# Metody optymalizacji - Lista 2

Bartosz Rzepkowski

14 maja 2017

## 1 Zadanie 1

### 1.1 Opis rozwiązania

Do funkcji rozwiązującej zadanie przekazywane są 2 argumenty:

- *order* - będąca tablicą tablic, która przechowuje informację o tym, ile desek o jakiej szerokości należy wyprodukować,
- *maxWidth* - standardowa szerokość deski, którą należy rozciąć na węższe deski.

Przykład parametrów:

- *order* = [[7, 110], [5, 120], [3, 80]] - należy wyprodukować 110 desek o szerokości 7 cali, 120 desek o szerokości 5 cali oraz 80 desek o szerokości 3 cali,
- $maxWidth = 22$ - standardowa deska ma szerokość 22 cali.

W celu rozwiązania zadania stworzono funkcję rekurencyjną, która generuje wszystkie możliwe podziały deski o standardowej długości na węższe deski. Przykładowo, dla powyższych parametrów (22 cale oraz zamówienie [[7, 110], [5, 120], [3, 80]]), zwraca zbiór (nazywany *allSubSets*) postaci:

$$[[3,3,3,3,5,5],[3,3,3,3,3,3],[5,5,5,7],[3],[3,3,3,3,3,7],[3,5,7,7],[7,7,7],...$$

W dalszej części sprawozdania długość zbioru *allSubSets* będzie oznaczana przez $n$.

Następnie dla każdej kombinacji ze zbioru *allSubSets* obliczane są odpowiadające im pozostałości, jakie otrzymamy po rozcięciu standardowej deski na mniejsze (pochodzące z tej kombinacji). Przykładowo dla $[5,5,5,7]$ nie otrzymamy żadnych pozostałości, a dla $[7,7,7]$ otrzymamy 1 cal odpadu. Dane te są przechowywane w tablicy *leftovers* długości $n$.

Podsumowując, funkcja rekurencyjna zwraca zbiór *allSubSets* długości $n$, którego każdemu elementowi (kombinacji węższych desek) przypisany jest element w tablicy *leftovers* z otrzymanymi odpadami.

### 1.1.1 Opis modelu

W modelu jako **zmienna** wykorzystywana jest tablica *boards* długości $n$.
**Funkcja celu** minimalizuje ilość otrzymywanych odpadów i ma postać:

$$\sum_{i=1}^{n} boards[i] * leftovers[i] \rightarrow min$$

Dla każdej pary z tablicy *order* nałożono również **ograniczenie**, narzucające wyprodukowanie odpowiedniej liczby desek. Zapisane w pseudokodzie ma postać:

```
for i in 1:length(order) do
    sum = 0
    for j in 1:n do
        for k in 1:length(allSubSets[j]) do
            if allSubSets[j][k] == order[i][1] then
                sum += boards[j]
            else
        end
    end
    sum ⩾ order[i][2]end
```

Powyższy kod można rozumieć następująco:

- dla każdego zamówienia (*[szerokość, ilość]*) zainicjuj zmienną *sum = 0* i wykonaj następujące czynności:
  - przejdź po wszystkich możliwych kombinacjach podziału standardowej deski,
  - przejdź po wszystkich elementach kombinacji,
  - jeśli element kombinacji jest równy rozpatrywanej *szerokości*, dodaj do *sum* wartość *boards*
- *sum* musi być większe lub równe rozpatrywanej *ilości*.

## 1.2 Wyniki

Dla przedstawionych danych otrzymano następujące wyniki:
[5,5,5,7] → 27.727272727272727273
[3,3,3,3,3,7] → 8.636363636363637
[3,5,7,7] → 36.81818181818181
Po zaokrągleniu wyników w górę otrzymujemy:
[5,5,5,7] → 28
[3,3,3,3,3,7] → 9
[3,5,7,7] → 37

# 2 Zadanie 2

## 2.1 Opis modelu (na podstawie programu dr hab. Pawła Zielińskiego)

Do programu przekazywane są następujące parametry:

- $p$ - wektor z informacją, ile czasu zajmuje wykonanie każdego zadania,
- $r$ - wektor dostępności zadania,
- $w$ - wektor wag zadań.

Wprowadźmy również następujące zmienne:

- $n$ - długość wektora $p$,
- $T$ - długość horyzontu czasowego $= max(r) + \sum(p) + 1$
- *Tasks*$= 1 : n$ - wszystkie kolejne zadania,
- *Horizon*$= 1 : T$ - wszystkie sloty w horyzoncie czasowym.

### 2.1.1 Zmienne

W modelu zastosowana będzie tablica $x$ o rozmiarach *Tasks x Horizon*, której komórki będą przyjmowały wartości binarne.

### 2.1.2 Funkcja celu

Model ma za zadanie minimalizować następującą funkcję celu:

$$\sum_{i=1}^{n} w[i] * c[i],$$

gdzie $c$ ma postać $(t-1)+p[j]$ (moment rozpoczęcia zadania + czas jego trwania), zatem funkcja celu ostatecznie ma następującą postać:

$$\sum_{i=1}^{n} \sum_{t=1}^{T} w[i] * ((t-1) + p[i]) * x[i,t] \to min.$$

### 2.1.3 Ograniczenia

Aby każde zadanie rozpoczęło się dokładnie jeden raz nałożono ograniczenie:

**for** $i$ *in Tasks* **do**
$\quad \sum_{t=1}^{T-p[i]+1} x[i,t] == 1$
**end**

Aby zadania nie wykonywały się wcześniej, niż przypisany im moment gotowości (element tablicy $r$) nałożono ograniczenie:

**for** $i$ *in Tasks* **do**

$\quad \sum_{t=1}^{T-p[i]+1}(t-1) * x[i,t] \geqslant$ r[i]

**end**

Ostatecznie, aby zadania nie nakładały się na siebie nałożono ostatnie ograniczenie:

**for** $t$ *in Horizon* **do**

$\quad \sum_{i=1}^{n} \sum_{s=max(1,t-p[i]+1)}^{t} x[i,s] \leqslant 1$

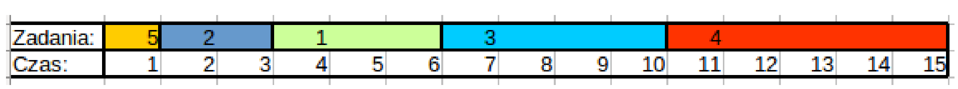
**end**

## 2.2 Wyniki

Dla przykładowych wektorów:

- p = [3, 2, 4, 5, 1],
- r = [2, 1, 3, 1, 0],
- w = [1, 1, 1, 1, 1]

otrzymano następujące wyniki:
[1, 4] = 1.0
[2, 2] = 1.0
[3, 7] = 1.0
[4,11] = 1.0
[5, 1] = 1.0
Na diagramie Gantt'a przedstawiają się one następująco:




# 3 Zadanie 3

## 3.1 Opis modelu

Do programu przekazywane są następujące parametry:

- $p$ - wektor z informacją, ile czasu zajmuje wykonanie każdego zadania,
- $M$ - ilość maszyn, na których możemy wykonywać zadania,
- *graph* - graf reprezentujący relacje poprzedzania.

Model w tym zadaniu ma bardzo podobną postać do modelu z zadania poprzedniego. Aby je pokazać, musimy ponownie zdefiniować pewne zmienne oraz dodać nowe (zmienne, które pojawiały się w poprzednim zadaniu, a nie zostały podane w tym pozostały bez zmian):

- $T = \sum(p) + 1$
- *Machines = 1:m*

### 3.1.1 Zmienne

Do tablicy $x$ z poprzedniego zadania dodajemy trzeci wymiar, stąd ma ona rozmiar *Tasks x Horizon x Machines*.

## 3.2 Funkcja celu

W omawianym zadaniu nie mamy podanego wektora wag dla zadań, dlatego nie będziemy go uwzględniać w funkcji celu. Ponadto, musimy także iterować po wszystkich maszynach, stąd funkcja celu ostatecznie przyjmuje postać:

$$\sum_{i=1}^{n}\sum_{t=1}^{T}\sum_{m=1}^{M}((t-1)+p[i]) * x[i,t,m] \to min.$$

## 3.3 Ograniczenia

Aby zadanie rozpoczęło się tylko raz musimy dodać ograniczenie:

**for** *i in Tasks* **do**
  $\sum_{t=1}^{T-p[i]+1}\sum_{m=1}^{M} x[i,t,m] == 1$
**end**

Aby zadania nie nakładały się na siebie nałożono ostatnie ograniczenie:

**for** *t in Horizon* **do**
  **for** *m in Machines* **do**
    $\sum_{i=1}^{n}\sum_{s=max(1,t-p[i]+1)}^{t} x[i,s,m] \leqslant 1$
  **end**
**end**

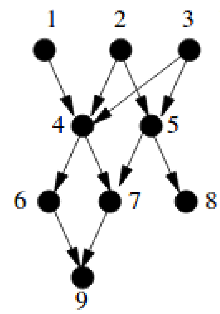
Ostatnie ograniczenie musi gwarantować, aby zadania wykonywały się zgodnie z zadanym grafem. Oznaczmy przez *neighbors(i)* wszystkich sąsiadów rozpatrywanego wierzchołka (tzn. wierzchołki, **do** których wychodzi krawędź z rozpatrywanego wierzchołka). Zatem ostatnie ograniczenie ma postać:

**for** *i in Tasks* **do**
  **for** *neighbor in neighbors(i)* **do**
    $\sum_{t=1}^{T-p[neighbor]+1}\sum_{m=1}^{M} x[neighbor,t,m] * (t-1) \geqslant$
     $\sum_{t=1}^{T-p[i]+1}\sum_{m=1}^{M}(x[i,t,m] * (t-1)) + p[i]$
  **end**
**end**

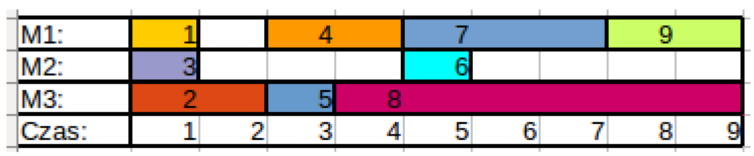
Zapewnia ono, że dla każdego wierzchołka wszyscy jego sąsiedzi żozpoczną działanie”dopiero po tym, jak on skończy (po upływie *p[i]* czasu od momentu rozpoczęcia jego pracy).

### 3.4 Wyniki

Dla wektora $p$ = *[1, 2, 1, 2, 1, 1, 3, 6, 2]* oraz grafu




otrzymano następujący rozkład:




# 4 Zadanie 4

Do funkcji rozwiązującej to zadanie są przekazywane takie same argumenty ja kdo poprzedniego, a ponadto następujące dwa:

- $r$ - wektor z wymaganymi zasobami dla poszczególnych zadań,
- $N$ - ograniczenie na wykorzystywany zasób.

### 4.1 Opis modelu

Model działa bardzo podobnie do modelu z poprzedniego zadania. Za ilość maszyn przyjęto w nim $n$. Przyjęto to założenie, ponieważ jeśli pozwoli na to limit zasobów (będzie dostatecznie duży), a graf relacji między zadaniami nie będzie zawierał żadnych krawędzi (wszystkie zadania będą mogły rozpocząć się w chwili "0"), wszystkie zadania będą mogły wykonywać się jednocześnie.

### 4.2 Funkcja celu

Funkcja celu ma taką samą postać jak w poprzednim zadaniu:

$$\sum_{i=1}^{n}\sum_{t=1}^{T}\sum_{m=1}^{M}((t-1)+p[i]) * x[i,t,m] \rightarrow min.$$

## 4.3 Ograniczenia

W modelu wykorzystano takie same ograniczenia jak w poprzednim. Ponadto dodano następujące ograniczenie na wykorzystanie zasobów:

**for** *t in Horizon* **do**

$$\sum_{i=1}^{n}\sum_{m=1}^{M}\sum_{s=max(1,t-p[i]+1)}^{t} x[i,s,m]*r[i] \leqslant N$$

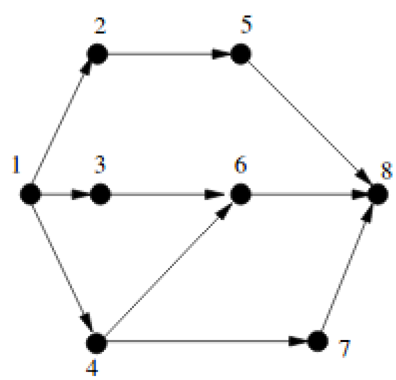
**end**

## 4.4 Wyniki

Dla następujących danych:

- p = [5, 4, 5, 4, 3, 5, 1, 6]
- r = [9, 17, 11, 4, 13, 7, 7, 17]
- N = 30

i grafu:




otrzymano następujący grafik: