



ALGORYTM PAKOWANIA CIĘŻARÓWEK

Streszczenie

Adaptacja algorytmu genetycznego do problemu optymalnego pakowania ciężarówek oraz
testy i wynik na przykładowych zestawach danych

Wojciech Długosz, Nicolas Duc, Kamil Bardziej

Spis treści

I. Dokumentacja algorytmu

1. Zagadnienie	2
1.1. Problem:	2
1.2. Ograniczenia:	2
1.3. Uogólnienia:	3
2. Model matematyczny	3
2.1. Dane stałe dla firmy:	3
2.2. Dane dotyczące zlecenia:	3
2.3. Zmienne decyzyjne:	3
2.4. Funkcja celu:	4
2.5. Warunki:	4
3. Dobór algorytmu	5
4. Adaptacja algorytmu genetycznego do naszego problemu	5
4.1. Chromosom:	5
4.2. Ogólna idea algorytmu genetycznego	6
4.3. Adaptacja kroków algorytmu:	6
5. Dane i parametry pracy algorytmu	8
5.1. Dane wejściowe	8
5.2. Zestawy parametrów algorytmu	9

II. Testy oraz Wyniki

1. Wpływ zmiany pojedynczych parametrów na wyniki	10
1.1. Zmienna populacja	11
1.2. Zmiana prawdopodobieństwa mutacji	12
1.3. Zmiana prawdopodobieństwa krzyżowania	13
1.4. Zmiana ilości punktów krzyżowania	14
1.5. Ostatecznie dobrane parametry	14
2. Wyniki	15
2.1. Zestaw 1	15
2.2. Zestaw 2 (duży)	17
2.3. Zestaw 3 (instacja testowa)	19

III. Dokumentacja GUI

1. Cel	21
2. Elementy	21

I. Dokumentacja algorytmu

Wojciech Długosz, Nicolas Duc, Kamil Bardziej

1. Zagadnienie

1.1. Problem:

Nasz problem obejmuje zagadnienie logistyczne związane z optymalnym załadowaniem towarów, mając podaną dostępną flotę ciężarówek oraz oczekiwaną listę paczek adresowanych do poszczególnych magazynów, tak by zminimalizować koszty transportu, zależne od wagi ładunku oraz dystansu pomiędzy centralą a poszczególnymi magazynami.

Założyliśmy, że dysponujemy kilokoma typami ciężarówek (każda o indywidualnych cechach, tj. ładowność oraz spalanie), a każda z nich docelowo transportuje towar tylko do jednego z magazynów. Towarem są adresowane do konkretnego magazynu paczki kurierskie, o konkretnych wagach. Każdy hangar oddalony jest o inną odległość od centrali.

Algorytm ma za zadanie tak dobrać i załadować ciężarówki odpowiednimi paczkami, by każda trafiła pod odpowiedni adres hangaru, jednocześnie minimalizując całkowite koszty.

1.2. Ograniczenia:

- mamy 3 typy ciężarówek: każda ma inną ładowność i spalanie
- są różne typy paczek, mają różne wagi
- musimy zabrać wszystkie paczki
- nie można przekraczać ładowności ciężarówek
- spalanie rośnie wraz z ładunkiem
- waga całego transportu nie przekracza ładowności dostępnej floty pojazdów

- rozwozimy paczki z centrali do poszczególnych miast które oddalone są o różne odległości
- ciężarówka jedzie tylko do jednego magazynu

1.3. Uogólnienia:

- spalanie rośnie proporcjonalnie do ładunku
- nie bierzemy pod uwagę przerw dla kierowców
- koszty eksploatacji uśredniamy dla każdego rodzaju ciężarówek, nie rozpatrujemy osobno serwisów awarii zużycia etc.
- paczki będą z góry adresowane do konkretnego magazynu

2. Model matematyczny

2.1. Dane stałe dla firmy:

M – Ilość magazynów

C – ilość ciężarówek

k – koszt paliwa za litr

a_c – wektor kosztów eksploatacji dla ciężarówki c

l_c – wektor ładowności wagowej dla ciężarówki c

$s_{min\ c}$ – wektor minimalnego zużycia paliwa dla ciężarówki c

$s_{max\ c}$ – wektor maksymalnego zużycia paliwa dla ciężarówki c

2.2. Dane dotyczące zlecenia:

P – ilość paczek

d_m – wektor dystansów do magazynu m

w_p – wektor wag paczek p

2.3. Zmienne decyzyjne:

x_{cp} - macierz decydująca czy ciężarówka c zabiera paczkę p , (wartości 0 lub 1)

y_{cm} - macierz decydująca czy ciężarówka c jedzie do magazynu m , (wartości 0 lub 1)

Indeksy:

c – ciężarówki

p – paczki

m – magazyny

2.4. Funkcja celu:

Funkcja celu minimalizuje koszty poprzez optymalne załadowanie ciężarówek

$$\min \left(\sum_{m=1}^M \left(\sum_{c=1}^C a_c y_{cm} + \sum_{c=1}^C d_m k s_{\min c} y_{cm} + \sum_{c=1}^C d_m k \frac{\sum_{p=1}^P w_p x_{cp}}{l_c} (s_{\max c} - s_{\min c}) y_{cm} \right) \right)$$

2.5. Warunki:

– warunek określający że waga paczek nie przekracza sumy ładowności

$$\sum_{p=1}^P w_p \leq \sum_{c=1}^C l_c$$

– warunek określający nie przekraczanie ładowności

$$\sum_{p=1}^P w_p x_{cp} \leq l_c \quad \text{dla } c = 1, \dots, C$$

– warunek określający że każdą paczkę bierzemy 1 raz

$$\sum_{p=1}^P x_{cp} = 1 \quad \text{dla } c = 1, \dots, C$$

- warunek określający że każda ciężarówka jedzie tylko do jednego magazynu

$$\sum_{m=1}^M y_{cm} = 1 \quad \text{dla } c = 1, \dots, C$$

3. Dobór algorytmu

Początkowym wyborem naszego zespołu był algorytm mrówkowy. Podjęliśmy tę decyzję, sądząc, że ów algorytm, sprawdzający się często w problemach dotyczących logistyki i transportu, będzie pasował do naszego problemu i pozwoli nam wyznaczyć najlepsze optymalne rozwiązanie.

Z biegiem czasu doszliśmy do wniosku, że tę metodę wygodnie stosuje się do przeszukiwania przestrzennego, jak również do rozwiązania problemów harmonogramowania prac w systemie produkcji. Wiedząc, że nasz problem nie sprowadza się do wyznaczania tras naszych ciężarówek, a jedynie do organizacji zasad pracy naszej firmy transportowej, zdecydowaliśmy się wziąć mniej dokładny, lecz prostszy w implementacji algorytm genetyczny. Dodatkowo, algorytm mrówkowy, przy zbyt dużej liczbie danych, staje się znacząco nieefektywny, co skutkowałoby nałożeniem dodatkowych ograniczeń na nasze zadanie.

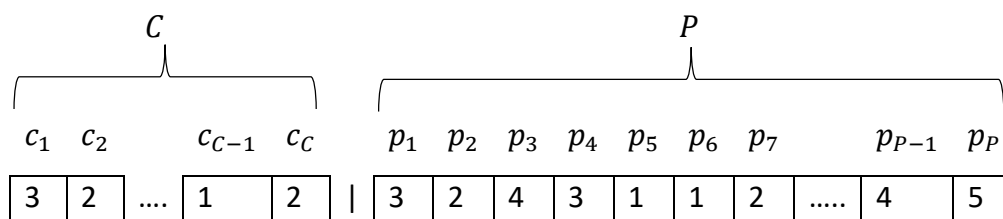
4. Adaptacja algorytmu genetycznego do naszego problemu

4.1. Chromosom:

Dla C ciężarówek i P paczek:

Pierwsze C miejsc w chromosomie reprezentuje ciężarówki, wartość w miejscu magazyn do którego jedzie

Ostatnie P miejsc (po C miejsc dla ciężarówek) reprezentują paczki, wartość w chromosomie reprezentuje ciężarówkę do której pakujemy



4.2. Ogólna idea algorytmu genetycznego

Inicjalizacja populacji:

Losujemy osobniki (rozwiązania)

Ocena osobników:

Wyliczamy wartości funkcji celu i
prawdopodobieństwa wyboru

Selekcja:

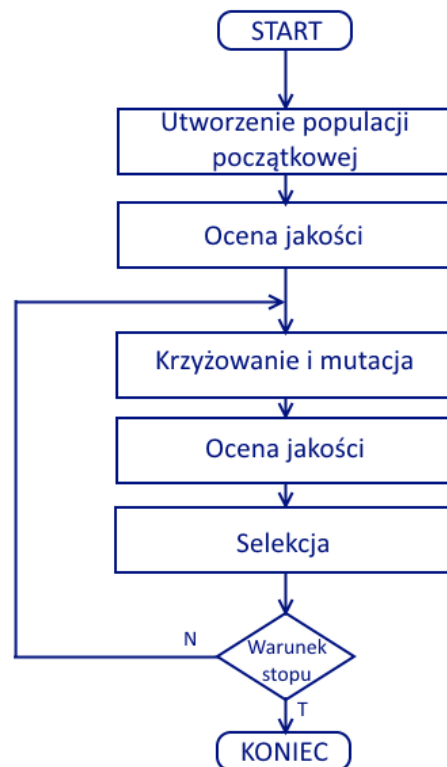
Wybieramy które rozwiązania krzyżować

Krzyżowanie:

Krzyżujemy wcześniej wybrane osobniki
według wybranej metody

Mutacja:

Losowa modyfikacja rozwiązania



Źródło: http://algorytmy.ency.pl/plik/algorytm_genetyczny_schemat_blokowy.png

4.3. Adaptacja kroków algorytmu:

a. Inicjalizacja populacji z użyciem heurystyki (funkcja `init_pop`):

Rozwiązania losowane są z użyciem heurystyki zaimplementowanej w funkcji `random_chromosome()`

Heurystyka zapewnia wstępne zapakowanie paczek do ciężarówek w losowy sposób w taki sposób że ładujemy ciężarówki paczkami aż będą one pełne i następnie bierzemy kolejną ciężarówkę.

Takie rozwiązanie zapewnia nam uniknięcie znacznej części rozwiązań niedopuszczalnych przy zwykłym losowaniu (np. przeładowania ciężarówek) a jednocześnie zapewnia nam dużą losowość

b. Ocena osobników (funkcja `fitness`):

Ocena polega na wyliczeniu prawdopodobieństwa wybrania osobnika na podstawie rankingu opierającego się na wartości funkcji celu

Ustawiamy osobników w kolejności od największego do najmniejszego największego według wartości funkcji celu po czym obliczamy prawdopodobieństwo wybrania zgodnie z miejscem w rankingu według wzoru:

$$p = \frac{m}{sum}, \text{ gdzie } m - \text{miejsce w rankingu, } sum - \text{suma osobników}$$

Taki wzór zapewnia nam że zawsze suma prawdopodobieństw jest równa 1

c. Selekcja (funkcja selection):

Wybieramy które rozwiązania krzyżować korzystając z standardowego mechanizmu koła ruletki

Ruletka polega na wylosowaniu liczby z przedziału 0 – 1 a następnie iterujemy po osobnikach dodając ich prawdopodobieństwa do siebie aż otrzymamy liczbę większą od wylosowanej

d. Krzyżowanie (funkcja cross):

Koncepcja

Krzyżujemy części chromosomu dotyczące paczek jadących pod jeden adres (podzielone pionowymi kreskami) w tych częściach bierzemy połowę z jednego rodzica i połowę drugiego

Rodzic 1															
c_1	c_2	c_3	c_4	c_5	c_6	1	1	1	1	2	2	2	3	3	
p_1	p_2	p_3	p_4	p_5	p_6	p_7				p_{P-1}	p_P				
2	2	1	-1	3	-1	3	3	3	3	1	1	2	5	5

Rodzic 2															
c_1	c_2	c_3	c_4	c_5	c_6	1	1	1	1	2	2	2	3	3	
p_1	p_2	p_3	p_4	p_5	p_6	p_7				p_{P-1}	p_P				
-1	1	2	1	-1	3	2	2	4	4	3	3	3	6	6

Część dotyczącą ciężarówek uzupełniamy po krzyżowaniu w potomku

Potomek															
c_1	c_2	c_3	c_4	c_5	c_6	1	1	1	1	2	2	2	3	3	
p_1	p_2	p_3	p_4	p_5	p_6	p_7				p_{P-1}	p_P				
-1	1/2	1/2	-1	3	3	2	2	3	3	3	3	2	6	5

Powstałe **konflikty** (ciężarówki jadą pod 2 adresy znanaczone na czerwono) rozwiązujemy zmieniając potomka tak by spełniał warunki dodając ciężarówki nie jadące nigdzie lub dopakowując te z wolnym miejscem. Naprawa realizowana jest w funkcji `fix_pop()`

e. Mutowanie (funkcja `mutation`)

Wybieramy odsetek populacji, który chcemy zmutować.

Losujemy pojedyncze geny drugiego chromosomu i zastępujemy losowymi liczbami z ustalonego przedziału.

Modyfikujemy odpowiednie geny pierwszego chromosomu, w ten sposób powstają “uszkodzone” osobniki, posiadające nadwyżkową liczbę genów na jednej pozycji chromosomu. Odpowiadają one za adresy paczek.

Naprawiamy populację – usuwamy podwójne geny, zastępując je pojedynczym, tak by każda ciężarówka jechała tylko pod jeden adres magazynu.

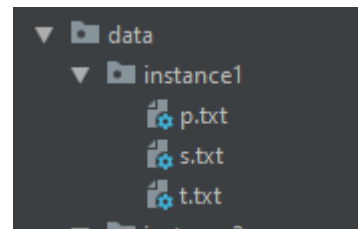
f. Kryterium stopu

Algorytm kończy działanie po wykonaniu zadanej liczby iteracji

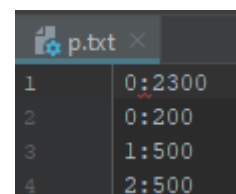
5. Dane i parametry pracy algorytmu

5.1. Dane wejściowe

Dane wejściowe zapisywane są w formacie `.txt` ze znakiem „:” jako separatorem. Znajdują się one w folderze `/data`. Dane dotyczące paczek magazynów i ciężarówek przechowujemy w osobnych plikach `txt` w jednym folderze np. `/data/instance1`

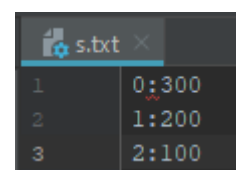


Paczki przechowywane są w formacie `id:waga`



1	0:2300
2	0:200
3	1:500
4	2:500

Magazyny w formacie `id:odległość`



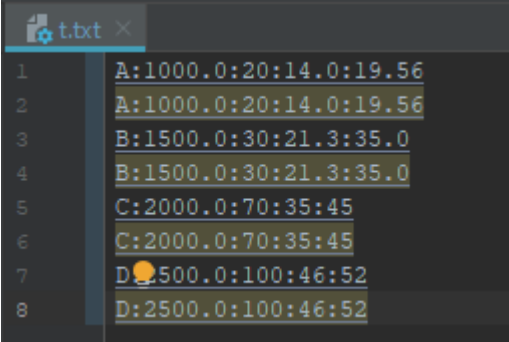
1	0:300
2	1:200
3	2:100

Ciążarówka natomiast mają więcej parametrów.

Są to kolejno:

Typ, ładowność, koszty eksploatacji, minimalne spalanie, maksymalne spalanie

Należy zaznaczyć że aby algorytm pracował poprawnie ciężarówek musi być więcej niż paczek. Algorytm wyznaczy rozwiązanie zminimalizowane jednak aby możliwa była losowość musimy zapewnić więcej ciężarówek aby algorytm mógł znajdować także te złe rozwiązania poprzez losowanie



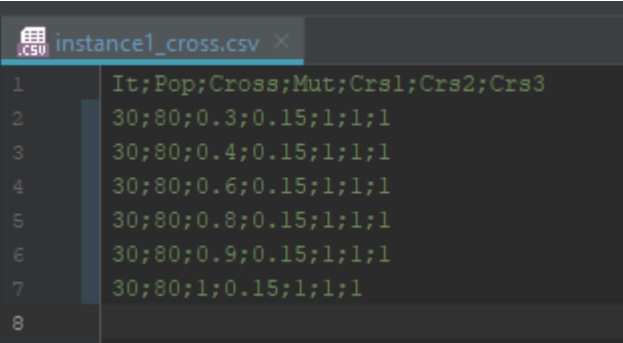
1	A:1000.0:20:14.0:19.56
2	A:1000.0:20:14.0:19.56
3	B:1500.0:30:21.3:35.0
4	B:1500.0:30:21.3:35.0
5	C:2000.0:70:35:45
6	C:2000.0:70:35:45
7	D:500.0:100:46:52
8	D:2500.0:100:46:52

5.2. Zestawy parametrów algorytmu

Przechowywane w formacie .csv zawierają kolejno

Iteracje, populacje, prawdopodobieństwa krzyżowania i mutacji oraz punkty krzyżowania.

W pokazanym na obrazku pliku są parametry dla zmiennego prawdopodobieństwa krzyżowania



1	It;Pop;Cross;Mut;Crs1;Crs2;Crs3
2	30;80;0.3;0.15;1;1;1
3	30;80;0.4;0.15;1;1;1
4	30;80;0.6;0.15;1;1;1
5	30;80;0.8;0.15;1;1;1
6	30;80;0.9;0.15;1;1;1
7	30;80;1;0.15;1;1;1
8	

II. Testy oraz Wyniki

W tej części dokumentacji przedstawimy proces w jakim dobraliśmy parametry pracy algorytmu tak aby działał on jak najlepiej oraz otrzymane wyniki dla różnych parametrów pracy

1. Wpływ zmiany pojedynczych parametrów na wyniki

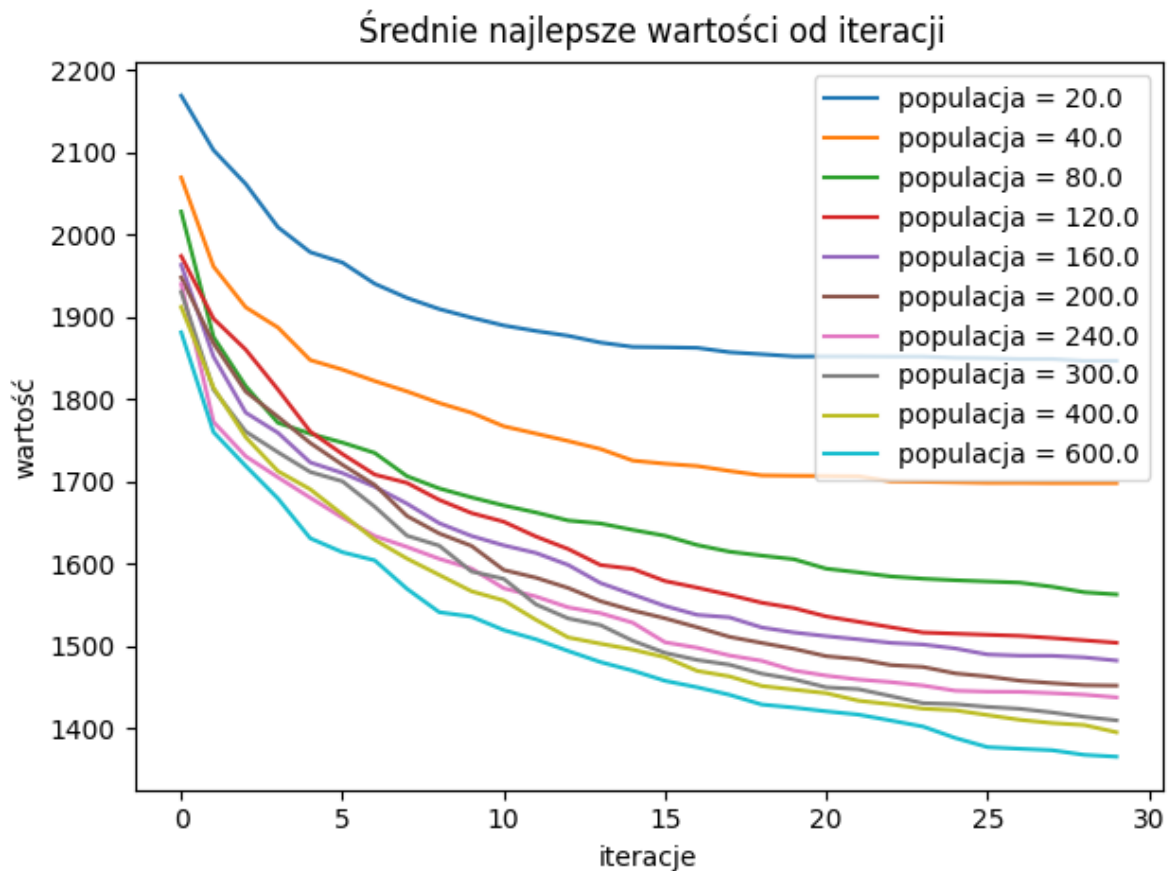
Wpływ zmiany pojedynczych parametrów sprawdziliśmy na podstawie zestawu 1.

W celu wyznaczenia jak najlepszych parametrów pracy algorytmu sprawdziliśmy jak algorytm zachowuje się przy zmianie jednego parametru. Rysując wykres przy zmiennym parametrze jesteśmy w stanie określić dla jakich wartości średnio otrzymujemy jak najlepsze wyniki a także w jaki sposób najlepsze wartości się zmieniają (przykładowo czy algorytm zbiega się za szybko czy za wolno)

Nie będą to na pewno parametry optymalne jednak dadzą one o wiele lepsze wyniki niż zestaw losowych parametrów

Wykresy były uśrednione dla 100 niezależnych uruchomień algorytmu co obrazuje w dobry sposób czy poprawiamy rozwiązania czy nie jednak należy pamiętać że jest to algorytm losowy więc przy pojedynczym uruchomieniu zawsze możemy otrzymać zły lub dobry wynik

1.1. Zmienna populacja



Średnie czasy od populacji

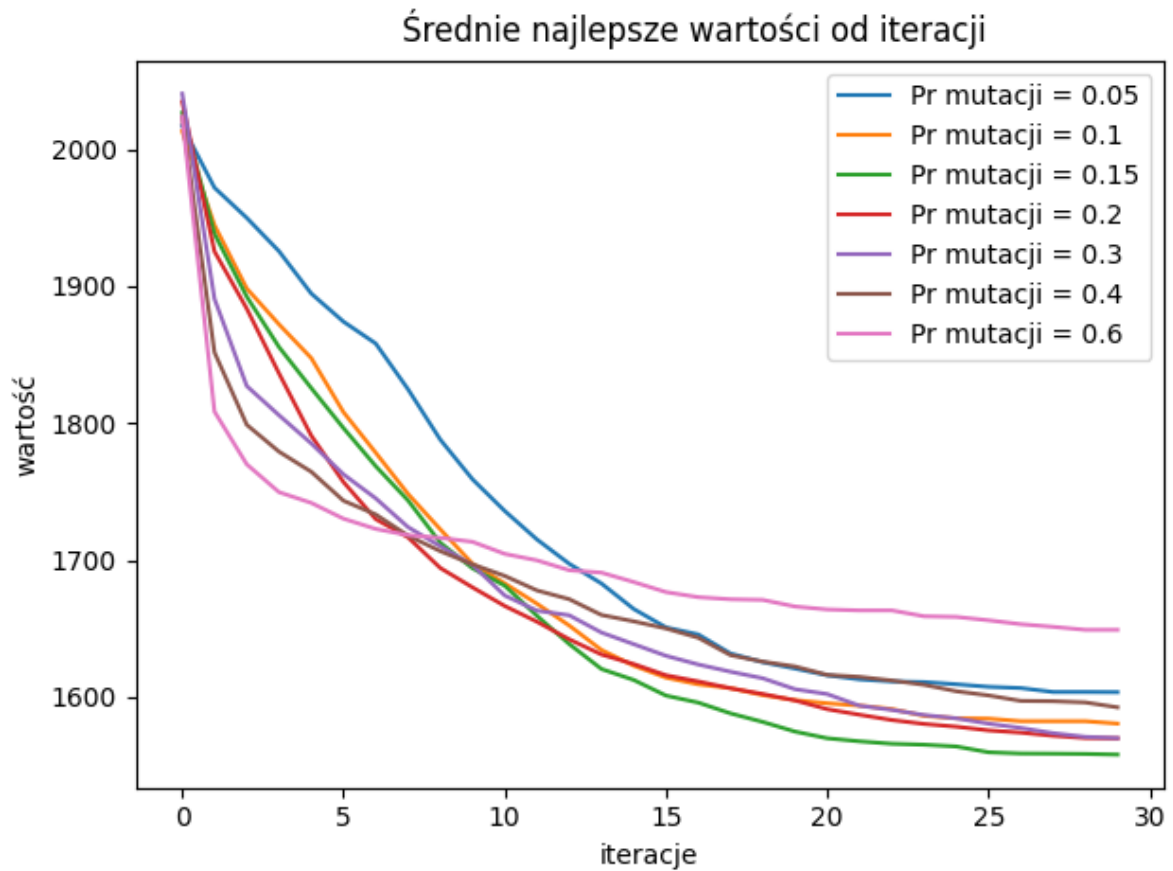
populacja	20	40	80	120	160
średni czas[s]	0.069	0.141	0.288	0.443	0.606
populacja	200	240	300	400	600
średni czas[s]	0.697	0.939	1.110	1.568	2.535

Wzrost populacji powoduje znaczne obniżenie średniej najlepszej wartości jednak powoduje także znaczne zwiększenie czasu wykonania algorytmu. Dla tego zestawu danych populacje rzędu 100-200 dają sensowne wyniki które można następnie ulepszyć zmieniając pozostałe parametry a jednocześnie nie wymagają takiego nakładu czasowego

Gdybyśmy jednak chcieli otrzymywać jak najdokładniejsze wyniki za każdym razem należało by zwiększyć populację do większych rozmiarów

Dodatkowo należy pamiętać że dla większego rozmiaru problemów (więcej paczek i magazynów) powinniśmy zwiększyć populację z racji na większą liczbę potencjalnych rozwiązań

1.2. Zmiana prawdopodobieństwa mutacji



Średnie czasy od prawdopodobieństwa mutacji

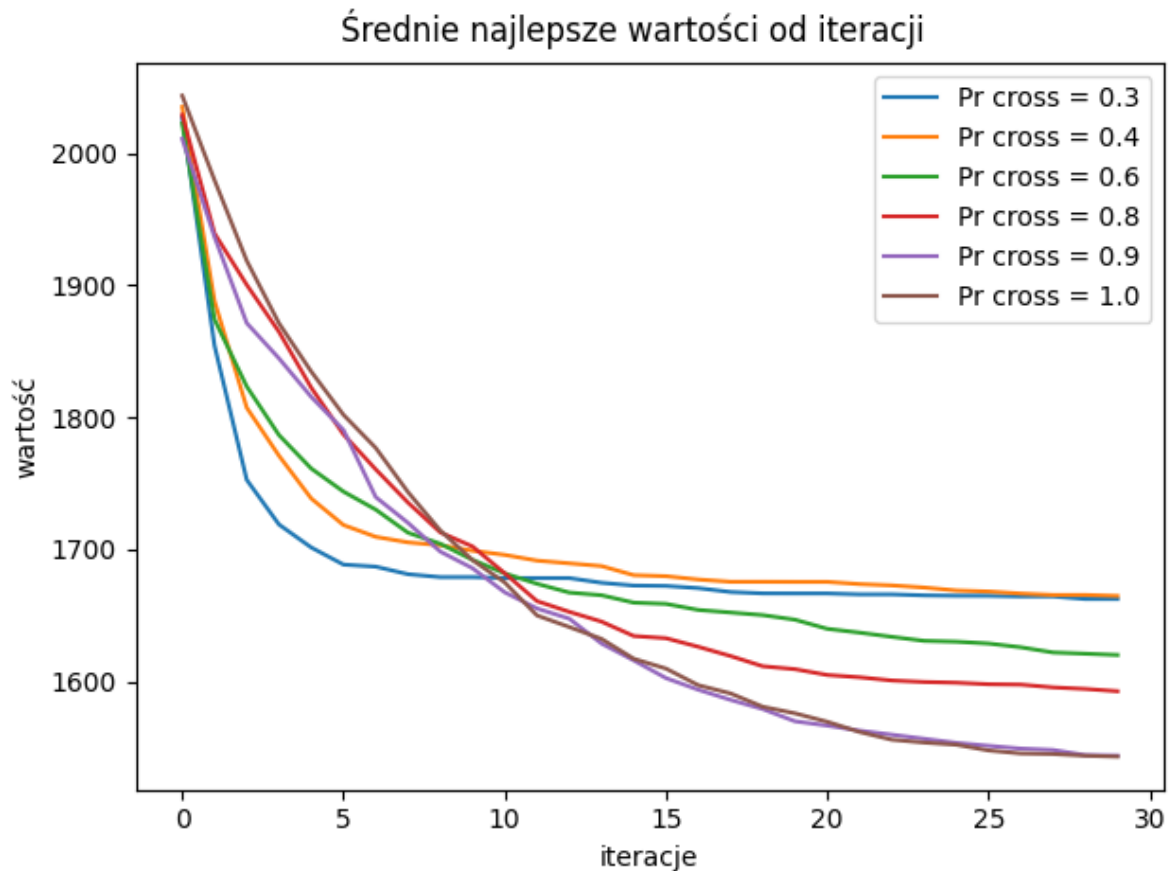
Pr mut	0.05	0.1	0.15	0.2	0.3	0.4	0.6
czas[s]	0.242	0.253	0.266	0.275	0.297	0.320	0.362

Średnie najlepsze wyniki są najmniejsze dla prawdopodobieństwa 0.15

Widzimy że zwiększanie parametru powoduje szybsze przeszukiwanie rozwiązań (wartości maleją szybko dla początkowych iteracji) jednak dla zbyt dużych prawdopodobieństw przeszukiwanie zbioru rozwiązań jest zbyt losowe i nie prowadzi do znalezienia sensownych wyników.

Warto zauważyć że odpowiednio duża mutacja pozwala osiągać lepsze wyniki przy niewielkim koszcie czasowym w porównaniu do zmiany populacji

1.3. Zmiana prawdopodobieństwa krzyżowania



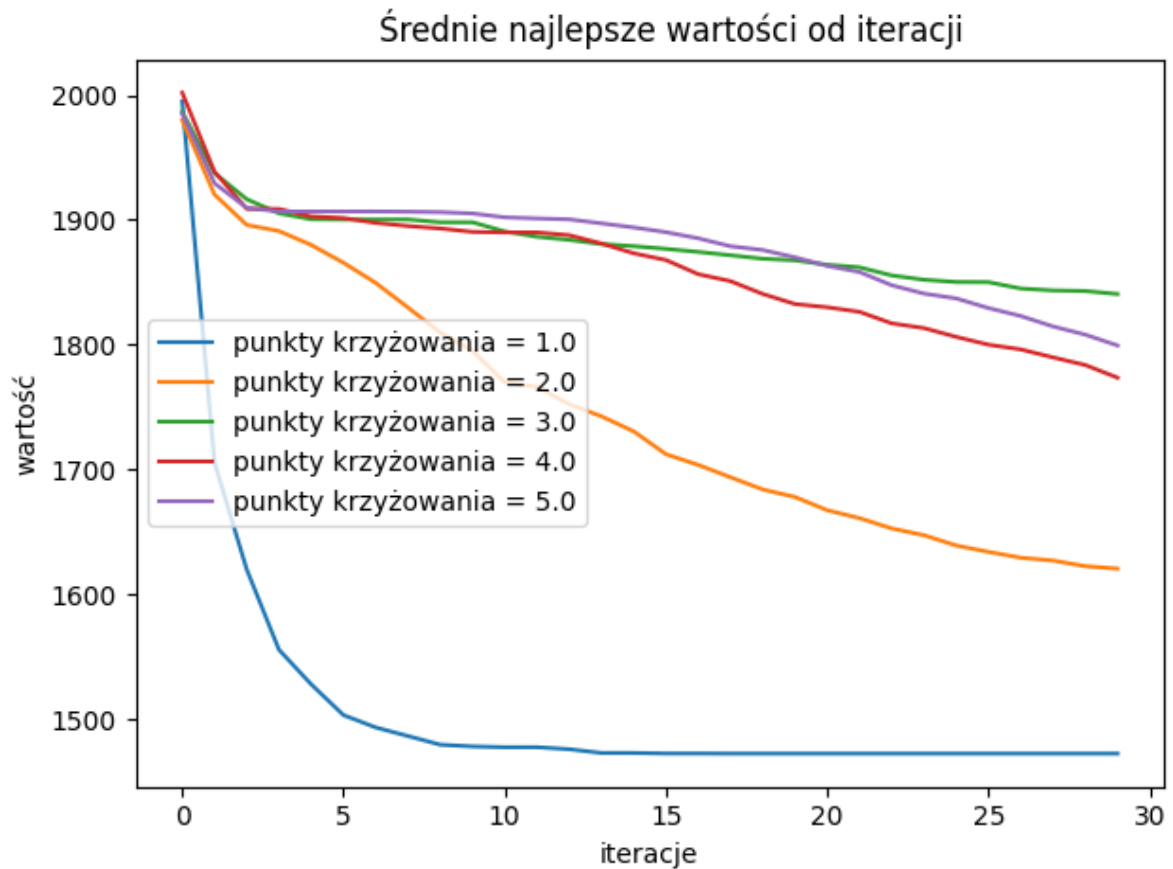
Średnie czasy od prawdopodobieństwa krzyżowania

Pr cross	0.3	0.4	0.6	0.8	0.9	1.0
czas[s]	0.190	0.204	0.236	0.281	0.305	0.320

Jak widać zwiększenie prawdopodobieństwa krzyżowania pomaga otrzymać dokładne wyniki częściej najlepsze wyniki otrzymaliśmy dla prawdopodobieństw z zakresu 0.9 – 1.0

Początkowo myśleliśmy że mniejszy współczynnik krzyżowania pomoże nam znajdować lepsze rozwiązania poprzez większe połączenie z poprzednią populacją (po krzyżowaniu zostaje w populacji kilku najlepszych rodziców) jednak okazało się to być niezbyt przydatne. Widzimy że może i przyspiesza to przeszukiwanie zbioru rozwiązań (niebieski wykres początkowo opada o wiele szybciej) jednak na dłuższą metę nie ma to sensu i algorytm o wiele szybciej zbiega do minimum lokalnego które dalekie jest od globalnego. Powoduje to brak dalszego polepszania rozwiązań

1.4. Zmiana ilości punktów krzyżowania



Widzimy że ewidentnie najlepsze wyniki otrzymamy dla 1 punktu krzyżowania wraz z wzrostem ich ilości wyniki drastycznie się pogarszają i przeszukiwanie zbioru rozwiązań jest wolniejsze

1.5. Ostatecznie dobrane parametry

Na podstawie powyższych wykresów i wniosków wyznaczyliśmy zestaw najlepszych parametrów do naszego algorytmu.

Populacja: 100 – 200 w przypadku kiedy zależy nam na czasie i jak największa gdy chcemy jak najdokładniejsze wyniki

Współczynnik krzyżowania: 0.9 – 1.0

Współczynnik mutacji: 0.15

Punkty krzyżowania: 1

2. Wyniki

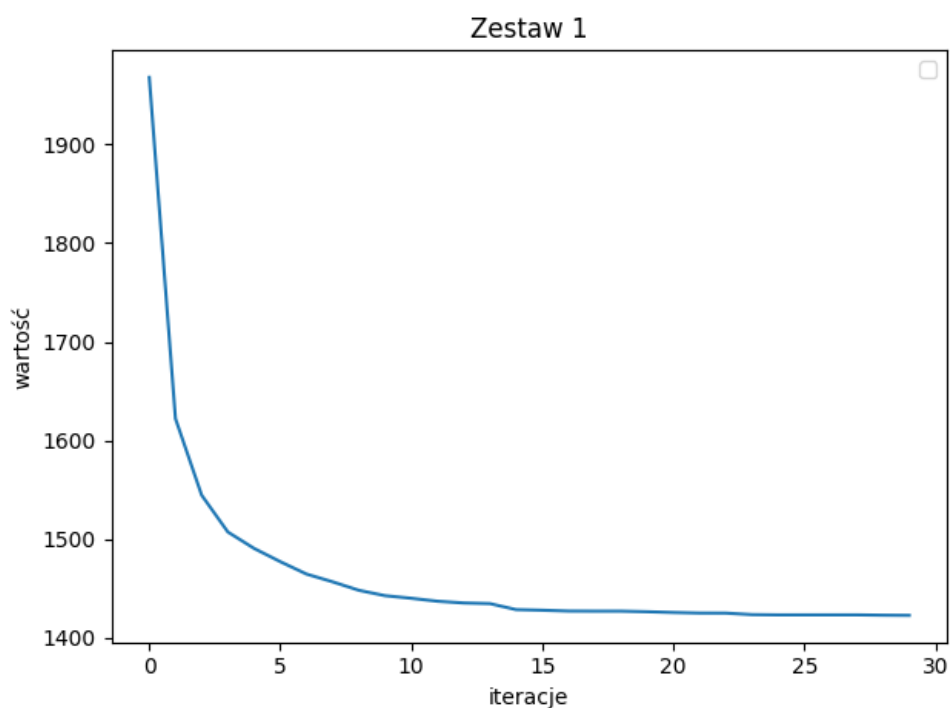
2.1. Zestaw 1

Dane:

Jest to zestaw o średniej wielkości

Magazyn 1 Wagi paczek [kg]	Magazyn 2 Wagi paczek [kg]	Magazyn 3 Wagi paczek [kg]
600	400	200
700	1567	100
500	450	1500
50	343	100
400	500	200
	400	2020
		200
		100

Wyniki otrzymane dla wyznaczonych w punkcie 1 parametrów uśrednione dla 200 powtórzeń



Średnio otrzymaliśmy wyniki na poziomie około 1420 zł za transport widzimy że algorytm osiąga najlepsze rozwiązanie już po około 15 iteracjach

Najlepsze otrzymane rozwiązanie:

Najlepsze rozwiązanie w 100 powtórzeniach

```
Średni czas dla zestawu 0 to: 1.3879483032226563
Koszt: 1155.5079999999998

Magazyn: 0:
Ciezarowka: 2000.0 , paczki: [0, 1, 2] suma wag: 1800.0
Ciezarowka: 1000.0 , paczki: [3, 4] suma wag: 450.0

Magazyn: 1:
Ciezarowka: 2000.0 , paczki: [5, 7, 8, 9] suma wag: 1693.0
Ciezarowka: 2000.0 , paczki: [6, 10] suma wag: 1967.0

Magazyn: 2:
Ciezarowka: 2000.0 , paczki: [11, 13, 14, 15] suma wag: 2000.0
Ciezarowka: 2500.0 , paczki: [12, 16, 18] suma wag: 2220.0
Ciezarowka: 2500.0 , paczki: [17] suma wag: 200.0
```

Paczki załadowane są w sensowny sposób widzimy że algorytm ewidentnie minimalizuje ilość wykorzystanych ciężarówek i pakuje paczki w taki sposób by „zapchać” ciężarówkę do pełna. Zdarza się jednak że z niewiadomych przyczyn wybiera największą ciężarówkę i pakuje do niej pojedynczą małą paczkę np. ciężarówka o ładowności 2500 jadąca do magazynu 3 zabiera paczkę o wadze 200.

2.2. Zestaw 2 (duży)

Dane:

Zestaw ten zawiera dużą ilość paczek jadących pod 5 różnych adresów. Posiadamy 4 różne typy ciężarówek

Magazyn 0: 10 paczek o łącznej wadze 4500 kg

Magazyn 1: 12 paczek o łącznej wadze 7320 kg

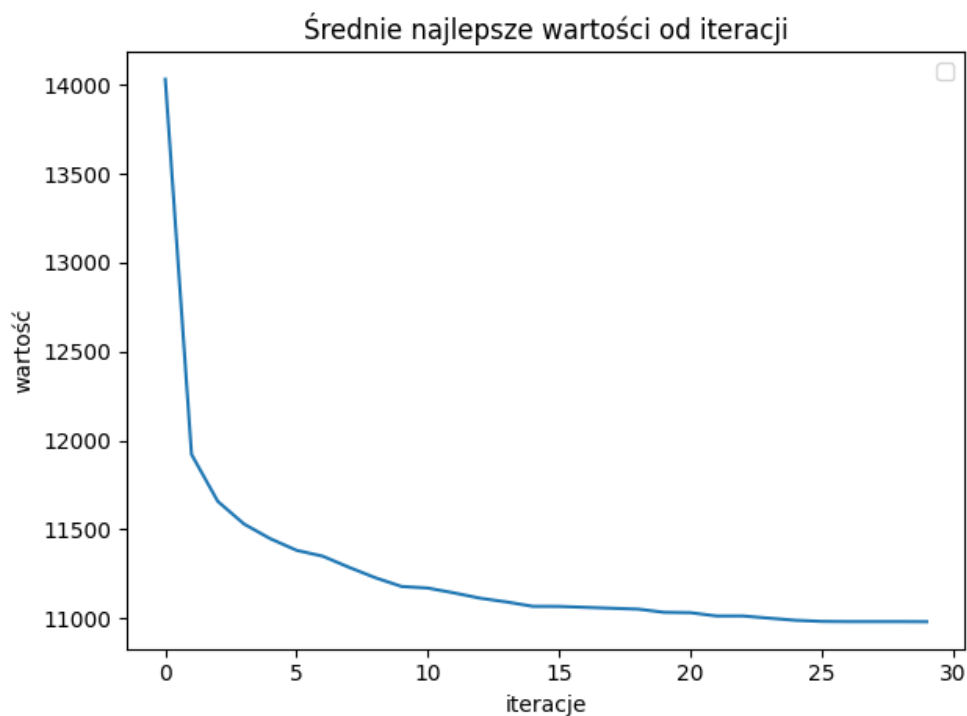
Magazyn 2: 8 paczek o łącznej wadze 4420 kg

Magazyn 3: 9 paczek o łącznej wadze 4903 kg

Magazyn 4: 12 paczek o łącznej wadze 7313 kg

Średnie wyniki:

Wyniki otrzymane dla parametrów wyznaczonych w punkcie pierwszym i populacji 300 uśrednione dla 50 powtórzeń



Średni czas 8.072

Dla większej ilości paczek znacznie zwiększa się czas wykonania algorytmu widać jednak że dalej skutecznie pomniejsza on wartości funkcji celu i znajduje coraz lepsze rozwiązania.

Średnie najlepsze otrzymane rozwiązanie to około 11000

Najlepsze otrzymane rozwiązanie:

```
Koszt: 9997.32

Magazyn: 0:
Ciezarowka: 2500.0 , paczki: [0, 1, 5, 8, 9] suma wag: 2350.0
Ciezarowka: 1000.0 , paczki: [2] suma wag: 500.0
Ciezarowka: 2000.0 , paczki: [3, 4, 6, 7] suma wag: 1650.0

Magazyn: 1:
Ciezarowka: 2500.0 , paczki: [10, 13, 15, 21] suma wag: 1543.0
Ciezarowka: 2500.0 , paczki: [11] suma wag: 1567.0
Ciezarowka: 2000.0 , paczki: [12, 18] suma wag: 900.0
Ciezarowka: 1500.0 , paczki: [14] suma wag: 500.0
Ciezarowka: 2000.0 , paczki: [16, 19] suma wag: 743.0
Ciezarowka: 2500.0 , paczki: [17] suma wag: 1567.0
Ciezarowka: 2500.0 , paczki: [20] suma wag: 500.0

Magazyn: 2:
Ciezarowka: 2500.0 , paczki: [22, 23, 24, 28] suma wag: 2000.0
Ciezarowka: 2000.0 , paczki: [25] suma wag: 100.0
Ciezarowka: 2000.0 , paczki: [26] suma wag: 200.0
Ciezarowka: 2500.0 , paczki: [27, 29] suma wag: 2120.0

Magazyn: 3:
Ciezarowka: 1000.0 , paczki: [30, 35] suma wag: 793.0
Ciezarowka: 1000.0 , paczki: [31, 37] suma wag: 1000.0
Ciezarowka: 1000.0 , paczki: [32, 38] suma wag: 800.0
Ciezarowka: 2500.0 , paczki: [33, 34, 36] suma wag: 2310.0

Magazyn: 4:
Ciezarowka: 2000.0 , paczki: [39, 43, 49, 51] suma wag: 1543.0
Ciezarowka: 2500.0 , paczki: [40, 46, 50] suma wag: 900.0
Ciezarowka: 2000.0 , paczki: [41] suma wag: 500.0
Ciezarowka: 2500.0 , paczki: [42, 48] suma wag: 2070.0
Ciezarowka: 2500.0 , paczki: [44, 45] suma wag: 2100.0
Ciezarowka: 2500.0 , paczki: [47] suma wag: 200.0
```

Widzimy że najlepsze rozwiązanie ma wartość poniżej 10000 co jest znacznie lepszym wynikiem od wartości średniej. Dla większych problemów algorytm wyraźnie gorzej znajduje rozwiązania, często zdarza się że ciężarówki są niemal puste pomimo że jest miejsce na ich paczki w innych pojazdach

2.3. Zestaw 3 (instacja testowa)

Dane:

Zestaw ten zawiera bardzo małą ilość paczek jadących pod 3 różne adresy. Posiadamy 4 różne typy ciężarówek

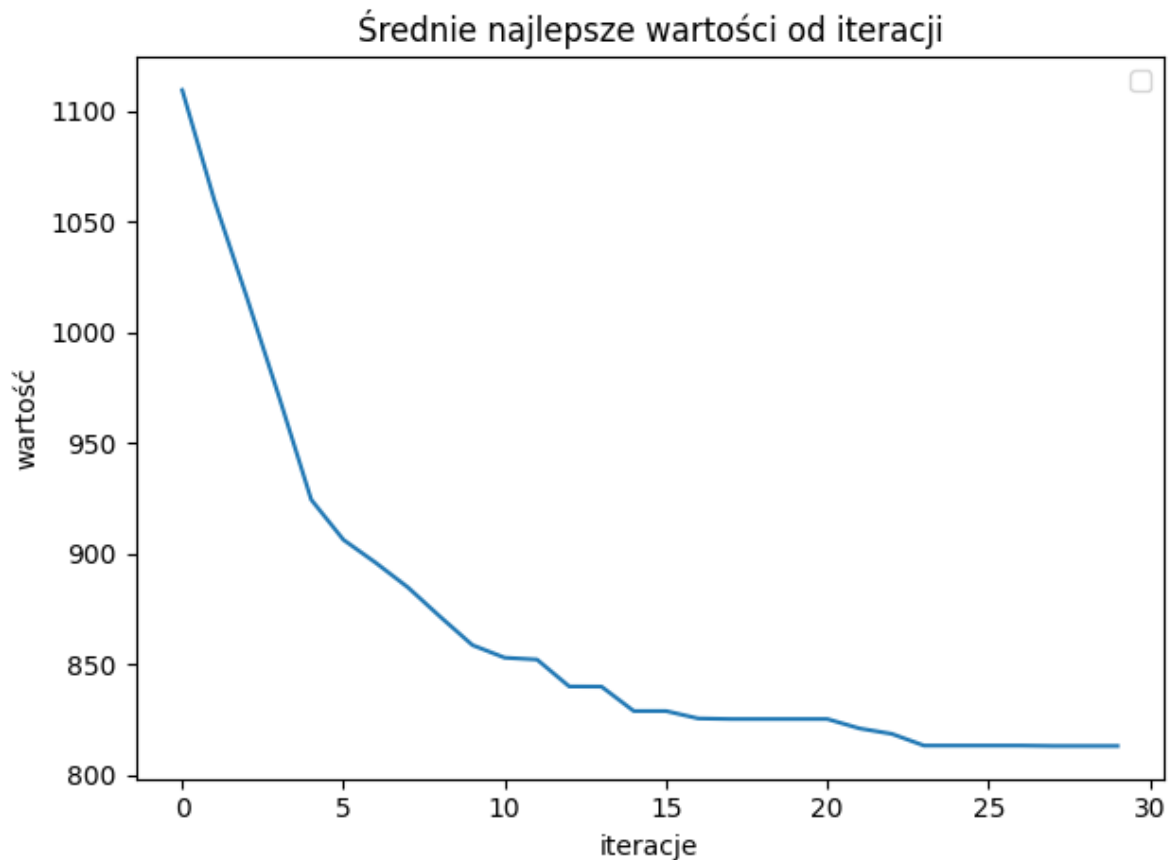
Magazyn 0: 2 paczki o łącznej wadze 2500kg

Magazyn 1: 1 paczka o wadze 500kg

Magazyn 2: 1 paczka wadze 500kg

Średnie wyniki:

Wyniki otrzymane dla parametrów wyznaczonych w punkcie pierwszym i populacji 300 uśrednione dla 50 powtórzeń



Otrzymane rozwiązanie:

```
Średni czas dla zestawu 0 to: 0.08828418731689452
Koszt: 541.4

Magazyn: 0:
Ciezarowka: 2500.0 , paczki: [0, 1] suma wag: 2500.0

Magazyn: 1:
Ciezarowka: 1000.0 , paczki: [2] suma wag: 500.0

Magazyn: 2:
Ciezarowka: 2500.0 , paczki: [3] suma wag: 500.0
```

Algorytm działa poprawnie, odpowiednio pakując paczki pod konieczne adresy. Widoczny jest jednak błąd omówiony wyżej, paczka o wadze 500kg zamiast zostać przydzielona do ciężarówki o ładowności 1000kg/m³, została zapakowana do największego z dostępnych pojazdów.

III. Dokumentacja GUI

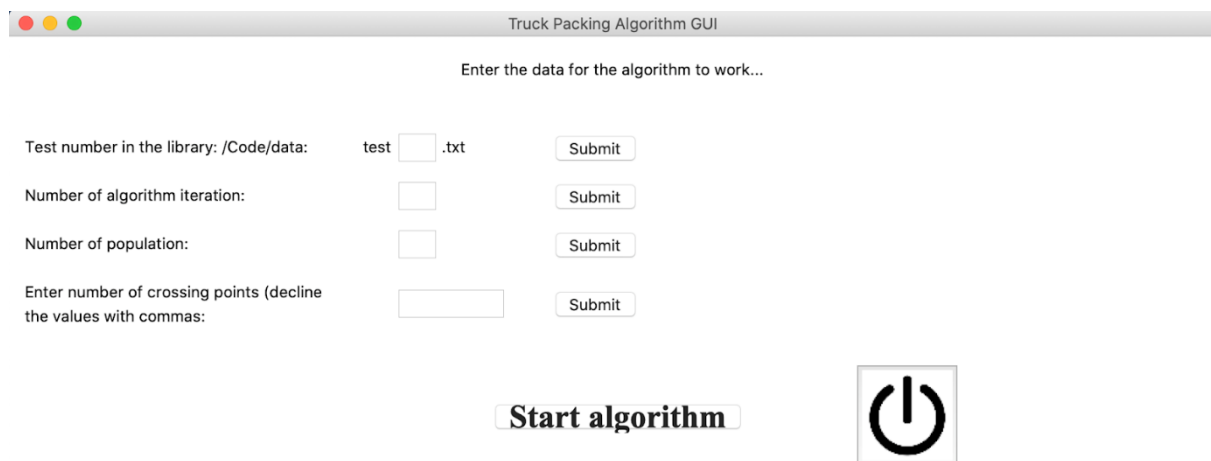
1. Cel

W celu łatwego i przejrzystego poruszania się po kolejnych, najważniejszych krokach algorytmu, przygotowaliśmy graficzny interfejs użytkownika (GUI). Ścieżka dostępu: .../Truck-Packing-Algorithm/Code/gui.py. Aby uruchomić

2. Elementy

Okno główne:

Miejsce w którym dokonujemy wyboru najistotniejszych parametrów algorytmu oraz przedstawiamy najlepszy otrzymany wynik. Działania poszczególnych przycisków są adekwatne do ich opisów tekstowych bądź graficznych. Ważne by przy wyborze punktów krzyżowania ustawić takie wartości aby były one dzielnikami liczby paczek. Generalnie zaleca się wybór punktów krzyżowania o wartości 1.



Truck Packing Algorithm GUI


Enter the data for the algorithm to work...

Test number in the library: /Code/data: test .txt

Number of algorithm iteration:

Number of population:

Enter number of crossing points (decline the values with commas:



The best solution obtained by this algorithm:

- Transportation cost: 1967.77 PLN
- Loading of trucks:
 0. Truck id: 19, Load: 2500.0, Packages: [0, 1, 2, 3, 4], Sum of weights: 2250.0, Address: 0
 1. Truck id: 14, Load: 2000.0, Packages: [5, 6], Sum of weights: 1967.0, Address: 1
 2. Truck id: 12, Load: 1000.0, Packages: [7, 10], Sum of weights: 850.0, Address: 1
 3. Truck id: 21, Load: 1500.0, Packages: [8, 9], Sum of weights: 843.0, Address: 1
 4. Truck id: 23, Load: 2500.0, Packages: [11, 12, 13, 14, 15, 17, 18], Sum of weights: 2400.0, Address: 2
 5. Truck id: 15, Load: 2500.0, Packages: [16], Sum of weights: 2020.0, Address: 2

Uwaga: Przycisk -Start algorithm- nie uruchomi algorytmu dopóty, dopóki nie zostaną poprawnie uzupełnione wszystkie okna. Informacje potwierdzające poprawność wpisanych danych pojawiają się po zatwierdzeniu ich przyciskiem -Submit-.

test	<input type="text" value="2"/>	.txt	<input type="button" value="Submit"/>	Confirmation: Your data set is: test2.txt
	<input type="text" value="1,2"/>		<input type="button" value="Submit"/>	Enter correct data, the name should be a number
	<input type="text" value="20"/>		<input type="button" value="Submit"/>	Confirmation: Number of population is: 20
	<input type="text" value="1,2"/>		<input type="button" value="Submit"/>	Too low number of cuts

Okno pomocnicze:

Miejsce, w którym przedstawione są informacje o ilości paczek, jakie mają zostać dostarczone pod konkretny adres (magazyn). Dane zawarte w tym oknie pozwalają poprawnie określić liczbę punktów krzyżowania, które należy określić w ostatnim polu danych wejściowych w oknie głównym. (Sposób doboru punktów krzyżowania: Patrz: 4.3 ,podp. Krzyżowanie).

List of used packages and storages	
Number of packages to storage no. 0:	5
Number of packages to storage no. 1:	6
Number of packages to storage no. 2:	8

Wnioski

Korzystanie z algorytmu genetycznego jako algorytmu głównego dla naszego problemu pozwoliło nam uzyskać przybliżone zoptymalizowane rozwiązania naszej funkcji celu. Ze względu na specyfikę tego algorytmu oraz jej heurystykę, rozwiązania nigdy nie są najlepsze, jednak plasują się blisko rozwiązania optymalnego. Największą wadą działania algorytmu zdaje się być nieumiejętne ładowanie niektórych paczek do niemalże pustych ciężarówek, jednocześnie zostawiając miejsce w innych. Powoduje to wzrost kosztów eksploatacji pojazdów a także paliwa i wpływa negatywnie na wynik. Powodem tego problemu jest najprawdopodobniej nieumiejętna implementacja jednej z funkcji dodatkowych. Po analizie kodu nie udało nam się jednak naprawić tego błędu.

Analizując zmienne parametry i ich wpływ na działanie algorytmu genetycznego zaobserwowaliśmy, że wzrost liczebności populacji wprawdzie poprawia nasze rozwiązanie, lecz tylko do pewnego momentu (potem wyniki nieznacznie się pogarszają, jednocześnie

znacznie dłużej wykonując algorytm). Przykładowo dla zestawu pierwszego dobrą populacją było 100-200 osobników.

Ważnym elementem naszych wniosków było zbadanie wpływu losowości w naszym kodzie. Poddając analizie część dotyczącą krzyżowania oraz mutacji doszliśmy do, zdaje się na pierwszy rzut oka, sprzecznych wniosków. Tak jak podejrzewaliśmy, wysokie wartości prawdopodobieństwa mutacji genu osobnika znacznie pogarszały jakość znalezionej odpowiedzi. Okazało się jednak, że w procesie wymierania oraz tworzenia wtórnej populacji wysokie prawdopodobieństwo krzyżowania genów było pożądane. Co ciekawsze, bardzo duży wpływ na ten proces miała decyzja o wyborze ilości punktów krzyżowania, a najlepszym wyborem było podzielenie chromosomu jedynie na dwie części.

Im bardziej złożony problem, tym mniej dokładnie rozwiązania i dłuższy czas wykonywania programu. Mimo to, porównując (przy dużej liczbie iteracji) uśredniony oraz najlepszy wynik widzimy jak trudna jest dokładna implementacja algorytmu genetycznego. Widzimy także jego główne wady, do których należy duży wpływ parametrów dodatkowych instancji na dokładność działania oraz potrzeba dokładnej analizy wielu zestawów testowych w celu określenia jakości rozwiązań.