A	a	B	b	C	c	...
A	$\infty$	23	30,5	20,25	5,575	32	...
a	$\infty$	$\infty$	11,775	14,6	19,2	9,95	...
B	30,5	11,775	$\infty$	13,125	28,275	6,025	...
b	20,25	14,6	$\infty$	$\infty$	20,1	17,6	...
C	5,575	19,2	28,275	20,1	$\infty$	28,8	...
c	32	9,95	6,025	17,6	$\infty$	$\infty$	...
...	...	...	...	...	...	...	...

Została uzyskana przez losowo wybrane punkty na mapie. Następnie zostały obliczone odległości wszystkich punktów do innych

Przykładowa trasa punktów odbioru i punktów dostawy.





## Krótki schemat algorytmu funkcji *best\_solution()*:

Lista możliwych podmian:

[('A', 'B'), ('A', 'C'), ('A', 'D'), ('A', 'E'), ('A', 'F'), ('A', 'G'), ('A', 'H'), ('A', 'I'), ('A', 'J')]  
[('B', 'C'), ('B', 'D'), ('B', 'E'), ('B', 'F'), ('B', 'G'), ('B', 'H'), ('B', 'I'), ('B', 'J')]  
[('C', 'D'), ('C', 'E'), ('C', 'F'), ('C', 'G'), ('C', 'H'), ('C', 'I'), ('C', 'J')]  
[('D', 'E'), ('D', 'F'), ('D', 'G'), ('D', 'H'), ('D', 'I'), ('D', 'J')]  
[('E', 'F'), ('E', 'G'), ('E', 'H'), ('E', 'I'), ('E', 'J')]  
[('F', 'G'), ('F', 'H'), ('F', 'I'), ('F', 'J')]  
[('G', 'H'), ('G', 'I'), ('G', 'J')]  
[('H', 'I'), ('H', 'J')]  
[('I', 'J')]

Przed

A -> J -> a -> j -> G -> g -> B -> I -> C -> b -> i -> E -> D -> e -> F -> f -> H -> c -> h -> d

Po podmianie

B -> J -> b -> j -> G -> g -> A -> I -> C -> a -> i -> E -> D -> e -> F -> f -> H -> c -> h -> d

Podmiany będą wykonywane na oryginalnej drodze za każdym razem aż do znalezienia najlepszej.

Po znalezieniu najlepszej podmiany zostanie ona wyrzucona z możliwych podmian a nowa znaleziona droga stanie się tą oryginalną.

Zakładamy że najlepszą podmianą było A i B

Lista możliwych podmian teraz:

[('A', 'C'), ('A', 'D'), ('A', 'E'), ('A', 'F'), ('A', 'G'), ('A', 'H'), ('A', 'I'), ('A', 'J')]  
[('B', 'C'), ('B', 'D'), ('B', 'E'), ('B', 'F'), ('B', 'G'), ('B', 'H'), ('B', 'I'), ('B', 'J')]  
[('C', 'D'), ('C', 'E'), ('C', 'F'), ('C', 'G'), ('C', 'H'), ('C', 'I'), ('C', 'J')]  
[('D', 'E'), ('D', 'F'), ('D', 'G'), ('D', 'H'), ('D', 'I'), ('D', 'J')]  
[('E', 'F'), ('E', 'G'), ('E', 'H'), ('E', 'I'), ('E', 'J')]  
[('F', 'G'), ('F', 'H'), ('F', 'I'), ('F', 'J')]  
[('G', 'H'), ('G', 'I'), ('G', 'J')]  
[('H', 'I'), ('H', 'J')]  
[('I', 'J')]

Przed

B -> J -> b -> j -> G -> g -> A -> I -> C -> a -> i -> E -> D -> e -> F -> f -> H -> c -> h -> d

Po

B -> J -> b -> j -> G -> g -> C -> I -> A -> c -> i -> E -> D -> e -> F -> f -> H -> a -> h -> d

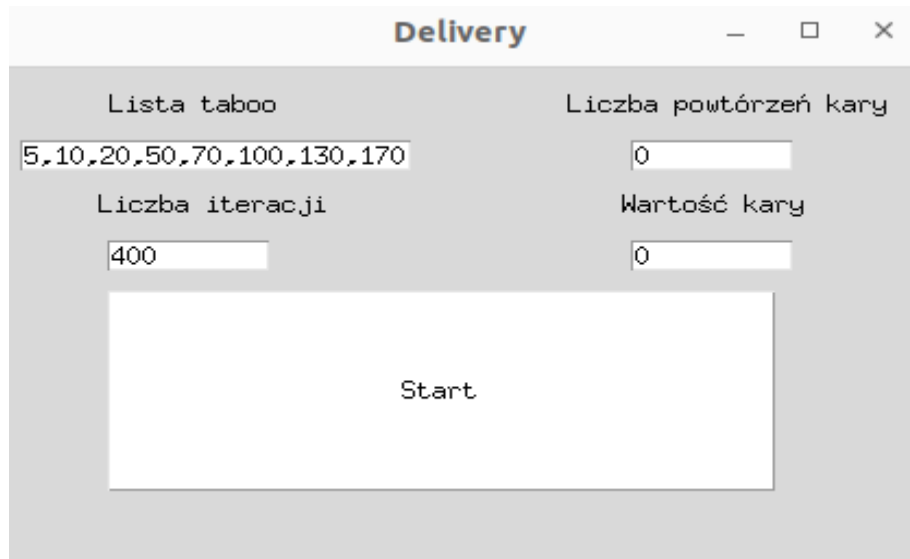
Przykład podany jest dla podmian (A,B) ale w algorytmie zastosowano podmiany **(A,B),(A,b),(a,b)**

Maksymalna ilość podmian to mniej niż  $3 * (N * (N-1) / 2)$  gdzie n to liczba restauracji. Definiuje to również złożoność algorytmu.

## 4.Aplikacja

Program losuje losowo miejsca dostawy, można wybrać je manualne modyfikując lekko program, opisane jest to w pliku readme programu. Wpisujemy po przecinku interesujące nas parametry i klikamy start.

Wynik ostateczny i wyniki otrzymane z kolejnych etapów programu będą widoczne w konsoli w formie opisanego tekstu.



Generowanie przykładowego rozwiązania:

A -> M -> E -> e -> D -> a -> d -> L -> H -> m -> l -> K -> k -> h -> l -> N -> C -> n -> i -> B -> F -> f -> b -> c -> J -> j -> G -> g

Nagroda przed podmianą: -70 PLN

NAJLEPSZE ROZWIĄZANIE UZYSKANO PO 9 ITERACJACH DLA DŁUGOŚCI TABLICY tabu 5 UZYSKANA NAGRODA: 240

l -> i -> B -> b -> D -> K -> d -> E -> H -> k -> e -> h -> A -> a -> M -> N -> n -> C -> c -> L -> F -> f -> l -> m -> J -> j -> G -> g

NAJLEPSZE ROZWIĄZANIE UZYSKANO PO 9 ITERACJACH DLA DŁUGOŚCI TABLICY tabu 10 UZYSKANA NAGRODA: 240

l -> i -> B -> b -> D -> K -> d -> E -> H -> k -> e -> h -> A -> a -> M -> N -> n -> C -> c -> L -> F -> f -> l -> m -> J -> j -> G -> g

.

.

.

.

NAJLEPSZE ROZWIĄZANIE UZYSKANO PO 85 ITERACJACH DLA DŁUGOŚCI TABLICY tabu 170 UZYSKANA NAGRODA: 300

M -> m -> l -> i -> K -> D -> k -> d -> J -> j -> B -> b -> L -> l -> E -> F -> f -> H -> e -> h -> A -> a -> C -> c -> N -> n -> G -> g

Ostateczne najlepsze rozwiązanie:

L -> l -> M -> m -> A -> a -> C -> c -> E -> e -> J -> j -> D -> d -> N -> n -> F -> f -> G -> g -> B -> b -> H -> h -> K -> k -> l -> i

Po znalezieniu najlepszego rozwiązania możemy oglądać wizualizacje drogi którą musiałby przebyć nasz dostawca.

Program generuje również wykresy poszukiwania najlepszego rozwiązania.

## 5. Testy

Testy zostały wykonane w pierwszej fazie tworzenia algorytmu gdzie w założeniach była podmiana jedynie **(A,B)** czyli restauracji.

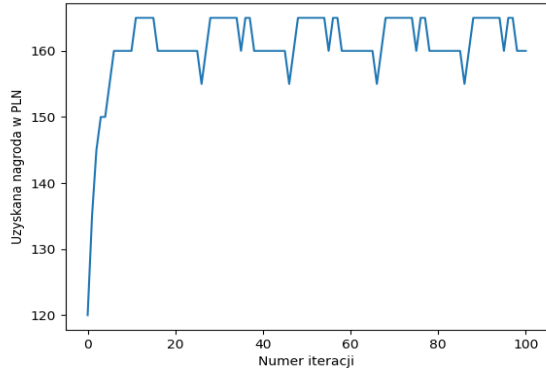
Następnie po dołożeniu dodatkowych funkcjonalności algorytmu, czyli możliwości podmiany “wszystkich z wszystkimi” wyniki jakie uzyskaliśmy były znacznie lepsze oraz wykres funkcji celu stawał się jednolity.

Algorytm był testowany kilkakrotnie dla długości tablicy tabu **[5,10,20,50,70,100,130,170]**, który zawierał te same dane.

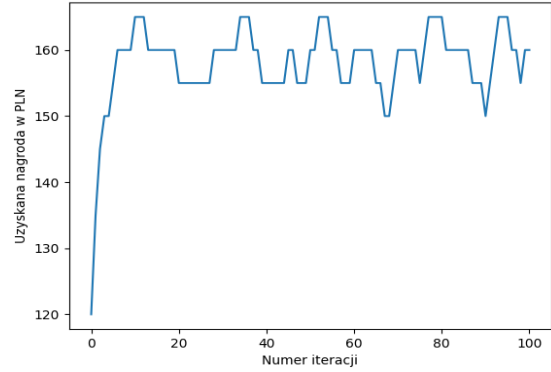
Wyjątek był dla algorytmu z podmianą podwójną bo tam możliwości było mniej więc i lista tabo musiała być mniejsza. Testy były przeprowadzane na 14 restauracjach i 14 odbiorcach.

## Przeszukiwanie z podmianami tylko (A,B)

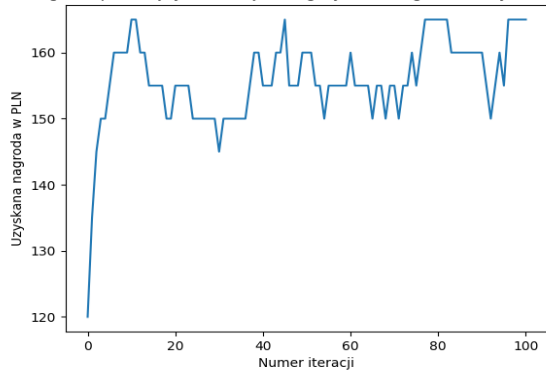
Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:5



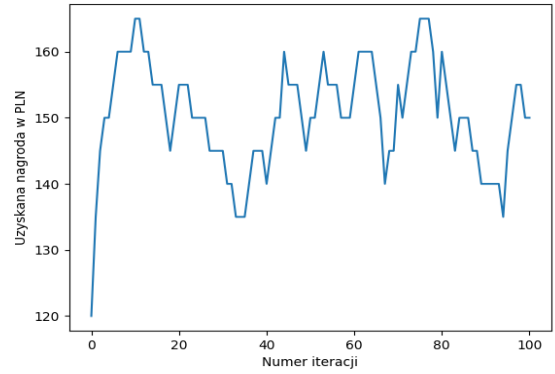
Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:10



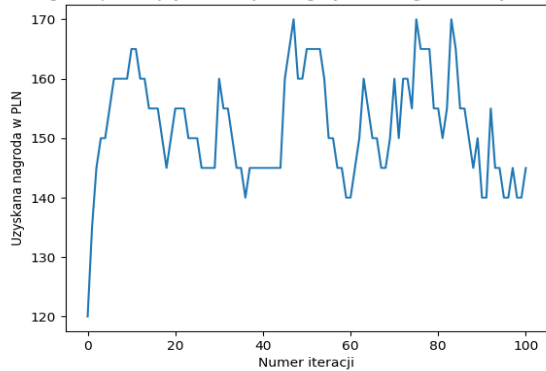
Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:15



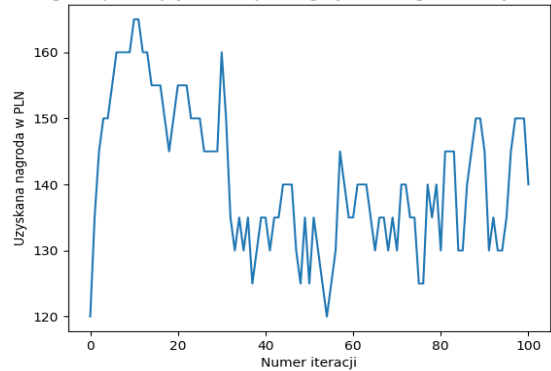
Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:23



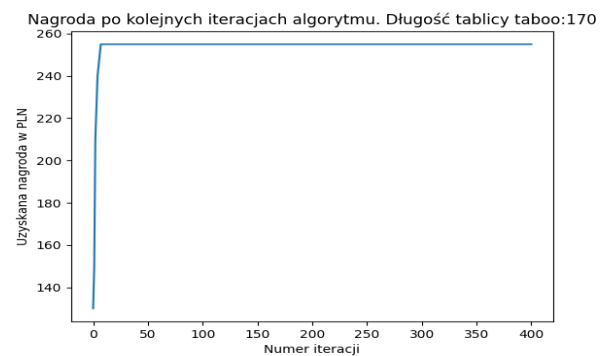
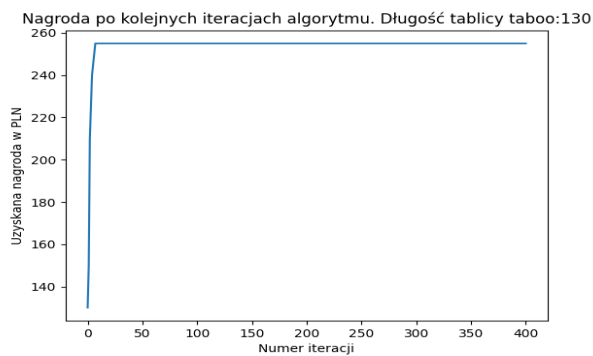
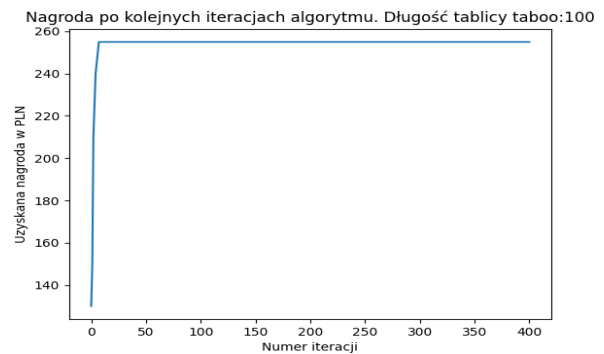
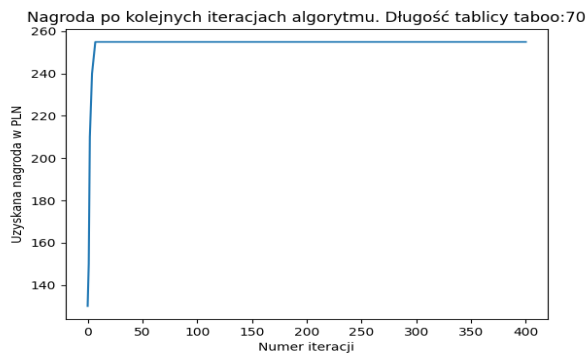
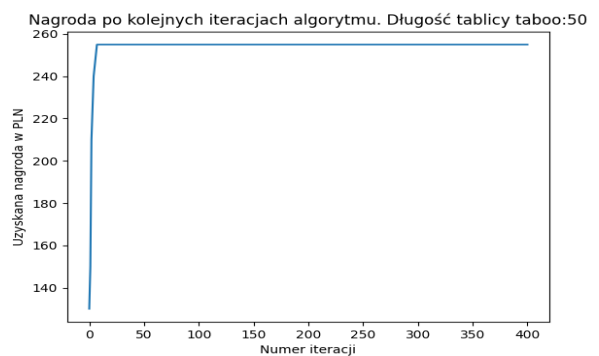
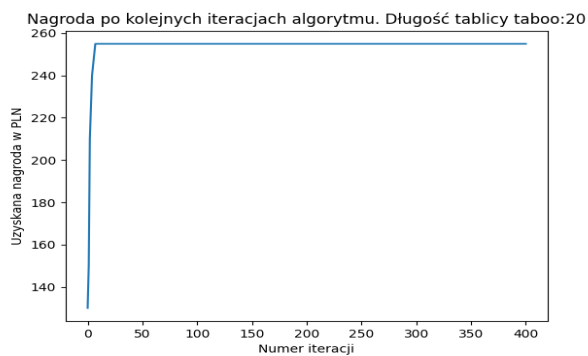
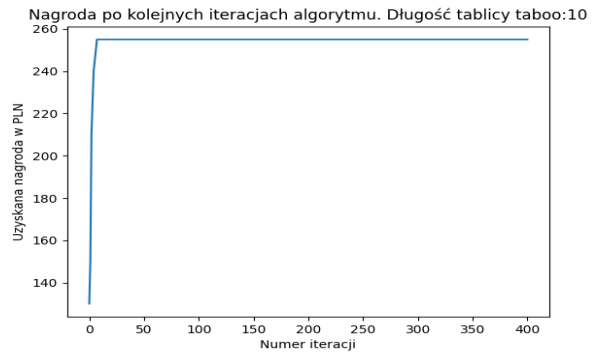
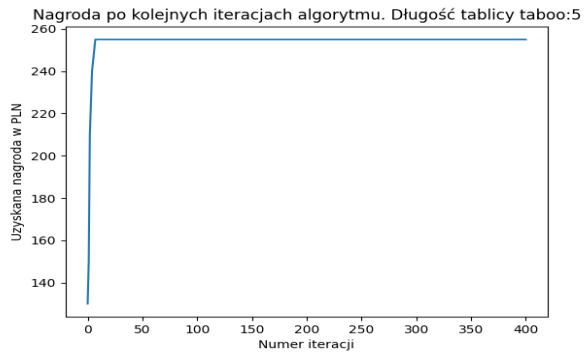
Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:30



Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:40

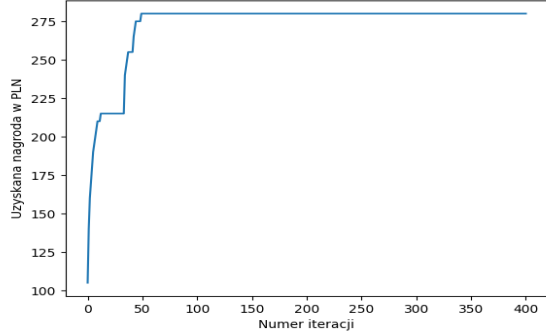


## Zastosowanie kary -30 pln jeśli znaleziona najlepsza podmiana będzie użyta już 10 raz. Jeśli po tej karze nie będzie wynik lepszy zostaje poprzednie najlepsze

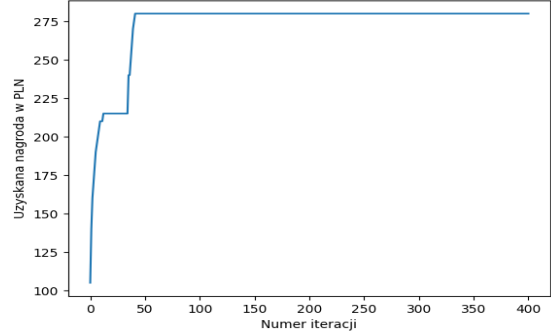


# Wyniki bez kary z wszystkimi możliwymi podmianami

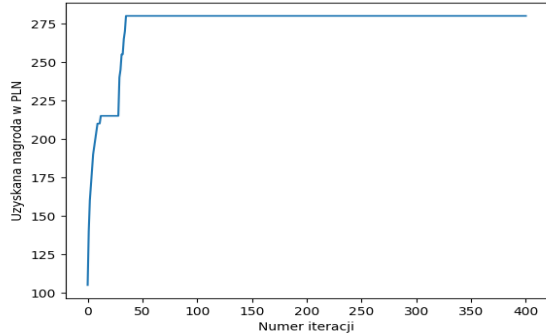
Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:5



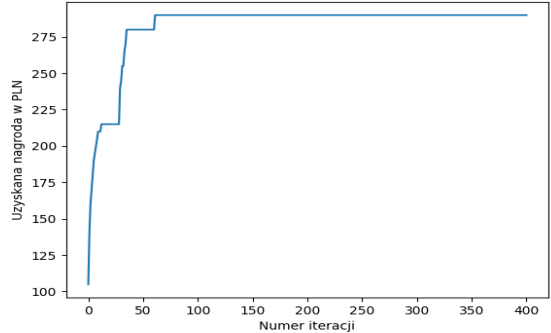
Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:10



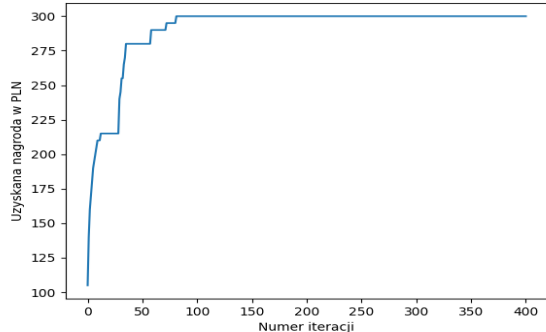
Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:20



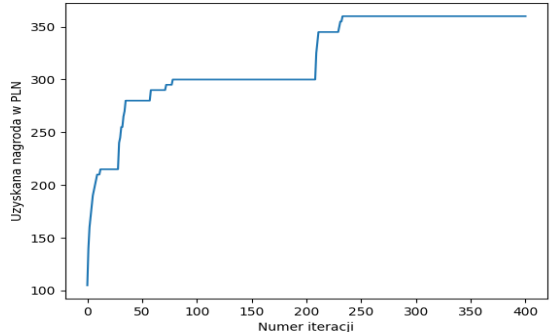
Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:50



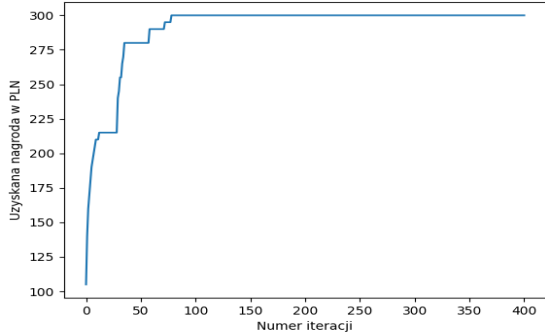
Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:70



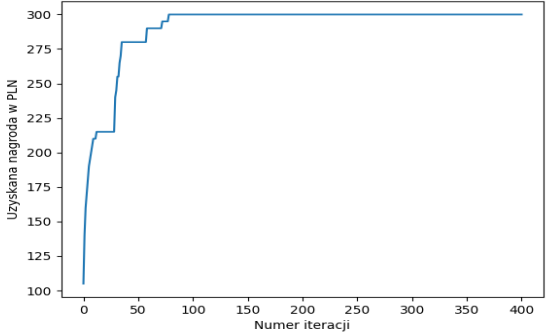
Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:100



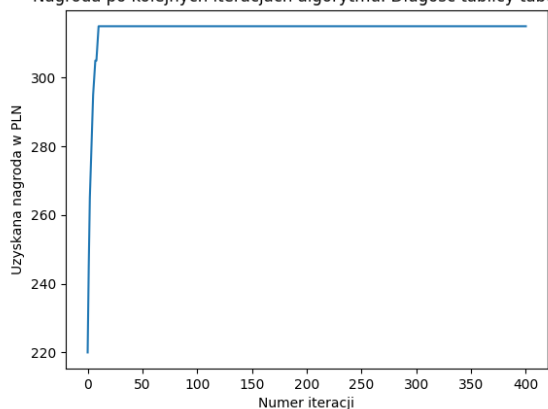
Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:130



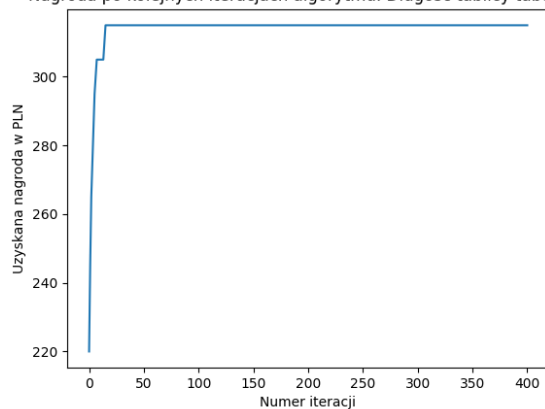
Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:170



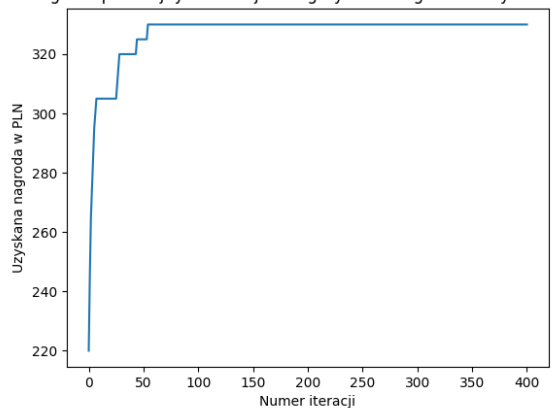
Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:5



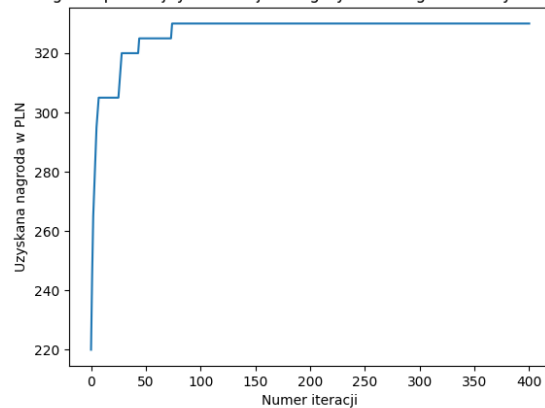
Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:10



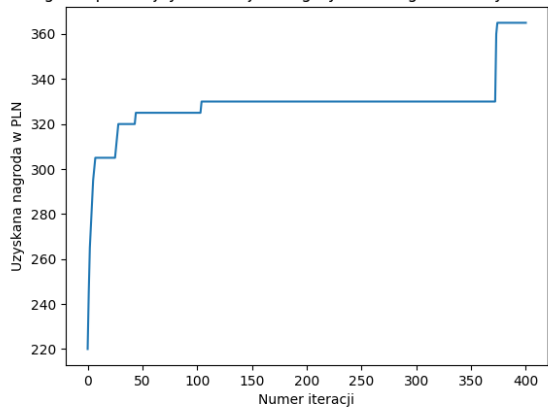
Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:50



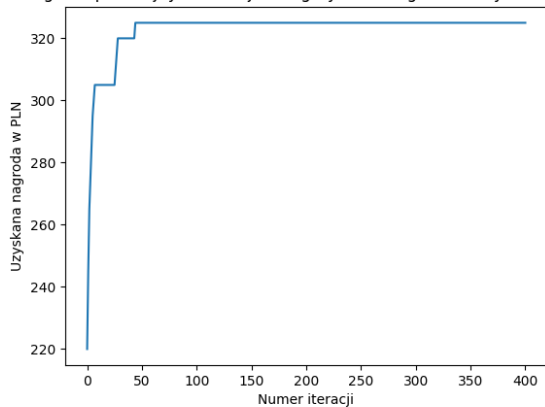
Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:70



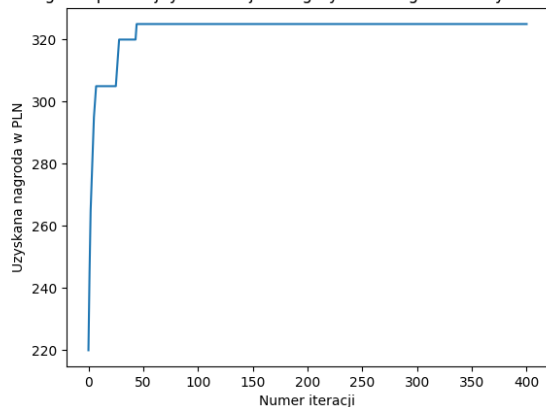
Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:100

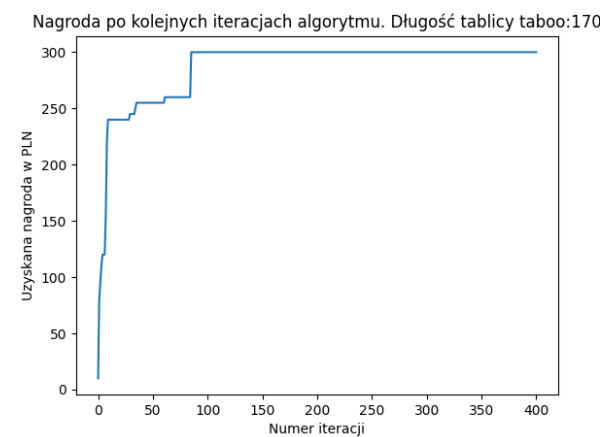
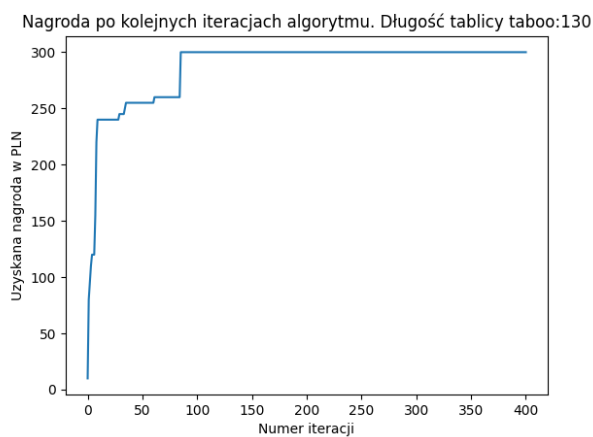
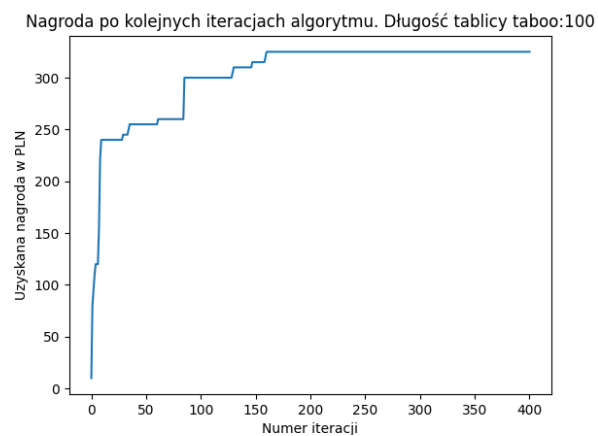
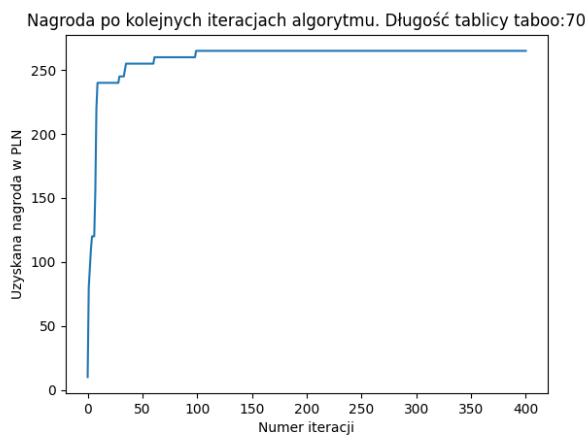
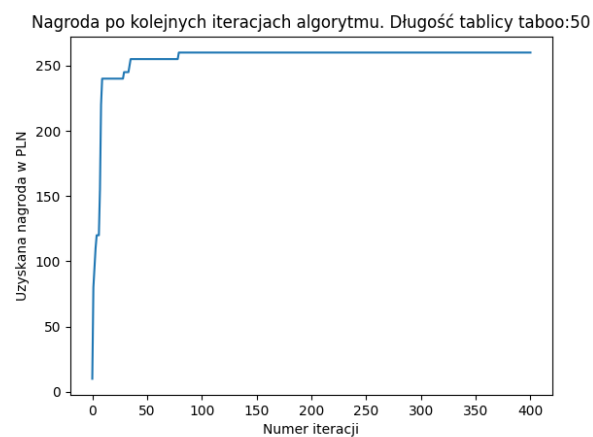
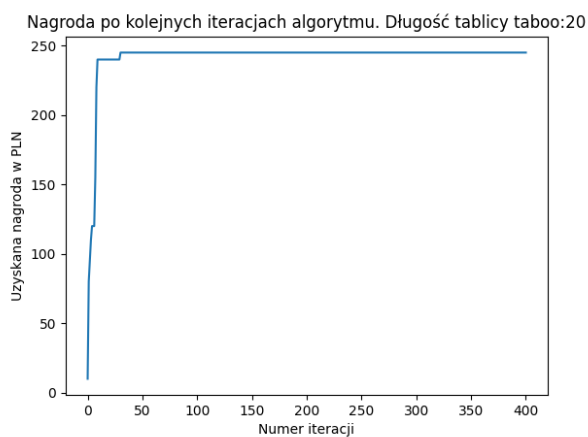
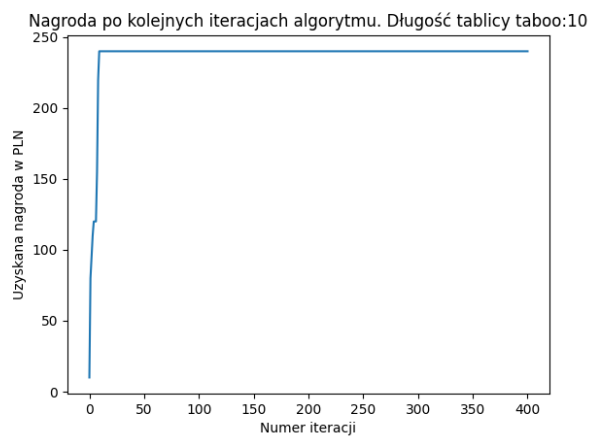
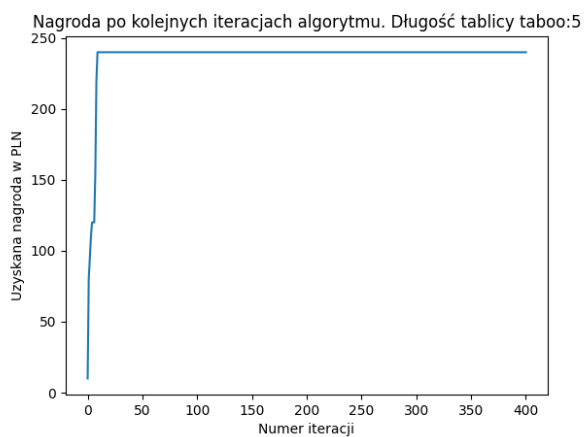


Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:130



Nagroda po kolejnych iteracjach algorytmu. Długość tablicy taboo:170

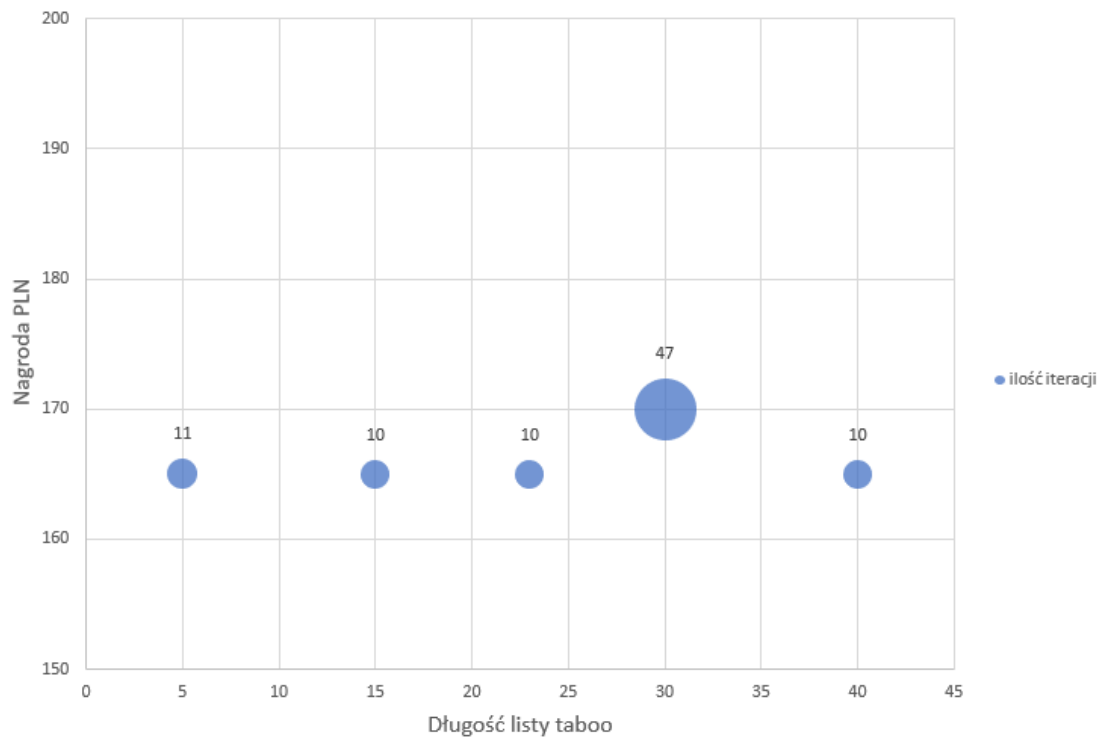




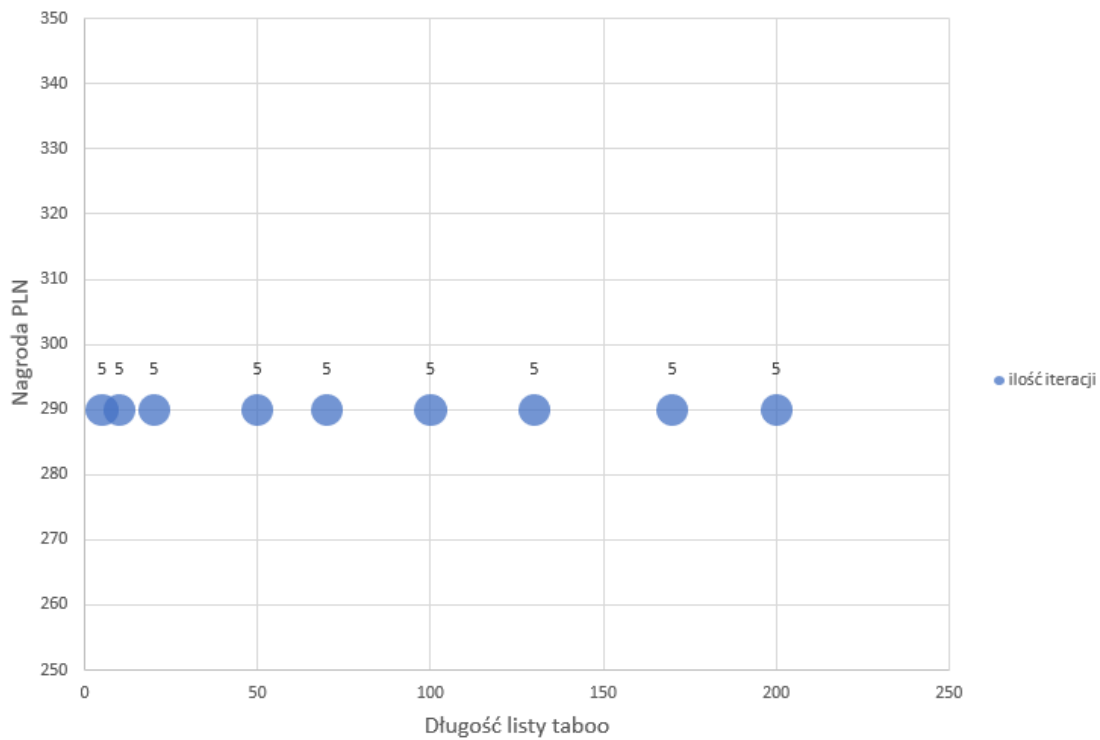


## Zbiorne porównanie wyników

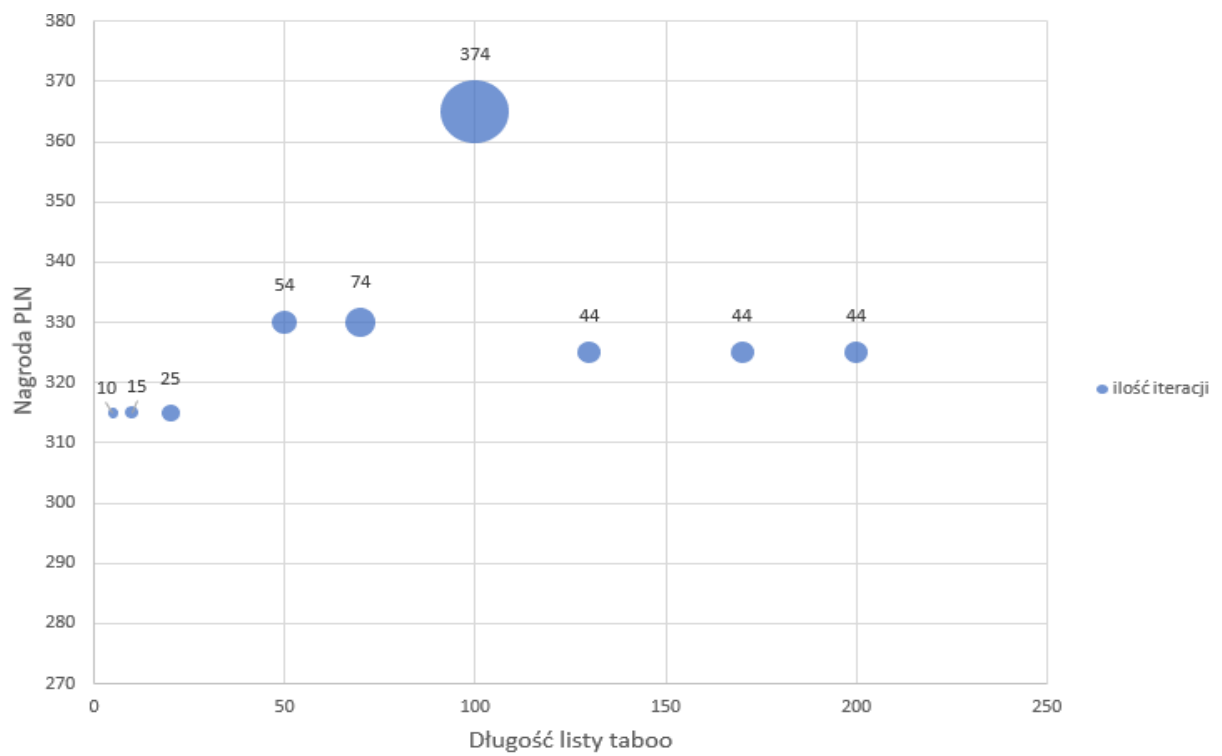
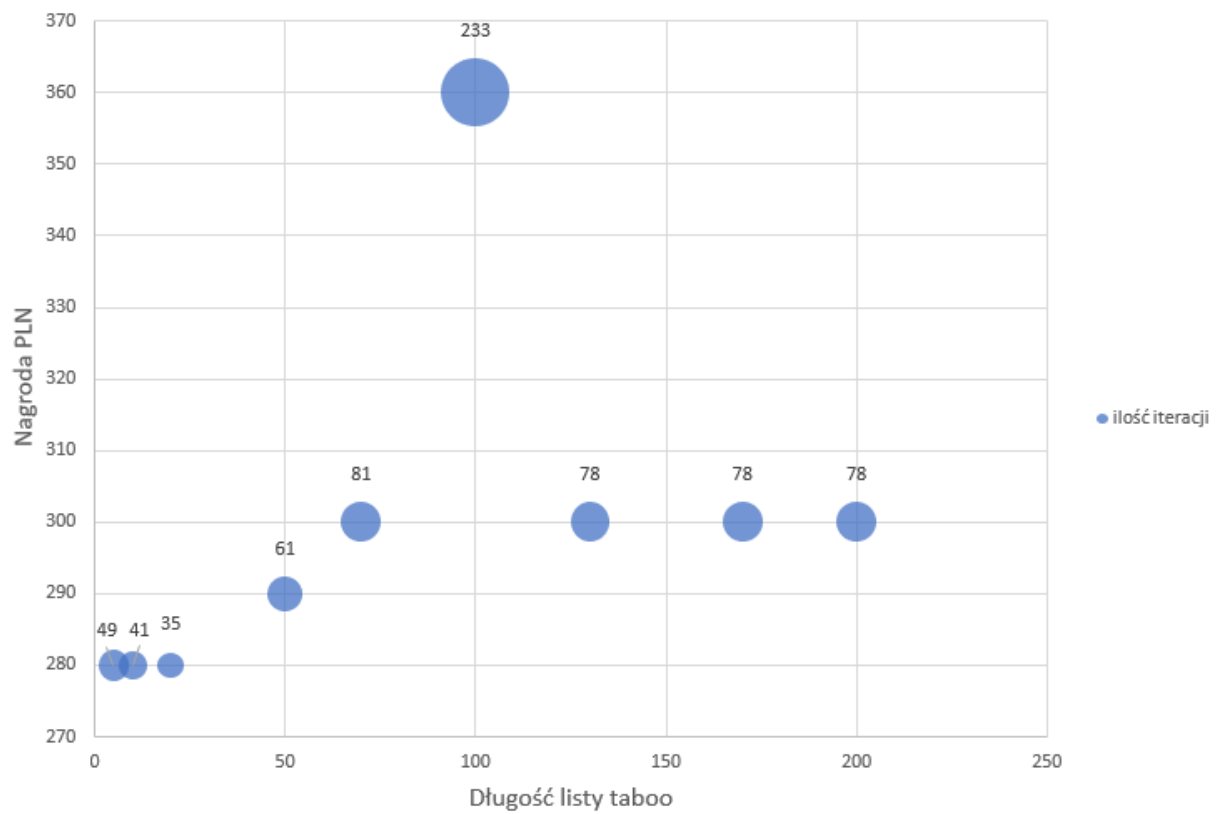
### Z podmianą tylko (A,B)

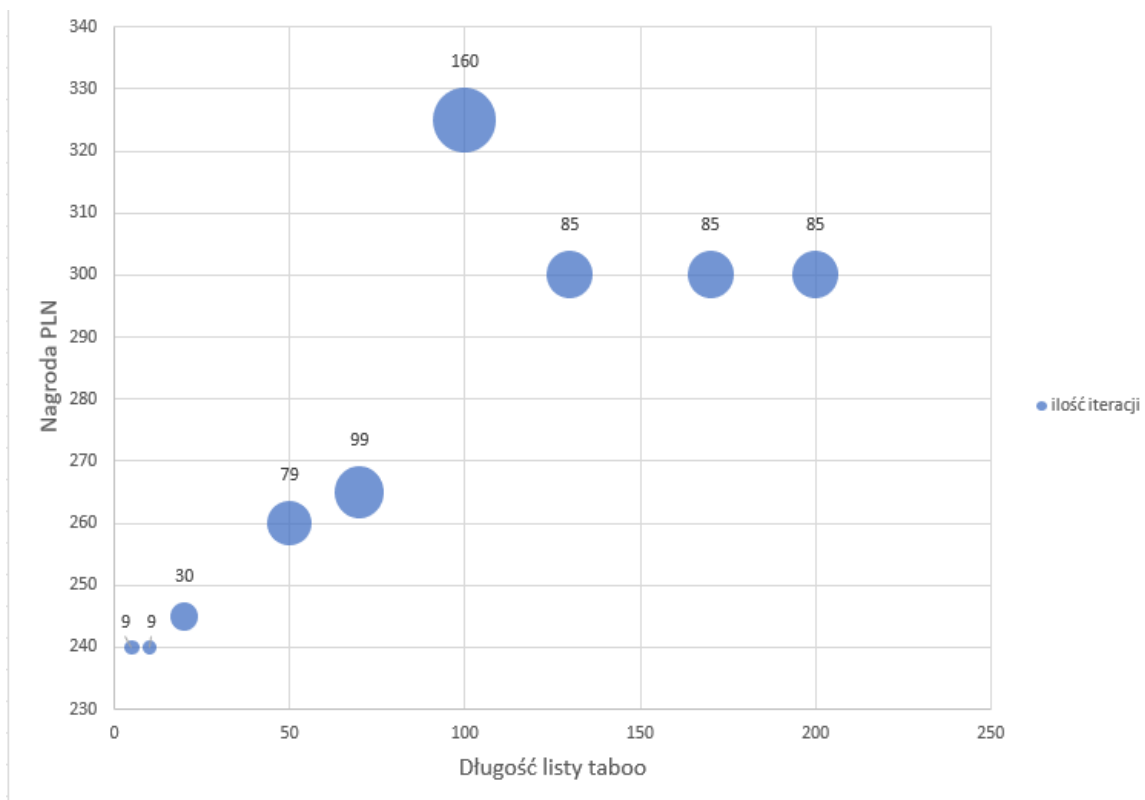


### Z karą -30 PLN po przekroczeniu 10 podmian



## Kolejne testy z wszystkimi możliwymi podmianami





## 6.Podsumowanie

Cały projekt okazał się dość ciekawym zadaniem, od samego początku wymagał on sporo planowania i połączenia odpowiednich elementów w jedną całość. Mogliśmy popracować z chociaż mocno przybliżonym realnym problemem.

Ważnym momentem podczas pracy nad algorytmem było dodatnie maksymalnej liczby podmian przez co algorytm zaczął pracować wyśmienicie.

Niestety zaimplementowana funkcja kary nie poprawiała algorytmu, a go psuła.

Algorytm uzyskiwał najlepsze wyniki dla długości tablicy tabu w okolicach połowy maksymalnej liczby podmian.

### Kierunki dalszego rozwoju:

Do dalszego rozwoju programu mogłoby się przyczynić kryterium aspiracji. Niewątpliwie przydałoby się zoptymalizować przeszukiwanie i sprawdzanie poprawności rozwiązania ponieważ generuje to wiele dodatkowych iteracji.

Dobrym pomysłem byłoby łatwiejsze dodawanie lub odejmowanie miejsc odbioru i dostawy ponieważ w początkowej fazie implementacji nie było to brane pod uwagę.

	Procentowy wkład pracy [%]		
Etap	Mateusz Łęcznar	Tristan Malatyński	Patryk Łuczak
Model matematyczny i opis problemu	30	70	0
Założenia modelu	20	30	50
Algorytm	70	20	10
Testy	60	30	10
Dokumentacja	40	30	30