

**DOKUMENTACJA PROJEKTU**

PRZEDMIOT Badania operacyjne i matematyczne wspomaganie decyzji

TEMAT PROJEKTU Przydział zadań dla pracowników z wykorzystaniem algorytmu tabu search

AUTORZY Dawid Kasieczka

Marek Piekarz

Automatyka i Robotyka, WEAIiIB

Kraków, styczeń 2018r.

Spis treści

[1. Wstęp 2](#_Toc503736886)

[1.1. Opis projektu 2](#_Toc503736887)

[1.2. Model matematyczny 2](#_Toc503736888)

[2. Algorytm tabu search 2](#_Toc503736889)

[2.1 Opis algorytmu 2](#_Toc503736890)

[2.2. Postać danych wejściowych 3](#_Toc503736891)

[2.3. Postać rozwiązania 4](#_Toc503736892)

[2.4. Definicja sąsiedztwa 6](#_Toc503736893)

[3. Testy 6](#_Toc503736894)

[3.1. Sposób przeprowadzania testów 6](#_Toc503736895)

[3.2. Zestaw I 7](#_Toc503736896)

[3.2.1. Struktura danych 7](#_Toc503736897)

[3.2.2. Test czasu 8](#_Toc503736898)

[3.2.3. Test kilku przebiegów 8](#_Toc503736899)

[3.2.4. Test sąsiedztwa 9](#_Toc503736900)

[3.2.5. Test rozmiaru sąsiedztwa 13](#_Toc503736901)

[3.2.6. Test rozmiaru listy tabu 14](#_Toc503736902)

[3.3. Zestaw II 15](#_Toc503736903)

[3.3.1. Struktura danych 15](#_Toc503736904)

[3.3.2. Test czasu 15](#_Toc503736905)

[3.3.3. Test kilku przebiegów 15](#_Toc503736906)

[3.3.4. Test rozmiaru sąsiedztwa 16](#_Toc503736907)

[3.3.5. Test rozmiaru listy tabu 16](#_Toc503736908)

[4. Wnioski 17](#_Toc503736909)

# 1. Wstęp

## 1.1. Opis projektu

Wybrany problem opisuje zagadnienie przydziału zadań dla grupy pracowników. Polega to na tym, że każda osoba otrzymuje serię przydzielonych jej prac, uwzględniając czas jaki potrzebuje na rozwiązanie każdego problemu. Przyjęto też, że możliwe jest łączenie pracowników w zespoły kilkuosobowe, zaznaczając że praca nad jednym zadaniem musi się rozpocząć i zakończyć w tym samym momencie dla każdego pracownika z zespołu. Do zaimplementowania wyżej wymienionego problemu wykorzystano algorytm tabu search. Program został napisany w języku C++.

## 1.2. Model matematyczny

Funkcja celu wyrażona jest następującym wzorem:

***N*** liczba zadań;

***gi = {0,1}*** informacja czy zadanie zostało wykonane w zamierzonym czasie;

***ti***  różnica czasu rzeczywistego zakończenia zadania, a planowanego, w przypadku wykonania w odpowiednim terminie ti = 0;

***pi­*** jednorazowa kara za niedotrzymanie terminu;

***p\_houri­***kara przyznawana za każdą jednostkę czasu powyżej przyjętego limitu;

Określone powyżej zadanie polega na minimalizacji funkcji celu. Funkcja ta wyraża jedynie karę za niewywiązanie się z przyjętych terminów, w związku z tym za wykonanie zadania w zamierzonym czasie nie jest naliczany żaden zysk.

# 2. Algorytm tabu search

## 2.1 Opis algorytmu

Algorytm realizuje poszukiwanie rozwiązania problemu, dla którego rozwiązanie dokładne jest z różnych powodów niemożliwe. W przeciwieństwie do algorytmu optymalizacji lokalnej, posiada on zachowanie pozwalające na wydostanie się z minimum lokalnego. Wykorzystuje w tym celu listę zabronień. Polega to na tym, że algorytm zapamiętuje sekwencję ostatnich „ruchów” rozwiązania i nie pozwala na ich odwrócenie (powrót do poprzedniego rozwiązania).

Jeśli jednak ruch będący na liście zabronień jest bardzo dobry dla funkcji celu, to zadziała kryterium aspiracji, które zignoruje listę tabu i wybierze to zabronione rozwiązanie. Kryterium aspiracji w tym konkretnym przypadku zadziała wtedy i tylko wtedy gdy, rozwiązanie będzie lepsze niż najlepsze dotychczas znalezione.

Lista zabronień jest tak zwaną pamięcią ktrótkoterminową, stworzoną jako kolejka FIFO, o ustalonym rozmiarze. Dodatkowym rodzajem pamięci jest pamięć średnioterminowa, która przechowuje ścieżkę rozwiązań o różnych wartościach funkcji celu. W przypadku gdy algorytm obierze nieefektywną drogę zostanie on „nawrócony” na jedno z rozwiązań z tej grupy. Pozwala to na zaoszczędzenie czasu w niektórych przypadkach.

Ze względu na fakt że nie jest to algorytm dokładny rozwiązanie optymalne nie jest gwarantowane.

## 2.2. Postać danych wejściowych

Dane zadanie określone jest przez zestaw wartości opisujących potrzebną ilość siły roboczej do zakończenia pracy. Przyjęto cztery dziedziny, zatem każdy problem skojarzony jest z wektorem czteroelementowym. Z kolei dla jednego pracownika określono zestaw pięciu wartości. Pierwsze cztery informują o wydajności, czyli ilości pracy jaką pracownik może wykonać w danej dziedzinie w jednej jednostce czasu. Odpowiadają one parametrom określonym dla problemów. Natomiast ostatni parametr dla pracownika to umiejętność pracy grupowej wyrażona w procentach. W momencie łączenia pracowników w zespół wydajność każdego z pracowników mnożona jest przez wartość pracy grupowej, a otrzymane wyniki są dodawane w ramach dziedzin ze sobą, otrzymując parametry wydajności zespołu.

Do każdego z problemów przyporządkowane są też następujące parametry, kara za niedotrzymanie terminu przyznawana jednorazowo, oraz kara wyrażona w kwocie "na" jednostkę czasu otrzymywana za przeciągnięcie maksymalnego terminu wykonania zadania. Ostatnim parametrem związanym z problemem jest deadline.

## 2.3. Postać rozwiązania

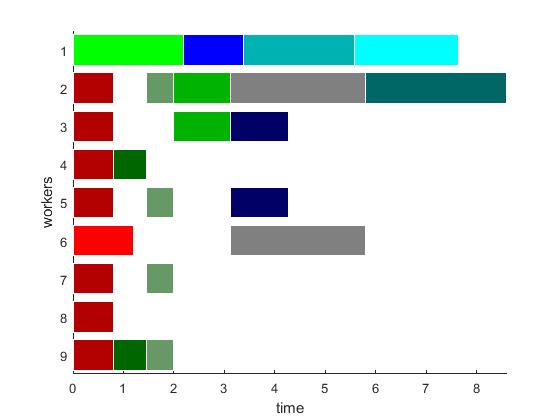
Dane otrzymane z przebiegu algorytmu podane są w postaci macierzy, z której można odczytać jakie zadania zostały przydzielone danemu pracownikowi, oraz jaki czas jest na nie poświęcony. Przykładowa macierz rozwiązania podana poniżej.

Tabela 1. Przykładowe rozwiąznie

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2.19008 | 1.18968 | 2.19008 | 2.05794 | 0 | 0 |
| 0.798073 | 0.650929 | 0.545788 | 1.12499 | 2.678 | 2.8 |
| 0.798073 | 1.19672 | 1.12499 | 1.15133 | 0 | 0 |
| 0.798073 | 0.650929 | 0 | 0 | 0 | 0 |
| 0.798073 | 0.650929 | 0.545788 | 1.12499 | 1.15133 | 0 |
| 1.18968 | 1.9301 | 2.678 | 0 | 0 | 0 |
| 0.798073 | 0.650929 | 0.545788 | 0 | 0 | 0 |
| 0.798073 | 0 | 0 | 0 | 0 | 0 |
| 0.798073 | 0.650929 | 0.545788 | 0 | 0 | 0 |
| 4 | 7 | 10 | 12 | 0 | 0 |
| 2 | 0 | 3 | 5 | 8 | 11 |
| 2 | 0 | 5 | 9 | 0 | 0 |
| 2 | 6 | 0 | 0 | 0 | 0 |
| 2 | 0 | 3 | 0 | 9 | 0 |
| 1 | 0 | 8 | 0 | 0 | 0 |
| 2 | 0 | 3 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 |
| 2 | 6 | 3 | 0 | 0 | 0 |

Liczba kolumn powyższej macierzy oznacza maksymalną liczbę zadań przydzielonych pracownikowi. Liczba wierszy to podwojona ilość pracowników. Górna połowa macierzy odpowiada za czas, natomiast dolna za kolejność zadań. Zerami zostały oznaczone przerwy w pracy pracownika, lub zakończenie jego pracy, w zależności od tego, czy na dalszych pozycjach w wierszu występuje wartość różna od zera.

Z wykorzystaniem środowiska Matlab dane wyjściowe algorytmu zostały zwizualizowane do postaci diagramu Gantta. Taka forma graficzna jest najlepsza do pokazania rozplanowania projektów w czasie. Przykładowe rozwiązanie pokazane zostało na Rys. 1.



Rys.1. Przykładowe rozwiązanie

Na powyższym przykładzie można zauważyć, że jeden kolor odpowiada jednemu problemowi, a białe fragmenty wykresu mówią o bezczynności pracownika w określonym czasie. Widać też, że niektóre z zadań zostały rozdzielone na wielu pracowników, którzy na czas rozwiązywania danego problemu tworzą zespół.

## 2.4. Definicja sąsiedztwa

W programie zdefiniowano sąsiedztwo w następujący sposób:

**ADD –** Rozwiązanie sąsiednie różni się od obecnego dodaniem kilku niepowtarzających się zadań do różnych pracowników.

**REMOVE -** Rozwiązanie sąsiednie różni się od obecnego zabraniem kilku zadań do różnych pracowników.

**ADD\_AND\_REMOVE –** Jest połączeniem dwóch poprzednich sąsiedztw. Oznacza jednoczesne dodanie oraz usunięcie kilku zadań dla pracowników w rozwiązaniu.

**SWAP –** Pracownik X wykonuje zadania pracownika Y, natomiast pracownik Y zadania pracownika X. Jest to równoznaczne w zamianie dwóch wierszy miejscami w rozwiązaniu.

**TAKE –** Polega na zabraniu elementu z wiersza X i rozszerzeniu wiersza Y o ten element. Odpowiada zabraniu zadania od jednego pracownika i przydzieleniu innemu.

**TO\_LAZY –** Jedyne sąsiedztwo deterministyczne. Polega na znalezieniu pracownika z największą ilością zadań i przeniesienie na pracownika który ma ich najmniej.

**ALL –** Suma wszystkich powyższych. Wybór następuje przez losowanie jednego z sąsiedztw z odpowiednimi współczynnikami.

# 3. Testy

## 3.1. Sposób przeprowadzania testów

Dla określonych danych zostały wykonane testy, mające na celu sprawdzenie wpływu różnych parametrów na wartość funkcji celu. Przyjęto dwa zestawy danych, w poniższej tabeli znajdują się konfiguracje obu z tych zestawów.

Tabela 2. Konfiguracja zestawów testowych

|  |  |  |
| --- | --- | --- |
| Parametr | Zestaw I | Zestaw II |
| Liczba pracowników | 10 | 100 |
| Liczba umiejętności pracownika | 4 | 4 |
| Liczba problemów | 15 | 200 |

## 3.2. Zestaw I

### 3.2.1. Struktura danych

W poniższych tabelach zaprezentowano wartości przyjęte dla Zestawu I.

Tabela 3. Dane zdolności pracowników

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Lp.** | **U1** | **U2** | **U3** | **U4** | **teamwork** |
| **1.** | 9 | 8 | 7 | 10 | 84 |
| **2.** | 10 | 10 | 10 | 7 | 90 |
| **3.** | 9 | 8 | 10 | 10 | 70 |
| **4.** | 9 | 6 | 7 | 10 | 85 |
| **5.** | 10 | 7 | 10 | 6 | 92 |
| **6.** | 9 | 8 | 7 | 10 | 87 |
| **7.** | 9 | 8 | 6 | 10 | 87 |
| **8.** | 8 | 10 | 4 | 10 | 98 |
| **9.** | 11 | 11 | 9 | 9 | 81 |
| **10.** | 10 | 8 | 11 | 6 | 80 |

Objaśnienia:

***Ux*** - umiejętność pracownika w danej dziedzinie;  
***teamwork*** - zdolność pracy grupowej wyrażona w procentach;

Tabela 4. Dane dotyczące problemów

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Lp.** | **Z1** | **Z2** | **Z3** | **Z4** | **pi** | **p\_houri** | **deadline** |
| **1.** | 1 | 2 | 3 | 4 | 1515 | 4 | 1 |
| **2.** | 4 | 5 | 6 | 7 | 1965 | 3 | 2 |
| **3.** | 9 | 10 | 11 | 12 | 1487 | 4 | 3 |
| **4.** | 2 | 5 | 8 | 2 | 1200 | 8 | 4 |
| **5.** | 4 | 7 | 8 | 9 | 1984 | 6 | 5 |
| **6.** | 2 | 6 | 2 | 8 | 2582 | 3 | 6 |
| **7.** | 1 | 2 | 3 | 4 | 1515 | 2 | 7 |
| **8.** | 5 | 6 | 7 | 8 | 1965 | 5 | 8 |
| **9.** | 9 | 10 | 11 | 12 | 1487 | 55 | 9 |
| **10.** | 2 | 5 | 8 | 2 | 1200 | 8 | 10 |
| **11.** | 4 | 7 | 8 | 9 | 1984 | 6 | 11 |
| **12.** | 2 | 6 | 2 | 8 | 2582 | 1 | 12 |
| **13.** | 9 | 10 | 8 | 12 | 1515 | 2 | 7 |
| **14.** | 4 | 5 | 8 | 2 | 1200 | 8 | 10 |
| **15.** | 4 | 6 | 8 | 8 | 2582 | 3 | 6 |

Objaśnienia:

***Zx*** zapotrzebownie w danej dziedzinie

***pi*** jednorazowa kara za niedotrzymanie terminu;

***p\_houri­***kara przyznawana za każdą jednostkę czasu powyżej przyjętego limitu;

***deadline*** oczekiwany czas zakończenia zadania;

### 3.2.2. Test czasu

W poniższym teście sprawdzono zależność wartości funkcji celu od czasu działania algorytmu. Wszystkie przebiegi algorytmu zostały wykonane dla włączonych wszystkich opcji sąsiedztwa. Wyniki przedstawione są w poniższej tabeli.

Tabela 5. Zależność czasu działania programu

|  |  |  |  |
| --- | --- | --- | --- |
| **Czas [s]** | **Początkowa wartość funkcji celu** | **Końcowa wartość funkcji celu** | **Liczba iteracji** |
| 1 | 4976,48 | 3168,43 | 422 |
| 5 | 4976,48 | 2762,47 | 2877 |
| 10 | 4976,48 | 1515,2 | 6721 |
| 20 | 4976,48 | 1515,2 | 13145 |
| 100 | 4976,48 | 1515,2 | 66495 |

### 3.2.3. Test kilku przebiegów

Kolejny test miał na celu sprawdzenie wpływu czynników losowych na przebieg algorytmu. Wykonano 10 prób, na podstawie których zanotowano wartość funkcji celu. Test został przeprowadzony dla pełnej konfiguracji sąsiedztwa oraz dla czasu 20s działania algorytmu.

Rys. 2. Wartość funkcji celu w kolejnych przebiegach

Wartość średnia: **1518,88**Odchylenie standardowe: **4,60**

### 3.2.4. Test sąsiedztwa

Pierwszym rodzajem przeprowadzonego testu, było sprawdzenie wpływu sąsiedztwa na wartość funkcji celu. Wszystkie przebiegi zostały wykonane z czasem działania algorytmu równym 60 s. Wybierając, które z sąsiedztw mają być włączone, a które wyłączone wyliczono wartość funkcji celu oraz zwizualizowano rozwiązanie.

Rys. 3. Wartość funkcji celu dla określonej konfiguracji sąsiedztwa

Można zauważyć, że dla trzech konfiguracji sąsiedztwa wartość funkcji celu zbiega się do jednej wartości, natomiast trzy pozostałe konfiguracje nie pozwalają na osiągnięcie funkcji celu na poziomie 1500 w czasie 60s.

Na poniższych wykresach pokazano wizualizację rozwiązania dla wymienionych powyżej konfiguracji sąsiedztwa.

D:\Studia\taboo search\testy\d1_progi_b\untitled.tifD:\Studia\taboo search\testy\d1_progi_a\untitled.tifRys. 4. ADD + REMOVE

Rys. 5. REMOVE + ADD\_AND\_REMOVE

D:\Studia\taboo search\testy\d1_progi_e\untitled.tifD:\Studia\taboo search\testy\d1_progi_d\untitled.tifRys. 6. ADD\_AND\_REMOVE + SWAP + TAKE

Rys. 7. SWAP + TAKE + TO\_LAZY

D:\Studia\taboo search\testy\d1_progi_wszystkie\untitled.tifD:\Studia\taboo search\testy\d1_progi_f\untitled.tifRys. 8. ADD\_ AND\_REMOVE + SWAP + TAKE + TO\_LAZY

Rys. 9. ALL THRESHOLDS

### 3.2.5. Test rozmiaru sąsiedztwa

W tym teście sprawdzono wpływ długości listy tabu oraz rozmiaru sąsiedztwa na wartość funkcji celu. Test przeprowadzono dla czasu 60s oraz pełnej konfiguracji sąsiedztwa, aby wyeliminować wpływ innych czynników na przebieg algorytmu.

Tabela 6. Wpływ rozmiaru sąsiedztwa na funkcję celu

|  |  |  |  |
| --- | --- | --- | --- |
| **Lp.** | **Rozmiar listy tabu** | **Rozmiar sąsiedztwa** | **Wartość funkcji celu** |
| **1.** | 5 | 5 | 3024,09 |
| **2.** | 5 | 10 | 1967,42 |
| **3.** | 5 | 15 | 1522,4 |
| **4.** | 5 | 20 | 1521,33 |
| **5.** | 5 | 30 | 1515,2 |
| **6.** | 5 | 40 | 1515,24 |

Dane zawarte w powyższej tabeli zostały pokazane na Rys. 10. W legendzie zostały opisane jedynie wartości rozmiaru sąsiedztwa, gdyż rozmiar listy tabu pozostał w każdym przypadku taki sam, równy 5.

Rys. 10. Zależność funkcji celu od zmian rozmiaru sąsiedztwa

### 3.2.6. Test rozmiaru listy tabu

Poniższy test miał na celu sprawdzenie wpływu rozmiaru listy tabu na wartość funkcji celu. Przeprowadzono go przy pełnej konfiguracji sąsiedztwa, oraz dla czasu 120s, aby zniwelować wpływ innych czynników na przebieg algorytmu.

Tabela 7. Wpływ rozmiaru listy tabu na funkcję celu

|  |  |  |  |
| --- | --- | --- | --- |
| **Lp.** | **Rozmiar listy tabu** | **Rozmiar sąsiedztwa** | **Wartość funkcji celu** |
| **1.** | 5 | 5 | 3024,09 |
| **2.** | 15 | 5 | 3029,76 |
| **3.** | 30 | 5 | 3017,5 |
| **4.** | 45 | 5 | 1515,69 |
| **5.** | 60 | 5 | 2746,29 |
| **6.** | 75 | 5 | 3027,14 |
| **7.** | 90 | 5 | 3494,81 |

Dane zawarte w powyższej tabeli zostały pokazane na Rys. 11. W legendzie zostały opisane jedynie wartości rozmiaru listy tabu, gdyż rozmiar sąsiedztwa pozostał w każdym przypadku taki sam, równy 5.

Rys. 11. Zależność funkcji celu od zmian rozmiaru listy tabu

## 3.3. Zestaw II

### 3.3.1. Struktura danych

Dane zostały stworzone tak jak w punkcie 3.2.1. ale z powodu ich rozmiaru nie są umieszczone w dokumentacji, gdyż stała by się ona wtedy nieczytelna.

### 3.3.2. Test czasu

Poniższy test przeprowadzono przy pełnej konfiguracji sąsiedztwa. Zmiany wartości funkcji celu w zależności od czasu działania programu pokazane są w Tabeli 8.

Tabela 8. Zależność czasu działania programu

|  |  |  |  |
| --- | --- | --- | --- |
| **Czas [s]** | **Początkowa wartość funkcji celu** | **Końcowa wartość funkcji celu** | **Liczba iteracji** |
| 10 | 18011,9 | 14038,9 | 33 |
| 50 | 18011,9 | 13557,9 | 155 |
| 100 | 18011,9 | 11009,6 | 320 |
| 300 | 18011,9 | 9261,71 | 960 |
| 900 | 18011,9 | 9156,38 | 2873 |

### 3.3.3. Test kilku przebiegów

Podobnie jak dla Zestawu danych I, przeprowadzono test wielu przebiegów. Wykonano 10 przebiegów dla czasu wywołania programu 100s. Testowanie przy dłuższych czasach jest zbyt uciążliwe dlatego zdecydowano się na powyższy czas.

Rys. 12. Zmiany funkcji celu przy kolejnych wywołaniach programu

Wartość średnia: **11194,64**

Odchylenie standardowe: **894,08**

### 3.3.4. Test rozmiaru sąsiedztwa

W tym teście sprawdzono zachowanie wartości funkcji celu w zależności od wielkości sąsiedztwa. Na poniższym wykresie pokazano przebiegi, dla rozmiaru sąsiedztwa pokazanego na legendzie. Rozmiar listy tabu nie zmieniał się i wynosił 10.

Rys. 13. Zależność funkcji celu od rozmiaru sąsiedztwa

### 3.3.5. Test rozmiaru listy tabu

Analogicznie do powyższego testu sprawdzono wpływ wielkości listy zabronień na wartość funkcji celu. Na poniższym wykresie zostały pokazane przebiegi, dla rozmiaru listy tabu pokazanego na legendzie, przy stałym rozmiarze sąsiedztwa wynoszącym 10.

Rys. 14. Zależność funkcji celu od rozmiaru listy tabu

# 4. Wnioski

Pomimo dużej losowości wprowadzonej w zaimplementowany algorytm można zauważyć pewne prawidłowości. Są one widoczne lepiej dla Zestawu I, ponieważ dla niego obliczenia były prostsze i mniej czasochłonne. Wraz ze zwiększaniem czasu wywołania programu mogliśmy zaobserwować, że algorytm zbiegał się do jednej wartości. Dla dużego zestawu danych też można zauważyć taki efekt, prawdopodobnie byłby on lepiej widoczny przy dużo dłuższym czasie wywołania programu.

Kolejny z testów, który został przeprowadzony to sprawdzenie zmian wartości funkcji celu dla kolejnych wywołań programu z identycznymi parametrami. W przypadku Zestawu I można zaobserwować niewielkie odchylenie standardowe, co pozwala wnioskować o dobrym działaniu algorytmu. Natomiast dla Zestawu II widać większy rozrzut pomiędzy kolejnymi wywołaniami, lecz przy tak dużym rozmiarze problemu jest to spowodowane wieloma ścieżkami, które algorytm wybiera.

Test rozmiaru sąsiedztwa okazał się bardzo istotnym. Zwiększanie rozmiaru sąsiedztwa bardzo wpływa na zmniejszenie wartości funkcji celu. W teście dla mniejszego zestawu danych można zauważyć, że dla rozmiaru większego od 20 funkcja celu zbiega się do jednej wartości. Podczas testów większego zestawu danych ta prawidłowość potwierdziła się i zwiększanie rozmiaru sąsiedztwa powodowało poprawienie rozwiązania.

Jak można było się spodziewać test rozmiaru listy tabu pokazał, że zwiększanie wielkości do pewnego stopnia poprawia rozwiązanie, natomiast po przekroczeniu tej wartości rozwiązanie zaczyna się pogarszać. W związku z tym, z punktu widzenia funkcji celu, jest wypośrodkować rozmiar listy zabronień i zwiększać rozmiar sąsiedztwa.

Test sąsiedztwa został przeprowadzony jedynie dla problemu o mniejszym rozmiarze, gdyż wizualizacja Zestawu II jest bardzo ciężka i nieczytelna. Można zauważyć, że w zależności od konfiguracji sąsiedztwa zmienia się wartość funkcji celu, a każde z rozwiązań jest zupełnie inne.