Metody Optymalizacji w Zarządzaniu

# Spis treści



# 1 Przydział miejsc w parlamencie

## 1.1 Tu kilka stron o początkowych tematach

## 1.2 Tijdeman

### 1.2.1 Sformułowanie problemu

**k** oddziałów towarzystwa

$\lambda$**i** znormalizowana liczba członków w oddziale i, i=1,..,k

$$\sum_{i}^{K} \lambda i = 1 \tag{1}$$

$\omega$**i** numer oddziału, którego pochodził przewodniczący w okresie j; j=1,..;

**A**$\omega$**(i,n)** liczba przewodniczących z oddziału i w okresie [1,n];

zminimalizować D($\omega$)=

### 1.2.2 Rozwiązanie

$$\min_{i,n} |\lambda in - A\omega(i,n)| \leqslant 1 - \frac{1}{2K-2} \tag{2}$$

### 1.2.3 Wyznaczanie oddziałów spełniających nierówność

$$\sigma i = \lambda in - A\omega(i,n-1) \geqslant \frac{1}{2K-2} \tag{3}$$

$$\frac{1 - \frac{1}{2K-2} - \sigma i}{\lambda i} \rightarrow min \tag{4}$$

## 1.3 PRV

## 1.4 Problem Liu-Laylanda

- jeden proces,
- n cyklicznych, niezależnych, podzielnych zadań o czasie wykonania Ci i okresie Ti,
- zadanie musi być wykonane w okresie Ti

### 1.4.1 Algorytm statyczny

- posortuj zadania rosnąco według okresów Ti
- zadanie o krótszym okresie ma większy priorytet
- wykonuj zadanie tak długo jak nie pojawi się zadanie o wyższym priorytecie

### 1.4.2 Algorytm dynamiczny

- wyznacz dla każdego zadania linię krytyczną
- zadanie posiadające najbliższą linię krytyczną ma największy priorytet
- współczynnik wykorzystania pracy procesora

$$U = \sum_{i=1}^{n} \frac{Ci}{Ti} \tag{5}$$

- dla algorytmu ze stałym priorytetem

$$U = m(\sqrt[m]{2} - 1) \tag{6}$$

## 1.5 MAD - Mean Absolute Deviation

### 1.5.1 Sformułowanie problemu

- szeregowanie zadań na jednej maszynie
- n niezależnych i niepodzielnych zadań
- jeden, wspólny, żądany termin zakończenia dla wszystkich zadań di=d dla wszystkich i
- ti czas wykonania zadania i
- fi(ei)=ei; gi(ti)=ti
- wyprowadzenie funkcji celu

$$\sum_{i=1}^{n} |Ci - d| \tag{7}$$

### 1.5.2 Własności problemu

- najdłuższe zadanie uszeregowane jako pierwsze
- no idle time - brak przerwy mieędzy zadaniami
- V-shape uszeregowane zadania przed terminem zakończenia są uporządkowane według nierosnących czasów wykonania zadania, natomiast zadania które są uszeregowane po terminie zakończenia są uporządkowane według niemalejących czasów wykonania
- jedno zadanie kończy się dokładnie w żadanym terminie zakończenia $\left\lceil \frac{n}{2} \right\rceil$ zadań kończy się przed lub w żądanym terminie zakończenia
- problem restryktywny może przecinać zadanie

Algorym Kaneta

1. E=$\phi$, T=$\phi$
2. WHILE J$\neq \phi$ DO

   BEGIN Remove a job k from J such that pi = maxpi Insert job k into the last position in E. If J!=0 DO BEGIN remove a job k from J such taht pi=max(pi) insert a job k into the first position in T END END S=(E,T) (konkatenacja)

## 1.6 WSAD weighted sum of absolute deviations

### 1.6.1 Sformułowanie problemu

- n niezależnych, niepodzielnych zadań
- pi czas wykonania zadania i, i=1,..,n
- d wspólny, żądany termin zakończenia
- $\alpha$ jednostkowy koszt wykonania zadania przed terminem
- $\beta$ jednostkowy koszt wykonania zadania po terminie
- cel: zminimalizować

$$\sum_{i=1}^{n}(\alpha ei + \beta ti) \tag{8}$$

### 1.6.2 Własności

- no idle time
- V-shape
- dla problemu unrestricted zadanie k, gdzie $\left\lceil k^* = \frac{\beta n}{\alpha+\beta} \right\rceil$ kończy się w chwili d

## 1.7 WSAD unrestriced

Algorytm unrestricted (Bagchi, Sullivan, Chang, 1987)

- Zadania uporządkowane są według LPT (od najdłuższego)
- —E— liczba zadań w zbiorze E (przed terminem)
- —T— liczba zadań w zbiorze T (po terminie)

  Krok 1. Zbiór E, T=0 Krok 2 For k=1 TO n DO BEGIN usuń zadanie k ze zbioru J IF $(\alpha * |E|) < (\beta * (|T| + 1))$ dodaj zadanie jako ostatnie do zbioru E ELSE dodaj na początek zbioru T END S=(E+T)

## 1.8 WSAD restriced

## 1.9 TWET - total weighted...

### 1.9.1 Sformułowanie problemu

- n niezależnych, niepodzielnych zadań
- pi czas wykonania zadania i
- $\alpha i$ jednostkowy koszt wykonania zadania i przed terminem
- $\beta i$ jednostkowy koszt wykonania zadania i po terminie
- d wspólny żądany termin zakończenia

### 1.9.2 Funkcja celu

zminimalizować $\sum_{i=1}^{n} (\alpha iei + \beta iti)$

### 1.9.3 Własności

- no idle time
- non-restriced - jedno zadanie kończy się w due-date
- NP-trudny nawet dla przypadku nierestryktywnego
- $E \in \{Ci < d\}, T \in \{Ci \geqslant d\}$, zadania w zbiorze T są uszeregowane według niemalającej wartości $\frac{pi}{\beta i}$, a zadania w zbiorze E są uszeregowane według nierosnącej wartości $\frac{pi}{\alpha i}$
- uszeregowanie optymalne

$$\sum_{i \in T} \beta i \leqslant \sum_{j \in Ei} \alpha j \tag{9}$$

- jeżeli znamy optymalny podział zadań na zbiory E i T, to znalezienie uszeregowania optymalnego jest łatwe
- istnieje algorytm pseudo-wielomianowy dla szczególnego przypadku, gdy dla każdego i i j

$$(\frac{pi}{\alpha i} < \frac{pj}{\alpha j}) \Rightarrow (\frac{pi}{\beta i} < \frac{pj}{\beta j}) \tag{10}$$

### 1.9.4 Algorytm pełnego przeglądu

1. Porządkowanie zadań według nierosnących wartości $\frac{pi}{\alpha i}$
2. Wszystkie zadania umieszczamy w zbiorze E
3. Na każdym poziomie przenosimy jedno zadanie ze zbioru E na początek zbioru T
4. Jeżeli naruszone są warunki (I) lub (II) to kończymy gałąź. Jeżeli uszeregowanie jest dopuszczalne, to obliczamy jego koszt.
5. Rozwijamy drzewo do sprawdzenia wszystkich dopuszczalnych uszeregowań

## 1.10 CTV - Completion Time Variance

### 1.10.1 Sformułowanie problemu

- n niezależnych, niepodzielnych zadań
- d wspólny, żądany termin zakończenia
- pi czas wykonania zadania i=1,..,n
- $\alpha i = \beta i = 1$ i=1,..,n