Dokumentacja projektowa

Program do układania diety - ALHE 17L

Adam Małkowski
Piotr Włodkowski

1. Opis problemu

Zadanie polega na ułożeniu N-dniowej diety, która dostarczać będzie wymaganą ilość wartości odżywczych (np. tłuszcz, białko, węglowodany). Należy również zadbać, by przygotowana dieta nie była zbyt monotonna. Te same produkty nie mogą pojawiać się zbyt często w jadłospisie.

Przyjęte założenia:

- Każdy dzień składa się z 3 posiłków (śniadanie, obiad, kolacja)
- Posiłek jest złożony z pewnego zbioru produktów oraz ich wag
- Dzienne zapotrzebowanie na wartości odżywcze jest rozdzielane w zadanych proporcjach pomiędzy posiłki
- Niespełnienie bądź przekroczenie danej normy w ramach posiłku skutkuje przeniesieniem zapotrzebowania na następny posiłek w ramach dnia (tj. jeżeli śniadanie będzie zbyt mało kaloryczne, to algorytm będzie próbował nadrobić to obiadem)
- Istnieją 4 listy produktów - produkty śniadaniowe, produkty obiadowe, produkty kolacyjne oraz produkty ogólne
- Z produktem związane jest *n* parametrów określających wartości odżywcze
- Każdy z *n* parametrów będzie miał przypisaną wagę, która określa, jak poszczególny parametr ważny jest w diecie ludzkiej
- Z każdym produktem związana jest jego rozdzielczość wagowa (w zależności od natury posiłku - kanapki są w sztukach, ryż w gramach)
- Z każdym produktem związana jest jego maksymalna i minimalna waga na posiłek
- Z każdym produktem związany jest parametr *l* określający jak często produkt może pojawiać się w diecie - 1 na *l* dni
- Jeden posiłek może się składać z maksymalnie *k* produktów

Przestrzeń przeszukiwań:

Problem został rozbity na podproblemy znajdowania poszczególnych posiłków. Przestrzeń przeszukiwań jest rozważana w kontekście problemu znalezienia pojedynczego posiłku, który spełnia zadane zapotrzebowanie na wartości odżywcze.

Punkt w przestrzeni jest pewnym zbiorem produktów (do *k* produktów, określonych jako parametr diety) wraz z ich masą (dozwoloną dla danego produktu).

Sąsiadują ze sobą punkty różniące się typem, masą albo obecnością jednego lub kilku z produktów.

Punkty muszą spełniać rygory zadania - co oznacza, że wszystkie mogą być terminalne.

Postać funkcji celu (w problemie minimalizacji):

W ramach jednego posiłku - norma różnicy wektorów danego zapotrzebowania oraz otrzymanych wartości odżywczych. (Z uwzględnieniem różnych wag poszczególnych wartości odżywczych)

2. Opis zaimplementowanej aplikacji

Wykorzystana metaheurystyka:
Implementacja opiera się o połączenie dwóch metaheurystyk - algorytmu wykorzystującego mechanizm TABu oraz algorytmu ewolucyjnego. Zastosowanie tabu służy do zapewnienia różnorodności diety. Algorytm ewolucyjny służy do znajdowania składu poszczególnych posiłków.

Program na wstępnie pobiera informacje na temat dostępnych produktów (w tym ich wartościach odżywczych oraz możliwej częstości występowania), zadany wektor dziennego zapotrzebowania na poszczególne składniki odżywcze (wraz z wagami dotyczącymi ich ważności) oraz potencjalnie parametry działania algorytmu ewolucyjnego.

Iteracyjnie analizowane są kolejne dni, a w ramach nich kolejne posiłki, oraz uruchamiana jest metoda optymalizująca najbliższy posiłek. Przyjmuje ona listę produktów powstałą jako suma listy odpowiadającej danemu posiłkowi - śniadanie, obiad lub kolacja - oraz listy ogólnej - wszystko to, pomniejszone o elementy aktualnie będące w TABU. Metoda optymalizująca przyjmuje również wartość zapotrzebowania na wartości odżywcze obliczona jako określona część dziennego zapotrzebowania odpowiadająca danemu posiłkowi zmniejszona/zwiększona o niespełnienie wymagań dotyczących wartości odżywczych w poprzednim posiłku w ramach dnia. Po znalezieniu odpowiedniego posiłku informacje o nim są zapisywane oraz aktualizowane jest TABU. Zmniejszane są wartości mówiące ile czasu produkt ma pozostać w TABU, produkty których czas już nadszedł, opuszczają TABU, a produkty z nowo wygenerowanego posiłku są dodawane do TABU.

Konkretny posiłek jest znajdowany przy pomocy algorytmu ewolucyjnego.
Populacja początkowa jest wygenerowana przy pomocy metody generującej brutalnie posiłek, optymalizując go pod kątem jednego, losowo wybranego, parametru.
W kolejnych krokach algorytm:

- Z aktualnej populacji wybiera zbiór osobników (równoliczny całej populacji) poprzez wybieranie ze zwracaniem z prawdopodobieństwem odwrotnie proporcjonalnym do wartości funkcji celu danego osobnika.
- Iteracyjnie przeglądani są kolejni osobnicy otrzymanej populacji, każdy może zostać poddanym krzyżowaniu z zadanym prawdopodobieństwem. W procesie krzyżowania, z otrzymanej populacji losowany jest drugi osobnik. W przypadku gdy zostanie wylosowany ten sam osobnik - zostaje on zwracany, w przeciwnym przypadku iteracyjnie przeglądane są kolejne produkty pierwszego z posiłków, i dla każdego, z zadanym prawdopodobieństwem, losowany jest produkt z drugiego posiłku - w skład wynikowego posiłku wchodzą produkty które nie były modyfikowane oraz produkty zmodyfikowane charakteryzowane typem z drugiego z posiłków oraz wagą z pierwszego (potencjalnie przyciętą, tak, aby pozostały spełnione rygory). Zwracana

zostaje populacja złożona z osobników którzy nie zostali poddani krzyżowaniu oraz osobników powstałych w przypadkach poszczególnych krzyżowań.

- Iteracyjnie przeglądani są kolejni osobnicy otrzymanej populacji, każdy może zostać poddany mutacji. W ramach mutacji przeglądane są konkretne produkty występujące w posiłku, każdy może zostać zmieniony na jeden z 4 sposobów - może mieć zmieniony typ produktu, może mieć zmienioną wagę produktu, może zostać usunięty z posiłku albo zostać pozostawiony bez zmiany. Dodatkowo, do posiłku z zadanym prawdopodobieństwem może zostać dodany losowy produkt z listy dostępnych produktów i ze standardową wagą. Zwracana populacja powstaje jako zbiór osobników otrzymanych w niniejszym kroku.
- Populacja w następnym kroku powstaje z określonej liczby osobników najlepszych w poprzednim kroku oraz najlepszych osobników powstałych w procesie mutacji (ich liczba jest stała i wynosi ogólną liczebność populacji pomniejszoną o liczbę najlepszych osobników wybranych z poprzedniej populacji)

Proces ten trwa określoną liczbę iteracji i po wykonaniu się ostatni raz, z populacji wybierany jest najlepszy osobnik (posiłek) i zwracany.

W powyższym algorytmie występują następujące dane wejściowe:

1. Zbiór dostępnych produktów podzielony na listy odpowiadające konkretnym rodzajom posiłków (śniadanie, obiad, kolacja) oraz listy ogólnej
2. Dzienne zapotrzebowanie na wartości odżywcze (wektor *n* wymiarowy)
3. Wagi poszczególnych wartości odżywczych
4. Procentowy podział składników odżywczych na poszczególne posiłki (wektor *3* wymiarowy, np. [0.1;0.8;0.1] oznacza, że śniadanie i kolacja powinny zaspokajać po 10% dziennego zapotrzebowania na każdy ze składników odżywczych a obiad 80% tego zapotrzebowania)
5. Liczba dni diety

Oraz parametry:

1. Liczba osobników występujących w populacji
2. Liczba osobników elitarnych, mających pewność przejścia do następnej populacji bez zmian
3. Prawdopodobieństwo krzyżowania
4. Prawdopodobieństwo zmiany typu produktu w procesie krzyżowania
5. Prawdopodobieństwo zmiany typu produktu w procesie mutacji
6. Prawdopodobieństwo zmiany wagi produkt w procesie mutacji
7. Prawdopodobieństwo usunięcia produktu w procesie mutacji
8. Prawdopodobieństwo dodania nowego produktu w procesie mutacji

Strojenie parametrów było realizowane poprzez zmienianie wartości jednego z parametrów przy pozostawieniu innych, wielokrotnym uruchomieniu algorytmu i analizowanie wpływu zmiany na rezultaty. Parametry 1 i 2 zostały wybrane tak, aby ich rozsądne powiększenie nie przynosiło już dużej różnicy jakości (gdyż ich powiększenie znacznie wydłuża czas działania algorytmu). Parametry 3-8 zostały wybrane orientacyjnie, zgodnie z pewną intuicją autorów oraz poddane strojeniu opisanego w pierwszym zdaniu niniejszego akapitu.

3. Odstępstwa od dokumentacji wstępnej

Zmieniona została funkcja celu. Zachowany został sens, zebranie informacji na temat odchyleń poszczególnych wartości odżywczych od pożądanych, jednakże zrezygnowano z ich podnoszenia do kwadratu. Aktualna postać operuje na mniejszych liczbach i jest ładniejsza w interpretacji i reprezentacji jednocześnie nie powodując pogorszenia otrzymywanych rozwiązań.

4. Eksperymenty

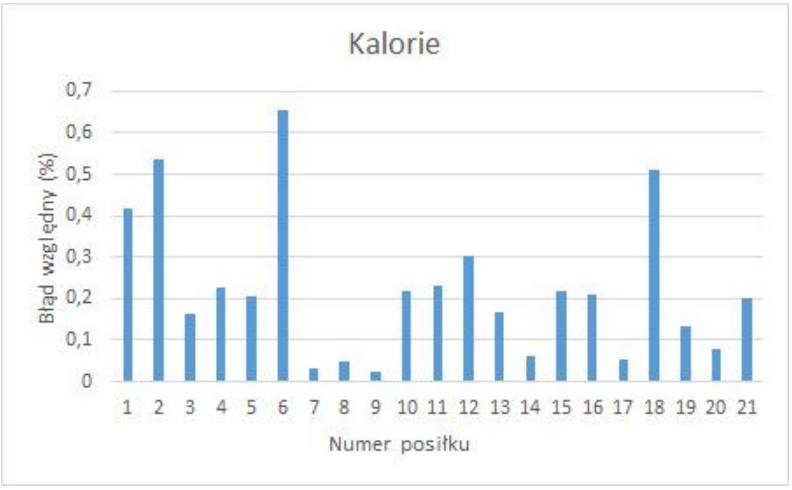
Eksperyment 1:

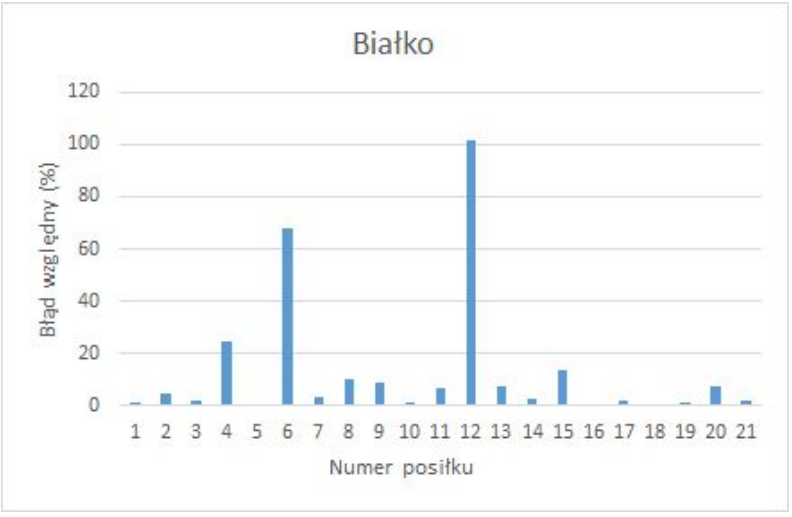
```
Liczba iteracji: 100
Liczebność populacji: 30
Liczebność elity: 10
Prawdopodobieństwo krzyżowania: 25
Prawdopodobieństwo zamiany typu podczas krzyżowania: 15
Prawdopodobieństwo zmiany typu w ramach mutacji: 10%
Prawdopodobieństwo zmiany wagi produktu w ramach mutacji: 50%
Prawdopodobieństwo usunięcia produktu w ramach mutacji: 5%
Prawdopodobieństwo dodania produktu w ramach mutacji: 30%
Liczba dni diety: 7
Pożądane wartości odżywcze w posiłku:
```

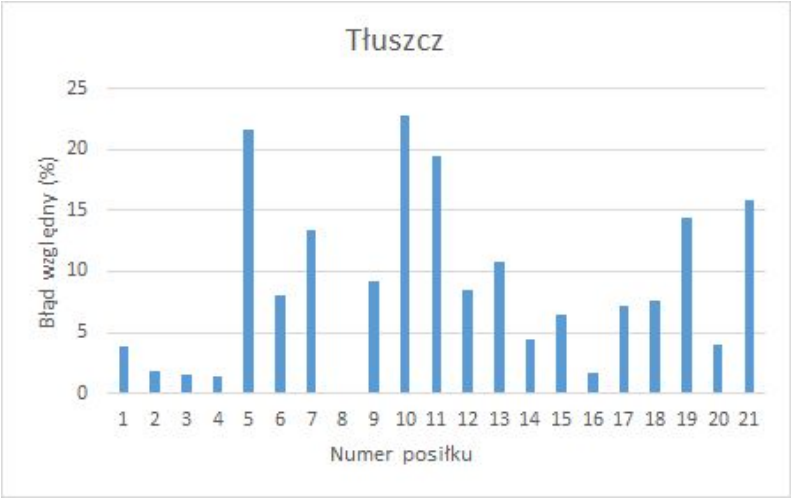
Nazwa	Wartość (w gramach)	Waga (jako ważność)
Kalorie	2000	1
Białko	70	1
Tłuszcz	120	1
Węglowodany	230	1

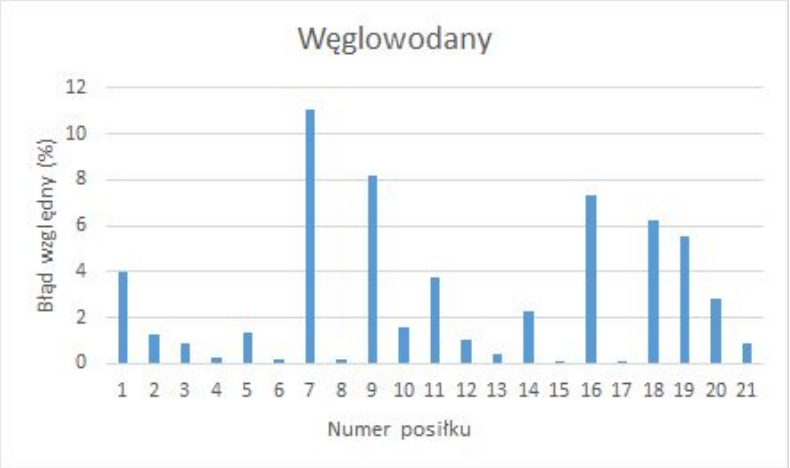
```
Udział konkretnych posiłków:
      Śniadanie:   30%
      Obiad:       50%
      Kolacja:          20%
```

Uzyskane wyniki:


Kalorie
Błąd względny (%)
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
Numer posiłku


Białko
Błąd względny (%)
120
100
80
60
40
20
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
Numer posiłku


Tłuszcz
Błąd względny (%)
25
20
15
10
5
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
Numer posiłku




```
Średnia wartość funkcji celu: 2.31
Liczba produktów unikalnych: 16
Liczba produktów nieunikalnych: 22
Średni czas pomiędzy podaniem tego samego produktu: 4.236
Minimalny czas pomiędzy podaniem tego samego produktu: 2
```

Interpretacja wyników:

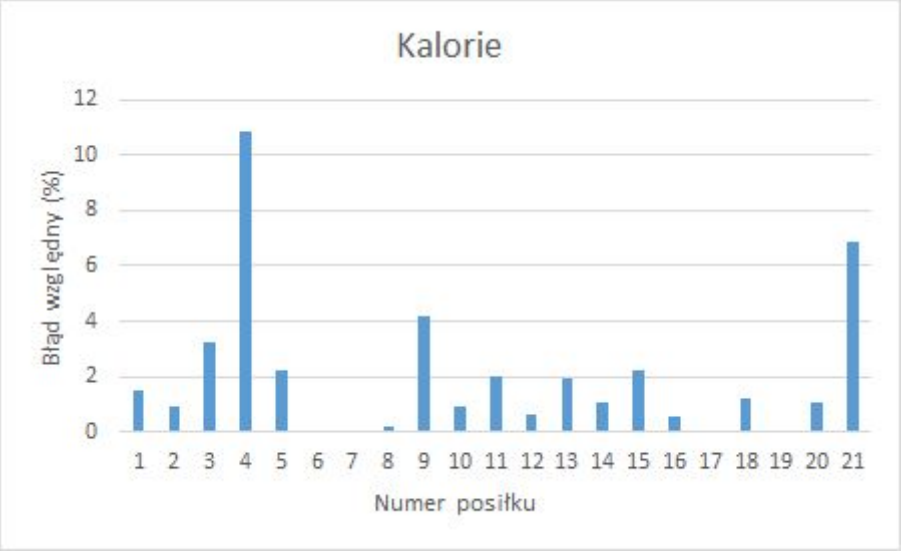
Algorytm znalazł dość rozsądne rozwiązanie problemu o czym świadczy średnia wartość funkcji celu bliska zera. Oznacza to, że sumaryczne odchyły są równe średnio 2 przy wartościach zapotrzebowania rzędu kilkuset.
W ciągu tygodnia, aż 42% produktów zostało wykorzystanych tylko raz, a w przypadku produktów wykorzystanych wielokrotnie, średni czas między ich podaniem wynosił około 4 - oznacza to, że w przypadku produktów śniadaniowych/obiadowych/kolacyjnych wykorzystywane było one z odstępem około 4 dni, a w przypadku produktów ogólnych, średnio co drugi dzień.
Na zamieszczonych wykresach widać, że rozwiązania są zoptymalizowane znacznie lepiej pod kątem ilości kalorii niż pozostałych wartości odżywczych (błędy względne jeżeli chodzi o kalorie są rzędu 1% gdzie w przypadku pozostałych wartości kilanaście/kilkadziesiąt razy większe). Wynika to z zastosowanych w przykładzie wag. Wszystkie 4 wartości odżywcze posiadają równe wagi i odchyły bezwzględne są dla różnych wartości odżywczych traktowane identycznie - w efekcie dla algorytmu równie dobrą sytuacja jest odchylenie o 5 od wartości kalorii oraz dokładne osiągnięcie wartości tłuszczu jak i sytuacja odwrotna (tzn. dokładne kalorie oraz niedokładny tłuszcz) - jeżeli chodzi o błędy względne, wartość 5 dla kalorii, na które zapotrzebowanie w ramach jednego posiłku wynosi od 400 do 1000 jest mała, a wartość błędu względnego dla odchyłu równego 5 dla tłuszczu przy zapotrzebowaniu od 24 do 60 jest znacznie bardziej zauważalna.

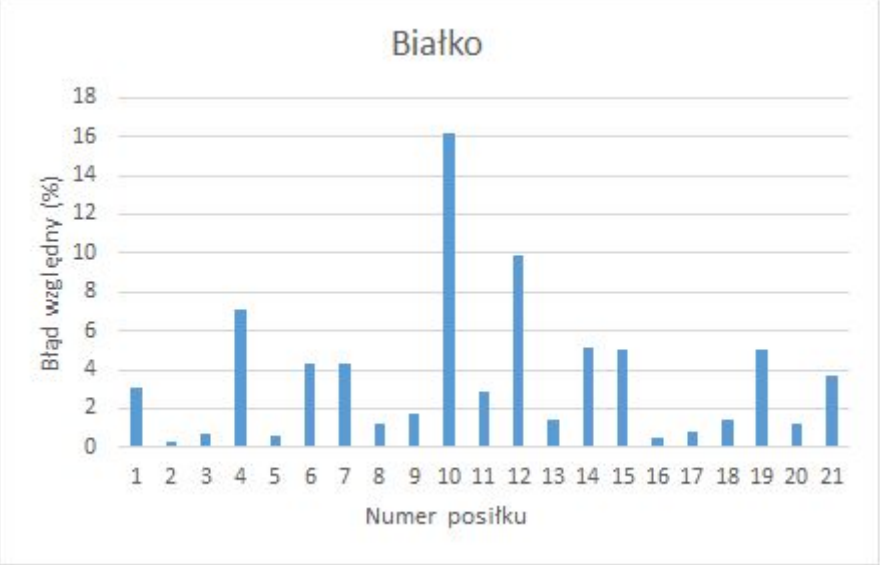
Eksperyment 2 - zwiększenie wag wartości odżywczych

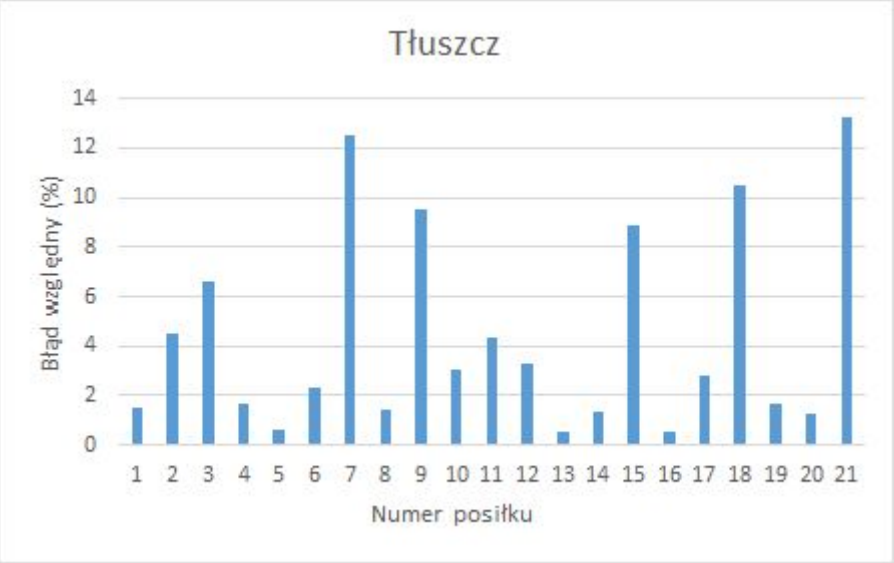
```
Zmiana w stosunku do eksperymentu 1
Pożądane wartości odżywcze w posiłku:
```

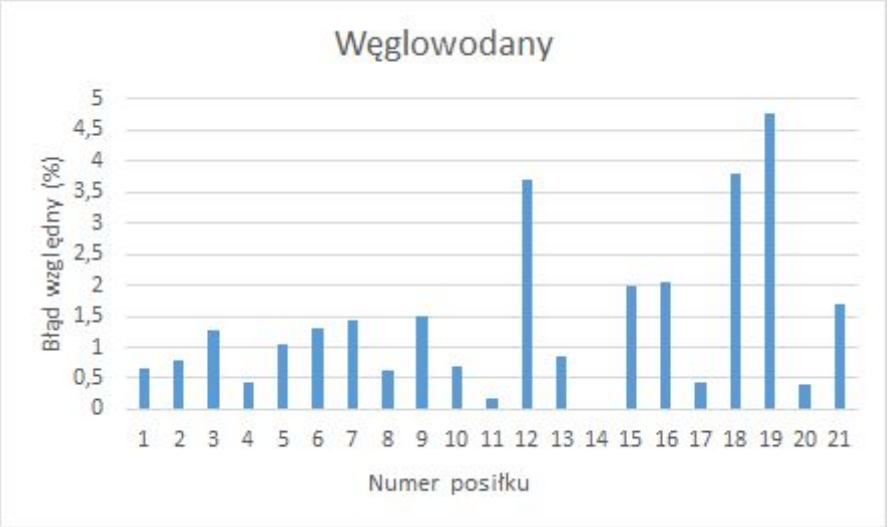
Nazwa	Wartość (w gramach)	Waga (jako ważność)
Kalorie	2000	1
Białko	70	10
Tłuszcz	120	10
Węglowodany	230	10

Uzyskane wyniki:










```
Średnia wartość funkcji celu: 1.38
Liczba produktów unikalnych: 8
Liczba produktów nieunikalnych: 25
Średni czas pomiędzy podaniem tego samego produktu: 3.531
Minimalny czas pomiędzy podaniem tego samego produktu: 2
```

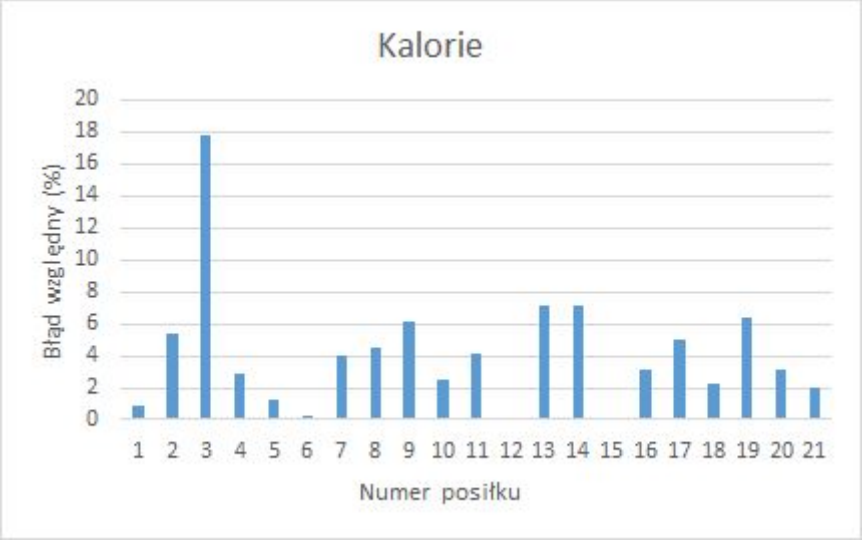
Interpretacja wyników:

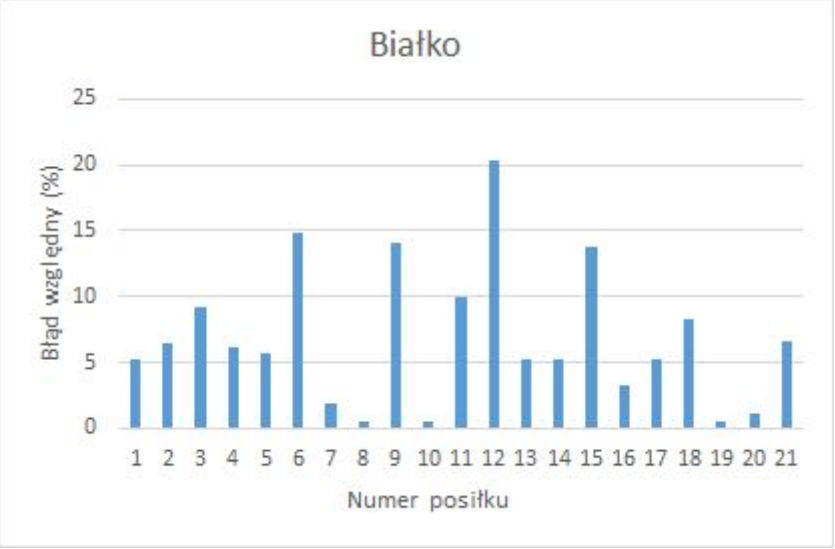
Zmiana wag nie zmienia znacząco parametrów opisujących rozwiązanie. Zmieniły się natomiast błędy względne dotyczące spełnienia zapotrzebowania na konkretne wartości odżywcze dla konkretnych posiłków. Skale wszystkich 4 wykresów w tym przypadku są znacznie bardziej wyrównane i konkretne wartości w nich prezentowane są w większości mniejsze od 10. Poprawiło się spełnienie zapotrzebowania na tłuszcz, białko i węglowodany, jednakże znacznie pogorszyło się dla kalorii.

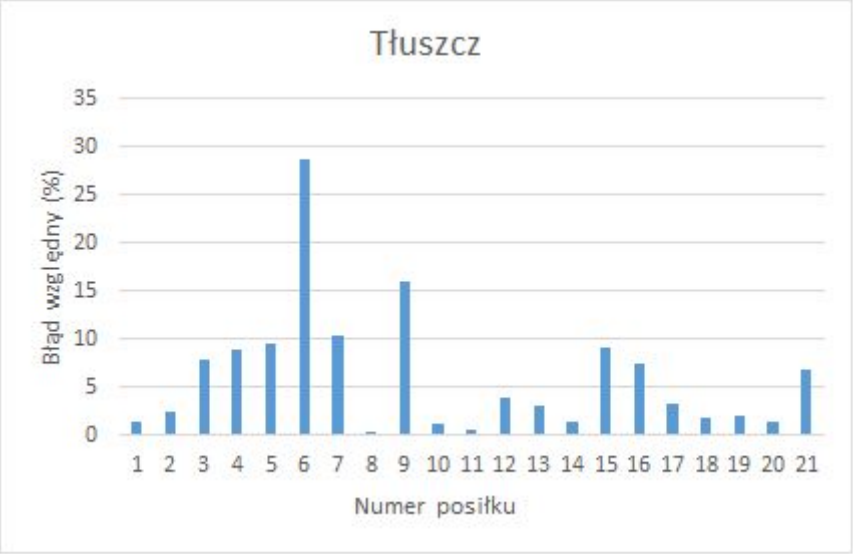
Eksperyment 3 - zmniejszenie liczby iteracji

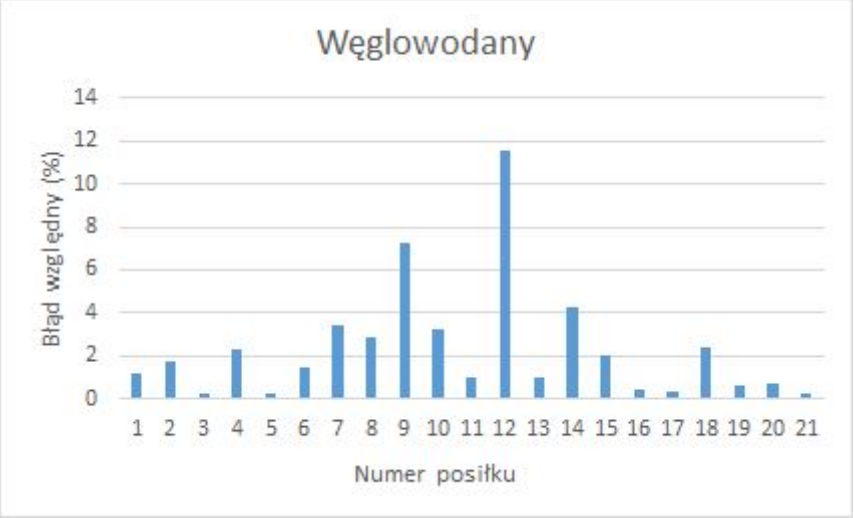
```
Zmiana w stosunku do eksperymentu 2:
Liczba iteracji: 25
```

Uzyskane wyniki:










```
Średnia wartość funkcji celu: 2.50576
Liczba produktów unikalnych: 12
Liczba produktów nieunikalnych: 21
Średni czas pomiędzy podaniem tego samego produktu: 3.44
Minimalny czas pomiędzy podaniem tego samego produktu: 2
```

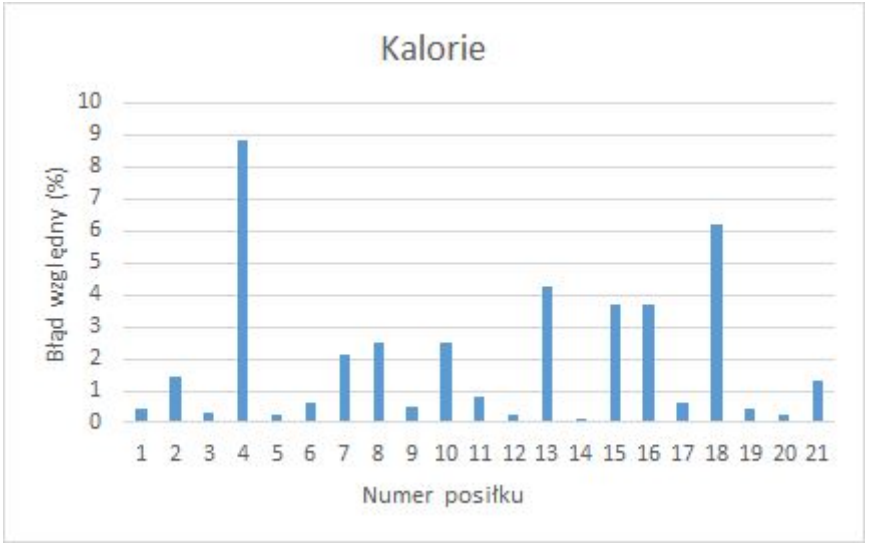
Interpretacja wyników:

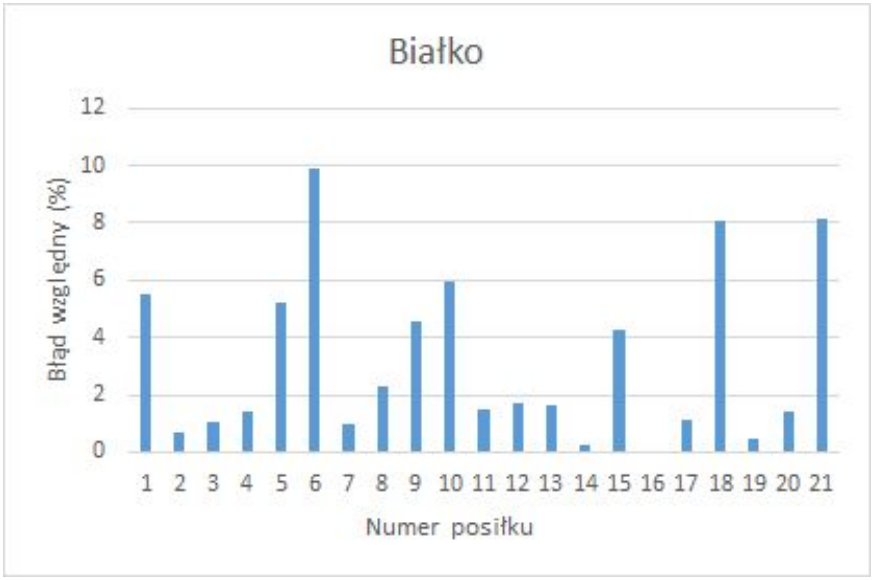
Zmniejszenie liczby iteracji nie wpłynęło bardzo znacząco na otrzymane wyniki dla tych danych wejściowych. Zwiększył się rozrzut błędów względnych dla poszczególnych wartości odżywczych dla poszczególnych posiłków.

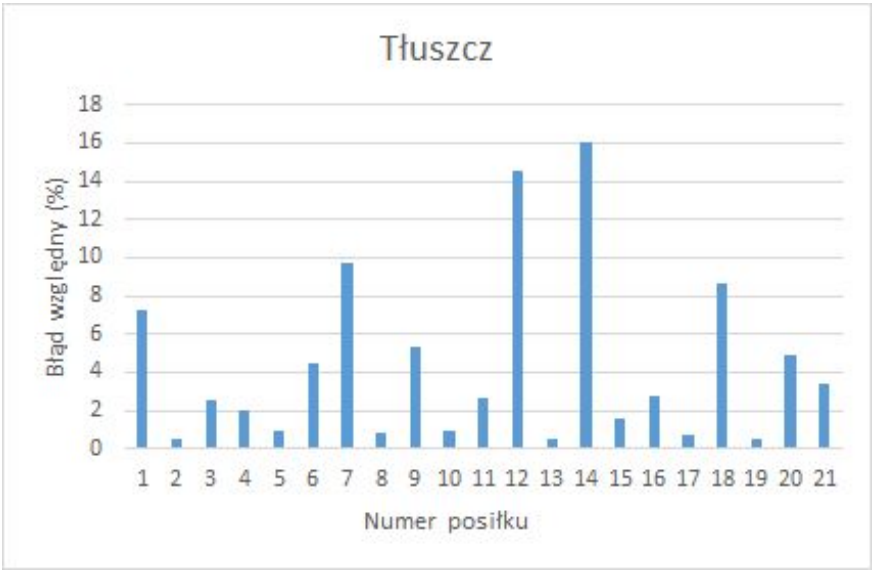
Eksperyment 4 - zmniejszenie liczby elementów elity

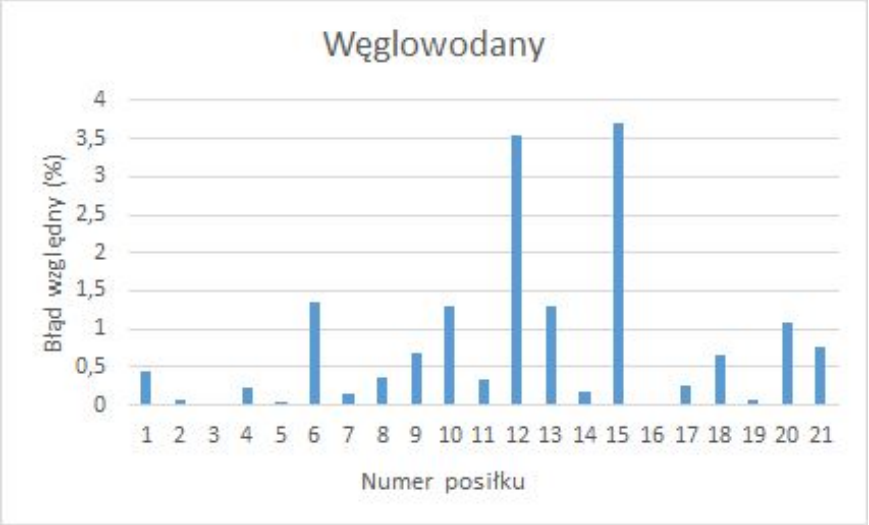
```
Zmiana w stosunku do eksperymentu 2:
Liczebność elity: 4
```

Uzyskane wyniki:









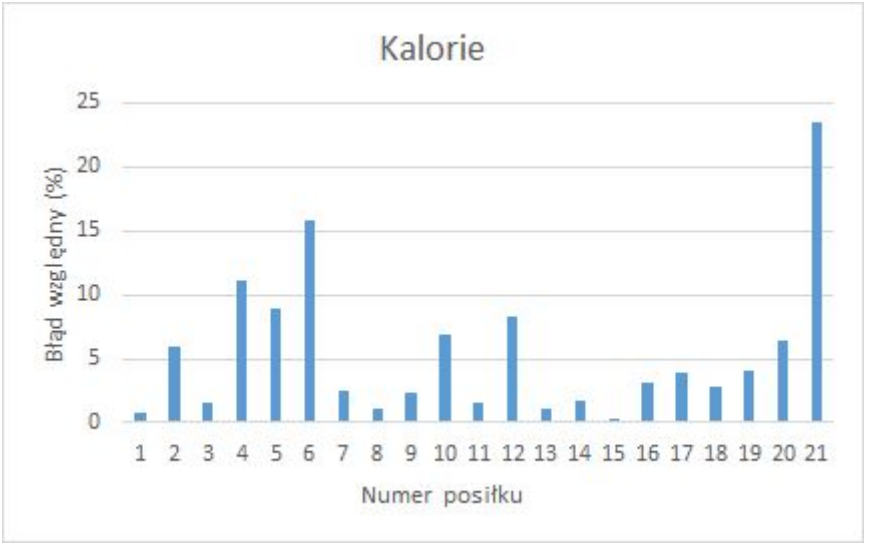
Średnia wartość funkcji celu: 1.2623
Liczba produktów unikalnych: 15
Liczba produktów nieunikalnych: 24
Średni czas pomiędzy podaniem tego samego produktu: 3.8598
Minimalny czas pomiędzy podaniem tego samego produktu: 2

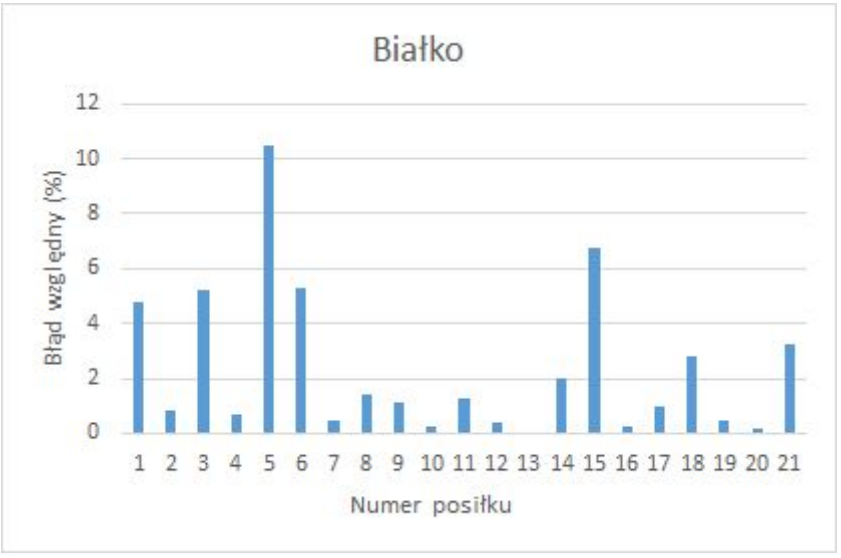
Eksperyment 5 - zmiana wag wartości odżywczych

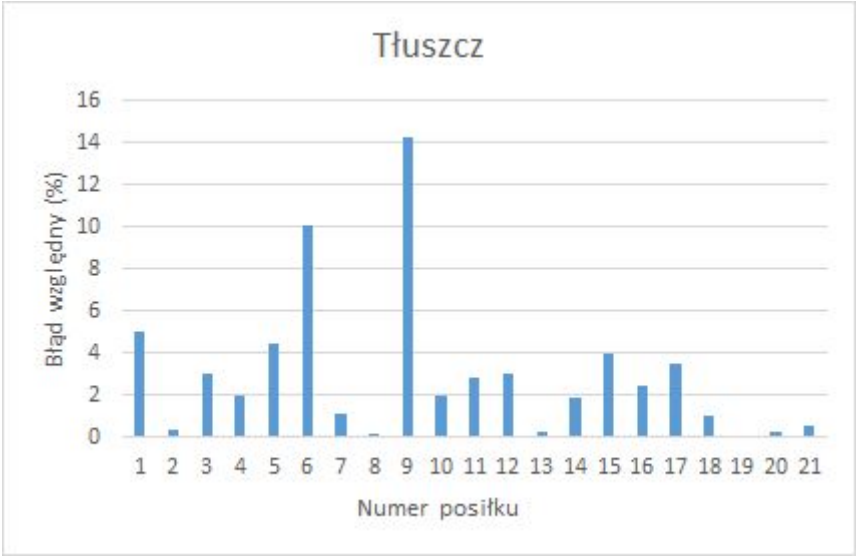
Zmiana w stosunku do eksperymentu 1
Pożądane wartości odżywcze w posiłku:

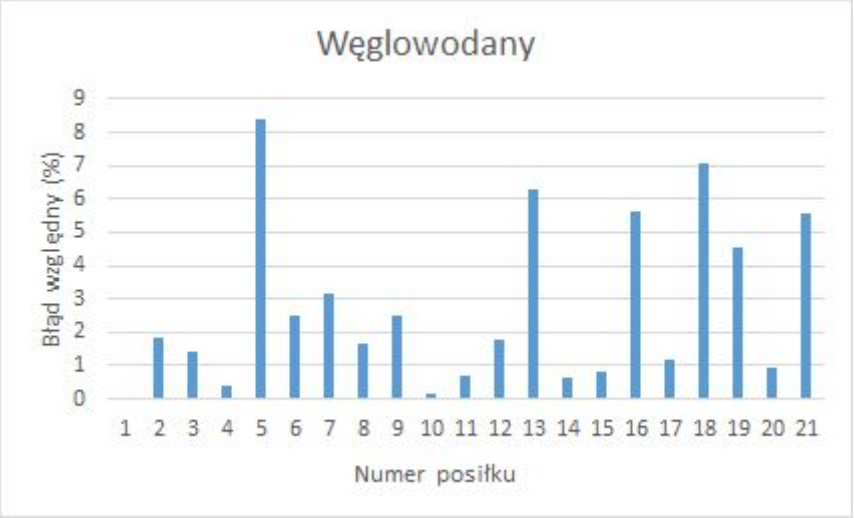
Nazwa	Wartość (w gramach)	Waga (jako ważność)
Kalorie	2000	1
Białko	70	200/7
Tłuszcz	120	50/3
Węglowodany	230	200/23

Uzyskane wyniki:










```
Średnia wartość funkcji celu: 1.4594
Liczba produktów unikalnych: 17
Liczba produktów nieunikalnych: 18
Średni czas pomiędzy podaniem tego samego produktu: 3.938
Minimalny czas pomiędzy podaniem tego samego produktu: 2
```

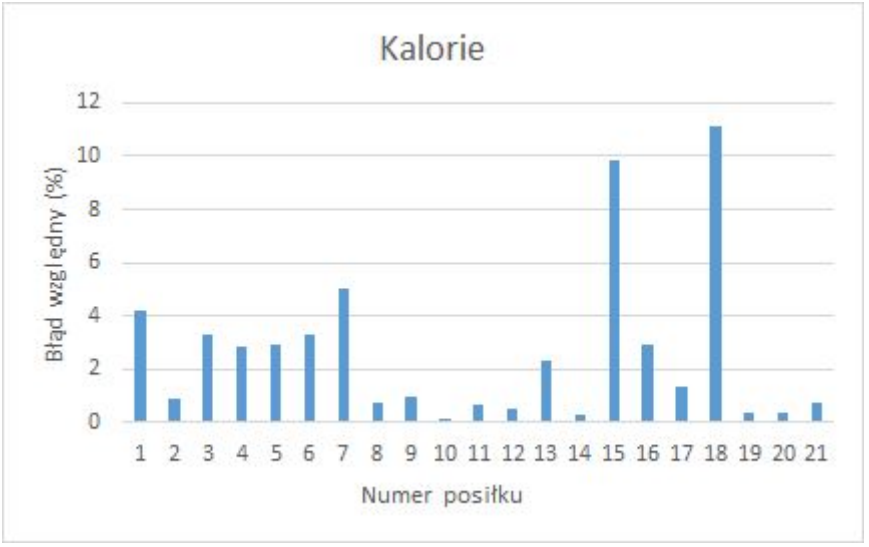
Interpretacja wyników:

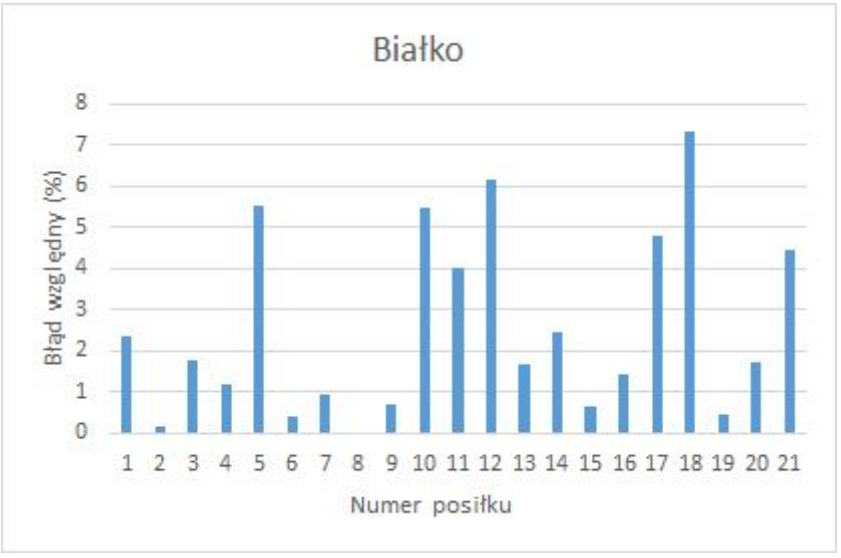
Wagi w niniejszym przykładzie zostały dobrane tak, aby możliwie najrówniej optymalizować każdy z parametrów. Wyniki są ewidentnie bliższe tego niż w eksperymencie 1. Efekt nie jest jednak idealny z powodu dyskretności wag dostępnych posiłków oraz natury metaheurstyk które nie zawsze zwracają optymalne rozwiązania.

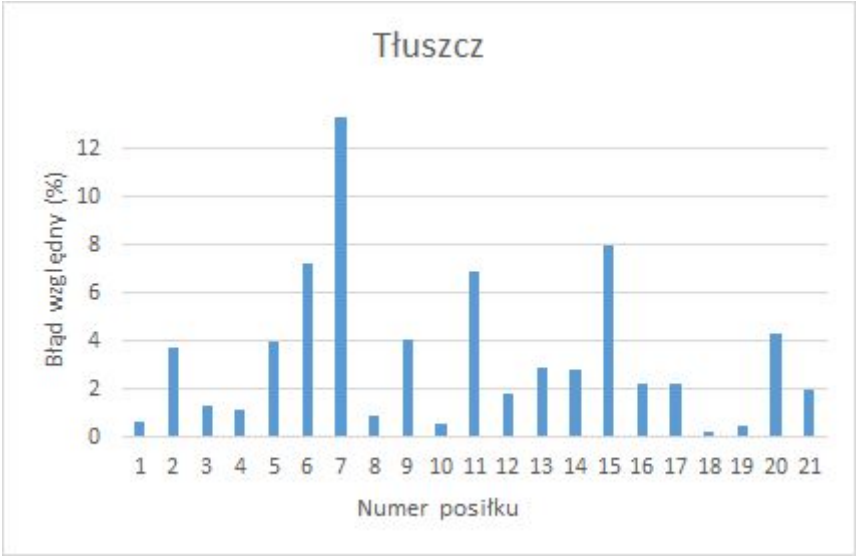
Eksperyment 6 - wyłączenie krzyżowania

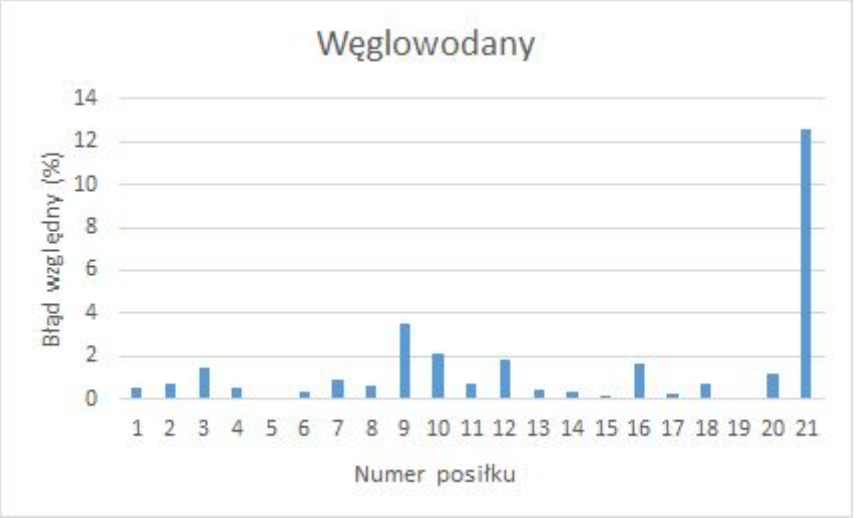
```
Zmiana w stosunku do eksperymentu 2:
Prawdopodobieństwo krzyżowania: 0
```

Uzyskane wyniki:










```
Średnia wartość funkcji celu: 1.3775
Liczba produktów unikalnych: 13
Liczba produktów nieunikalnych: 23
Średni czas pomiędzy podaniem tego samego produktu: 3.81
Minimalny czas pomiędzy podaniem tego samego produktu: 1
```

Interpretacja wyników:

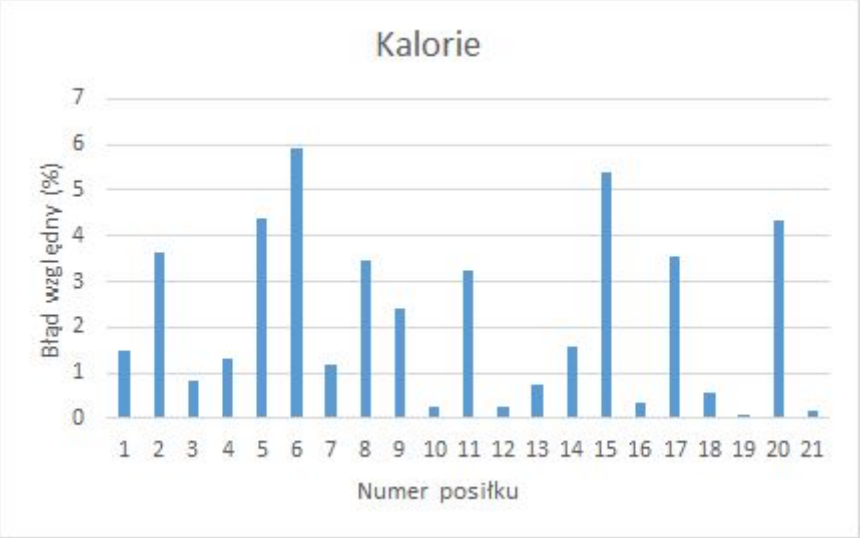
Dla tak dużej liczby iteracji stosowanie krzyżowania z prawdopodobieństwem jak w eksperymencie 1 nie wpływa znacząco na jakość otrzymywanych wyników czego potwierdzeniem są wyniki tego eksperymentu.

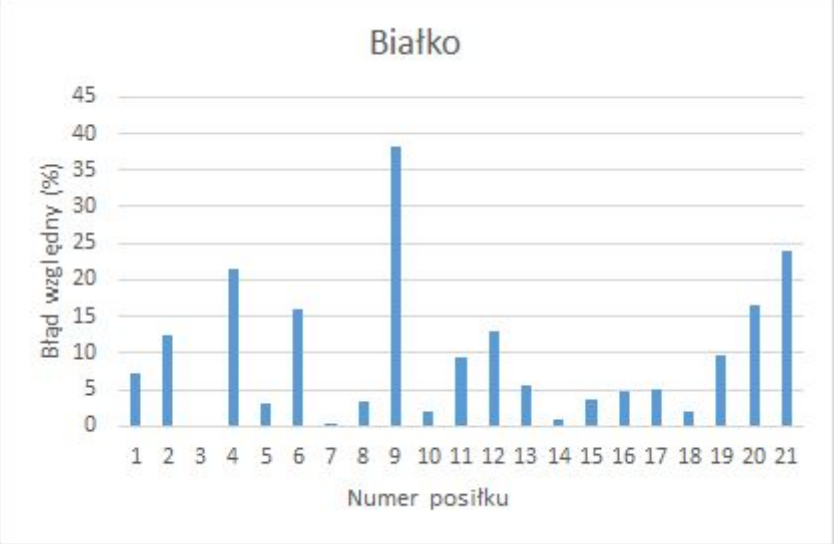
Następne eksperymenty wykonane są z liczbą iteracji 25, aby bardziej uwypuklić różnice płynące ze zmiany parametrów.

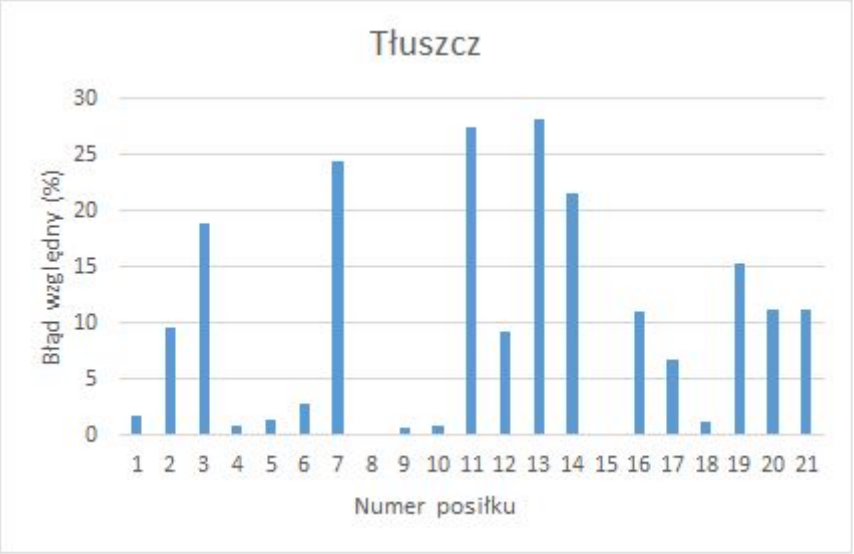
Eksperyment 7 - zmniejszenie liczby elementów populacji

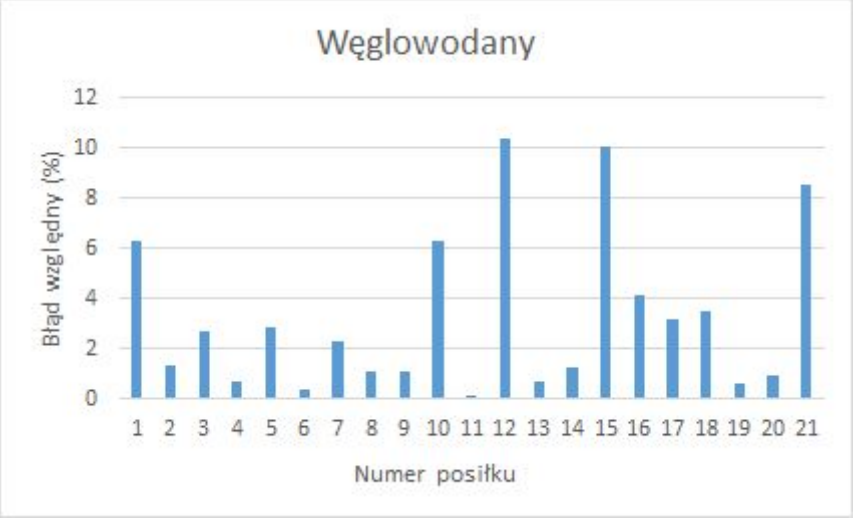
```
Zmiana w stosunku do eksperymentu 3:
Liczebność elity: 5
Liczebność populacji: 15
```

Uzyskane wyniki:










```
Średnia wartość funkcji celu: 3.203
Liczba produktów unikalnych: 15
Liczba produktów nieunikalnych: 22
Średni czas pomiędzy podaniem tego samego produktu: 4.029
Minimalny czas pomiędzy podaniem tego samego produktu: 1
```

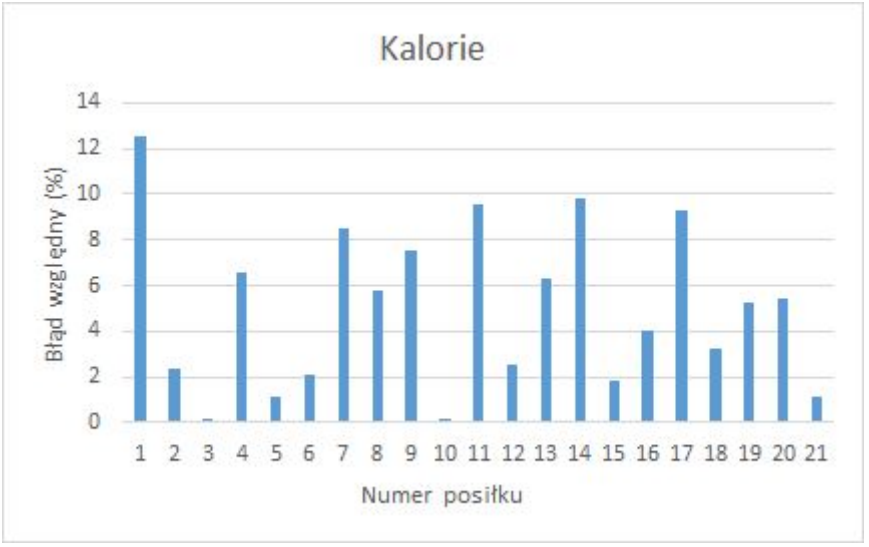
Interpretacja wyników:

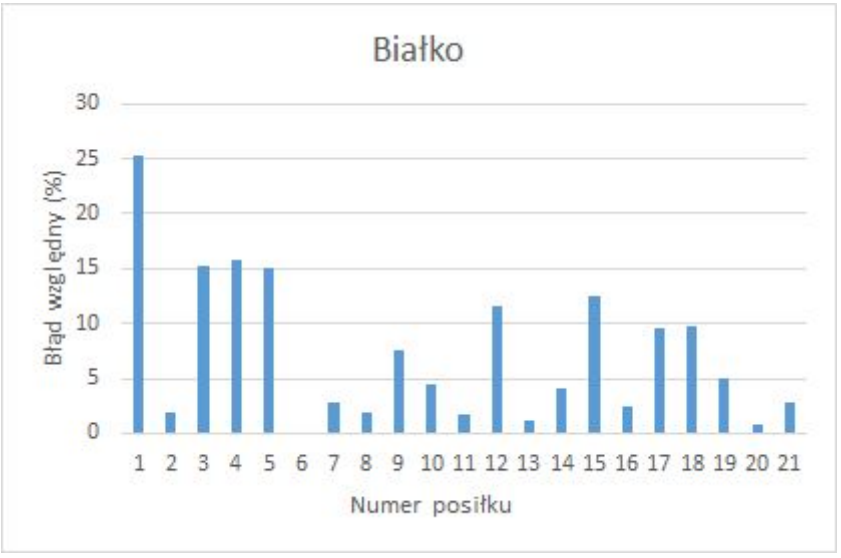
Zmniejszenie liczebności populacji oraz elity negatywnie wpływa na jakość wyników - pojawiają się błędy względne o wartościach kilkudziesięciu procent. Mimo to, algorytm ciągle daje rozwiązanie o niskiej średniej wartości funkcji celu.

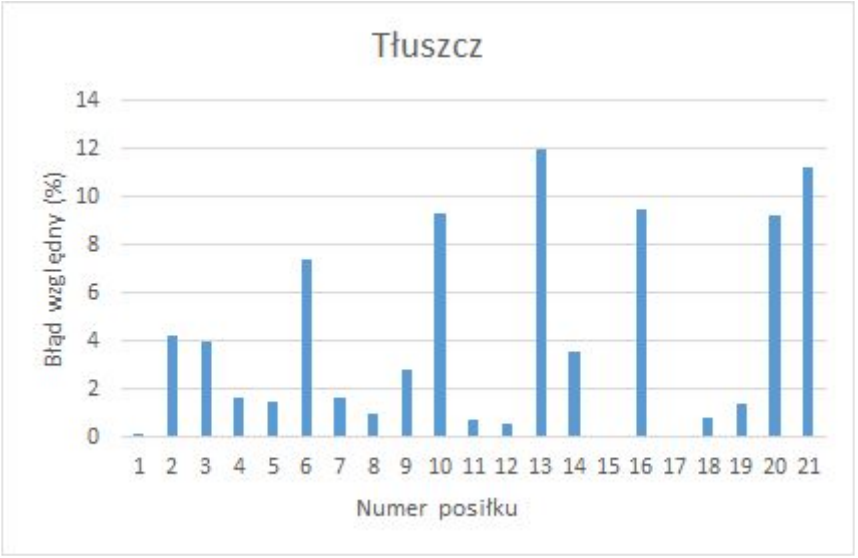
Eksperyment 8 - pewność krzyżowania

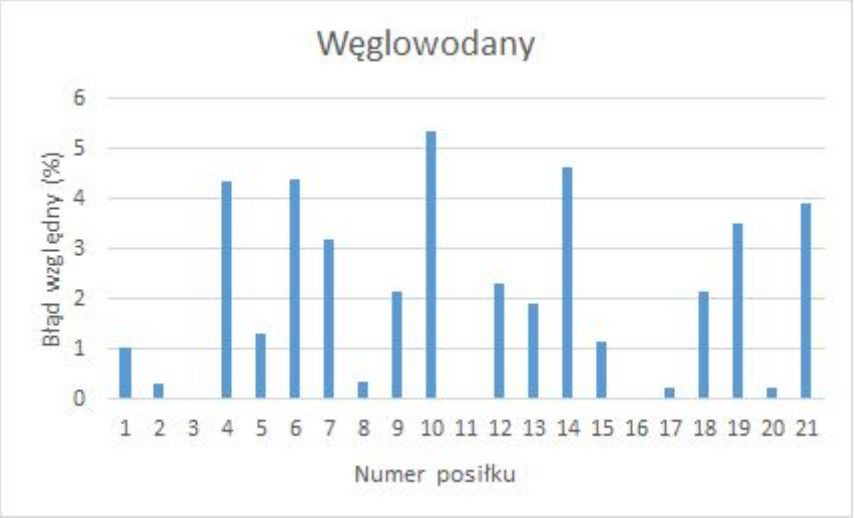
```
Zmiana w stosunku do eksperymentu 3:
Prawdopodobieństwo krzyżowania: 100
```

Uzyskane wyniki:









Średnia wartość funkcji celu: 2.58
Liczba produktów unikalnych: 16
Liczba produktów nieunikalnych: 21
Średni czas pomiędzy podaniem tego samego produktu: 4.33
Minimalny czas pomiędzy podaniem tego samego produktu: 1

Interpretacja wyników:

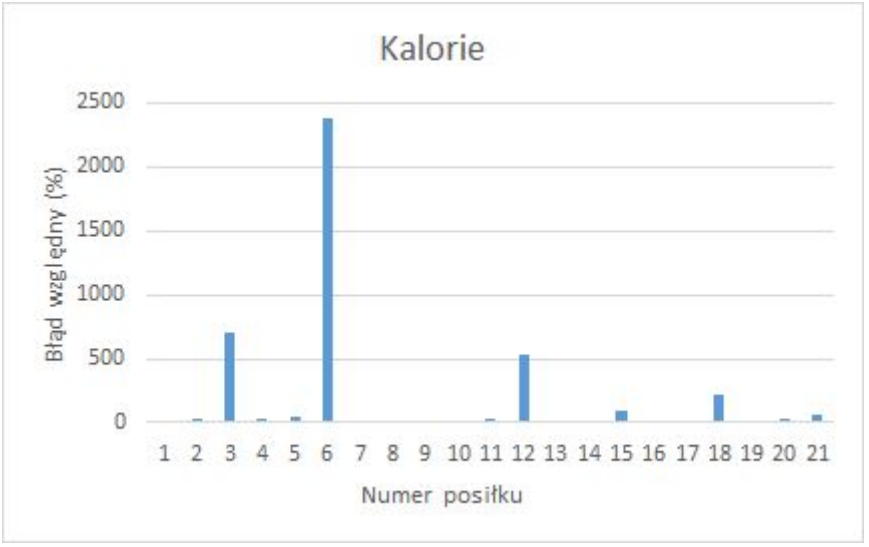
Pewność krzyżowania nie przynosi poprawy jakości dla użytych danych i parametrów.

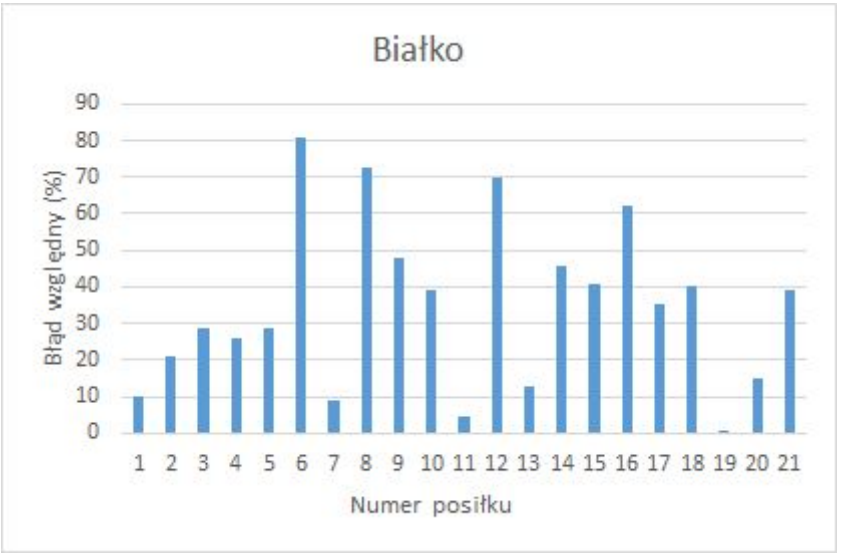
Eksperyment 9 - zwiększenie wagi jednej wartości odżywczej (białka)

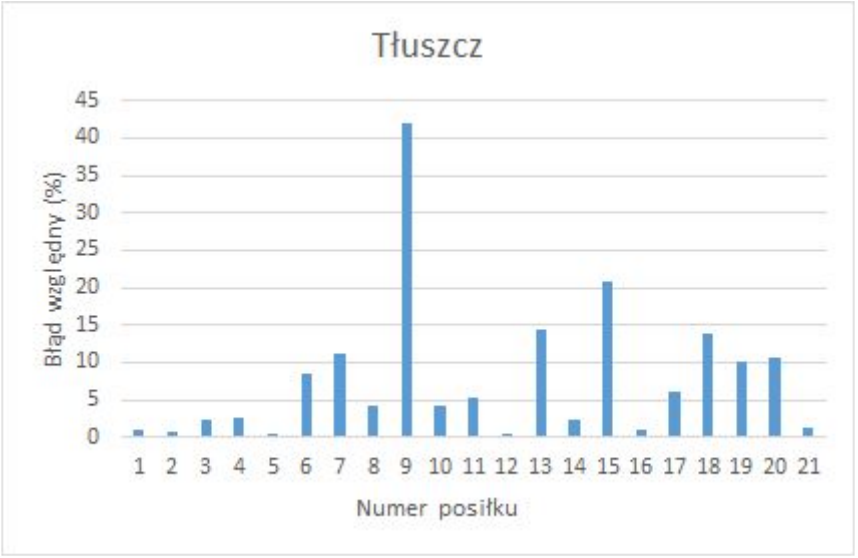
Zmiana w stosunku do eksperymentu 2
Pożądane wartości odżywcze w posiłku:

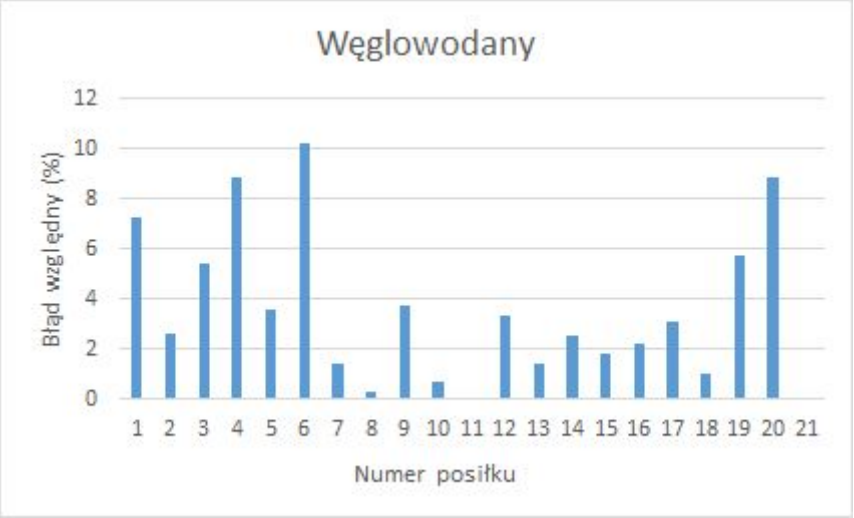
Nazwa	Wartość (w gramach)	Waga (jako ważność)
Kalorie	2000	1
Białko	300	10
Tłuszcz	120	10
Węglowodany	230	10

Uzyskane wyniki:










```
Średnia wartość funkcji celu: 20.94
Liczba produktów unikalnych: 10
Liczba produktów nieunikalnych: 19
Średni czas pomiędzy podaniem tego samego produktu: 3.3069
Minimalny czas pomiędzy podaniem tego samego produktu: 1
```

Interpretacja wyników:

Zwiększenie znaczenia białka w diecie poskutkowało znacznym pogorszeniem średniej wartości funkcji celu oraz zwiększeniem błędów względnych. Wynika to ze struktury produktów - główne źródło białka w diecie, kotlet schabowy, ma bardzo dużą rozdzielczość wagową oraz znaczną liczbę kalorii. W efekcie jest mało elastyczny i algorytm ma znacznie trudniejsze zadanie do rozwiązania. Bardzo ciekawie wygląda wykres błędów względnych w spełnianiu zapotrzebowania na kalorie. W posiłkach 3, 6, 12 i 18 błędy osiągają gigantyczne wartości - jest to spowodowane zaproponowaniem dużej liczby schabowych na obiad, tak by spełnić zapotrzebowanie na białko. Powoduje to, że obiad jest bardzo kaloryczny i zapotrzebowanie na kalorie na kolację jest bardzo niskie - rzędu paru, parunastu kalorii. W efekcie, algorytm żeby spełnić pozostałe wymagania dodaje produkty, zwiększa się znacznie liczba kalorii w kolacji i w porównaniu z zapotrzebowaniem jest kuriozalnie duża - np. 25 razy za duża.
Jeżeli chodzi o monotonność to sytuacja wygląda podobnie jak w poprzednich eksperymentach.

5. Wnioski

Zaproponowany algorytm dla wykorzystanych w eksperymentach danych w większości wyników proponuje dość rozsądne diety zarówno pod względem spełnienia zapotrzebowania na wartości odżywcze jak również nie zbyt monotonne - znaczna część produktów się nie powtarza w ramach diety, a powtarzające nie są wykorzystywane nagminnie. Wygenerowanie diety dla około 50 produktów na 7 dni trwa kilkanaście sekund.

Kod projektu oraz wykorzystane dane testowe znajdują się na repozytorium: https://github.com/amalkows/ALHE-projekt