Szeregowanie zadań

Wykład nr 4

dr Hanna Furmańczyk

Minimalizacja łącznego czasu zakończenia zadania $\sum C_j$. Zadania niezależne

- Obserwacja: krótkie zadania umieszczamy na początku - reguła SPT (ang. *Shortest Processing Time*)

Minimalizacja łącznego czasu zakończenia zadania $\sum C_j$. Zadania niezależne

- Obserwacja: krótkie zadania umieszczamy na początku - reguła SPT (ang. *Shortest Processing Time*)
- trzeba jeszcze znaleźć optymalne przypisanie zadań do procesorów

Alg. optymalny - $P||\sum C_j$, $P|pmtn|\sum C_j$ $(O(n \log n))$

1. Przyjmij, że liczba zadań dzieli się przez m (ew. wprowadź zadania puste).

Alg. optymalny - $P||\sum C_j$, $P|pmtn|\sum C_j$ $(O(n \log n))$

1. Przyjmij, że liczba zadań dzieli się przez m (ew. wprowadź zadania puste).
2. Uporządkuj je wg SPT.

Alg. optymalny - $P||\sum C_j$, $P|pmtn|\sum C_j$ $(O(n \log n))$

1. Przyjmij, że liczba zadań dzieli się przez m (ew. wprowadź zadania puste).
2. Uporządkuj je wg SPT.
3. Przypisuj kolejne m-ki zadań do maszyn (dowolnie).

Alg. optymalny - $P||\sum C_j$, $P|pmtn|\sum C_j$ $(O(n \log n))$

1. Przyjmij, że liczba zadań dzieli się przez m (ew. wprowadź zadania puste).
2. Uporządkuj je wg SPT.
3. Przypisuj kolejne m-ki zadań do maszyn (dowolnie).

Przykład: $m = 2$, $n = 5$, $\mathbf{p} = (2, 5, 3, 1, 3)$

Alg. optymalny - $P||\sum C_j$, $P|pmtn|\sum C_j$ $(O(n \log n))$

1. Przyjmij, że liczba zadań dzieli się przez m (ew. wprowadź zadania puste).
2. Uporządkuj je wg SPT.
3. Przypisuj kolejne m-ki zadań do maszyn (dowolnie).

Przykład: $m = 2$, $n = 5$, $\mathbf{p} = (2, 5, 3, 1, 3)$

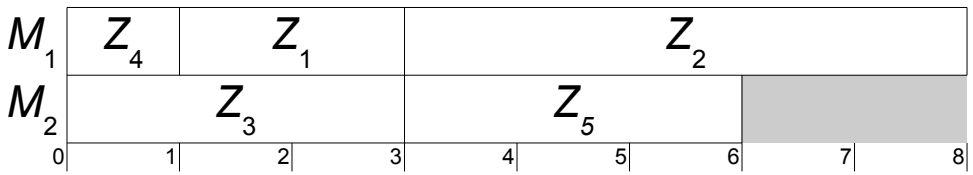
$SPT:$	Z_0	Z_4	Z_1	Z_3	Z_5	Z_2
p_i	0	1	2	3	3	5

Alg. optymalny - $P||\sum C_j$, $P|pmtn|\sum C_j$ $(O(n \log n))$

1. Przyjmij, że liczba zadań dzieli się przez m (ew. wprowadź zadania puste).
2. Uporządkuj je wg SPT.
3. Przypisuj kolejne m-ki zadań do maszyn (dowolnie).

Przykład: $m = 2$, $n = 5$, $\mathbf{p} = (2, 5, 3, 1, 3)$

SPT :	Z_0	Z_4	Z_1	Z_3	Z_5	Z_2
p_i	0	1	2	3	3	5

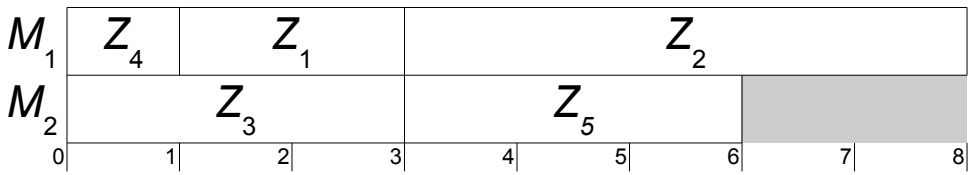


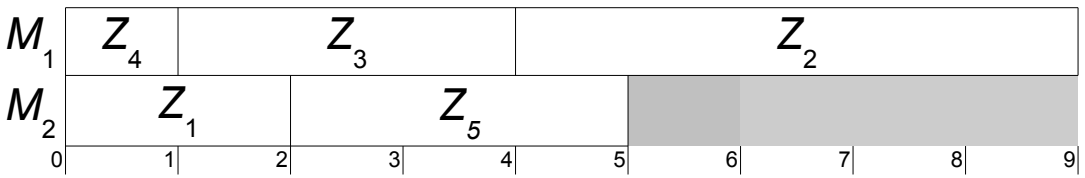
Alg. optymalny - $P||\sum C_j$, $P|pmtn|\sum C_j$ $(O(n \log n))$

1. Przyjmij, że liczba zadań dzieli się przez m (ew. wprowadź zadania puste).
2. Uporządkuj je wg SPT.
3. Przypisuj kolejne m-ki zadań do maszyn (dowolnie).

Przykład: $m = 2$, $n = 5$, $\mathbf{p} = (2, 5, 3, 1, 3)$

SPT :	Z_0	Z_4	Z_1	Z_3	Z_5	Z_2
p_i	0	1	2	3	3	5



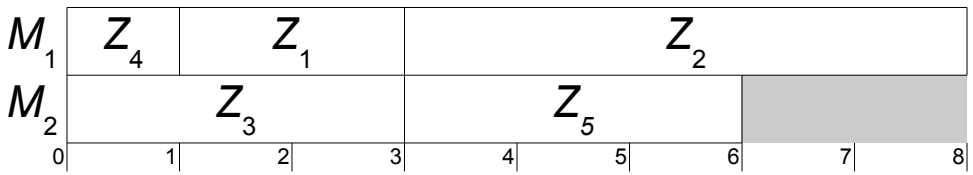


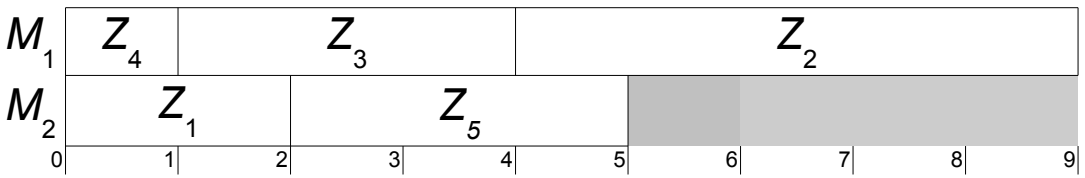
Alg. optymalny - $P||\sum C_j$, $P|pmtn|\sum C_j$ $(O(n \log n))$

1. Przyjmij, że liczba zadań dzieli się przez m (ew. wprowadź zadania puste).
2. Uporządkuj je wg SPT.
3. Przypisuj kolejne m-ki zadań do maszyn (dowolnie).

Przykład: $m = 2$, $n = 5$, $\mathbf{p} = (2, 5, 3, 1, 3)$

SPT :	Z_0	Z_4	Z_1	Z_3	Z_5	Z_2
p_i	0	1	2	3	3	5





$$\sum C_j = 21$$

Minimalizacja łącznego ważonego czasu zakończenia zadania $\sum w_j C_j$. Zadania niezależne, niepodzielne

Problem $P2||\sum w_j C_j$ ($P2|pmtn|\sum w_j C_j$) jest NP-trudny.

Minimalizacja łącznego ważonego czasu zakończenia zadania $\sum w_j C_j$. Zadania niezależne, niepodzielne

Problem $P2||\sum w_j C_j$ ($P2|pmtn|\sum w_j C_j$) jest NP-trudny.

$1||\sum w_j C_j$ - alg. optymalny $O(n \log n)$

Reguła Smitha - uogólnienie SPT:

- ustaw zadania w kolejności niemalejących p_j/w_j

Procesory identyczne, zadania zależne

- przypadki NP-trudne: $1|prec|\sum C_i$,

Procesory identyczne, zadania zależne

- przypadki NP-trudne: $1|prec|\sum C_i$, $P2|prec, p_j = 1|\sum C_j$,

Procesory identyczne, zadania zależne

- przypadki NP-trudne: $1|prec|\sum C_i$, $P2|prec, p_j = 1|\sum C_j$, $P2|chains, pmtn|\sum C_j$

Procesory identyczne, zadania zależne

- przypadki NP-trudne: $1|prec|\sum C_i$, $P2|prec, p_j = 1|\sum C_j$, $P2|chains, pmtn|\sum C_j$
- przypadki wielomianowe: $P|out - tree, p_j = 1|\sum C_j$ (adaptacja alg. Hu)

Procesory identyczne, zadania zależne

- przypadki NP-trudne: $1|prec|\sum C_i$, $P2|prec, p_j = 1|\sum C_j$, $P2|chains, pmtn|\sum C_j$
- przypadki wielomianowe: $P|out - tree, p_j = 1|\sum C_j$ (adaptacja alg. Hu)
- wersja ważona: Problem $1|prec, p_j = 1|\sum w_j C_j$ jest NP trudny

Minimalizacja maksymalnego opóźnienia - maszyny równoległe

- Aby opóźnienie $L_i = C_i - d_i$ zadania Z_i w harmonogramie było określone, zadania muszą być wyposażone w oczekiwane terminy zakończenia d_i.

Minimalizacja maksymalnego opóźnienia - maszyny równoległe

- Aby opóźnienie $L_i = C_i - d_i$ zadania Z_i w harmonogramie było określone, zadania muszą być wyposażone w oczekiwane terminy zakończenia d_i.
- Spóźnienie zadania $T_i = \max\{L_i, 0\}$ nie bierze pod uwagę wykonania się zadań przed terminem.

Minimalizacja maksymalnego opóźnienia - maszyny równoległe

- Aby opóźnienie $L_i = C_i - d_i$ zadania Z_i w harmonogramie było określone, zadania muszą być wyposażone w oczekiwane terminy zakończenia d_i.
- Spóźnienie zadania $T_i = \max\{L_i, 0\}$ nie bierze pod uwagę wykonania się zadań przed terminem.
- Wniosek: $T_{\max} = \max\{L_{\max}, 0\}$. Dlatego kryterium $T_{\max}$ nie rozważamy osobno – harmonogram $L_{\max}$-optymalny jest też $T_{\max}$-optymalny.

Minimalizacja maksymalnego opóźnienia - maszyny równoległe

- Aby opóźnienie $L_i = C_i - d_i$ zadania Z_i w harmonogramie było określone, zadania muszą być wyposażone w oczekiwane terminy zakończenia d_i.
- Spóźnienie zadania $T_i = \max\{L_i, 0\}$ nie bierze pod uwagę wykonania się zadań przed terminem.
- Wniosek: $T_{\max} = \max\{L_{\max}, 0\}$. Dlatego kryterium $T_{\max}$ nie rozważamy osobno – harmonogram $L_{\max}$-optymalny jest też $T_{\max}$-optymalny.
- kryterium $L_{\max}$ jest uogólnieniem $C_{\max}$, zagadnienia NP–trudne dla $C_{\max}$ pozostaną takie w przypadku $L_{\max}$

- mając do wykonania wiele prac z różnymi oczekiwanymi terminami zakończenia spóźnimy się „najmniej” zaczynając zawsze od „najpilniejszej” pracy,

- mając do wykonania wiele prac z różnymi oczekiwanymi terminami zakończenia spóźnimy się „najmniej" zaczynając zawsze od „najpilniejszej" pracy,
- inaczej: w różnych wariantach stosujemy regułę EDD (ang. *Earliest Due Date*) – wybieraj zadania Z_j w kolejności niemalejących oczekiwanych terminów zakończenia d_j

- mając do wykonania wiele prac z różnymi oczekiwanymi terminami zakończenia spóźnimy się „najmniej" zaczynając zawsze od „najpilniejszej" pracy,
- inaczej: w różnych wariantach stosujemy regułę EDD (ang. *Earliest Due Date*) – wybieraj zadania Z_j w kolejności niemalejących oczekiwanych terminów zakończenia d_j
- problem zadań niepodzielnych na jednej maszynie $(1||L_{\max})$ rozwiązuje właśnie szeregowanie według EDD.

Algorytm Liu $O(n^2)$ - oparty na regule EDD

1. Spośród dostępnych zadań przydziel maszynę temu, które ma najmniejszy wymagany termin zakończenia.
2. Jeśli zadanie zostało zakończone lub przybyło nowe - wróć do punktu 1.

Algorytm Liu $O(n^2)$ - oparty na regule EDD

1. Spośród dostępnych zadań przydziel maszynę temu, które ma najmniejszy wymagany termin zakończenia.
2. Jeśli zadanie zostało zakończone lub przybyło nowe - wróć do punktu 1.

Przykład - osobne slajdy

Minimalizacja $L_{\max}$ - zadania niezależne, niepodzielne

- Niektóre przypadki NP–trudne: $P2||L_{\max}$, $1|r_j|L_{\max}$
- Przypadki wielomianowe:
  - zadania jednostkowe: $P|p_j = 1, r_j|L_{\max}$, $Q|p_j = 1|L_{\max}$
  - $1||L_{\max}$ (wg EDD) - rozw. optymalne

Minimalizacja $L_{\max}$ - zadania zależne, podzielne

$1|pmtn, prec, r_j|L_{\max}$ - zmodyfikowany alg. Liu $O(n^2)$

1. Określ zmodyfikowane terminy zakończenia zadań:

$$d_j^* = \min\{d_j : \min\{d_i : Z_j \prec Z_i\}\}$$

2. Szereguj według EDD dla nowych d_j^* z wywłaszczaniem zadania, gdy pojawia się nowe, wolne, z mniejszym zmodyfikowanym terminem zakończenia
3. Powtarzaj 2 aż do uszeregowania wszystkich zadań.

Minimalizacja $L_{\max}$ - zadania zależne, podzielne

$1|pmtn, prec, r_j|L_{\max}$ - zmodyfikowany alg. Liu $O(n^2)$

1. Określ zmodyfikowane terminy zakończenia zadań:

$$d_j^* = \min\{d_j : \min\{d_i : Z_j \prec Z_i\}\}$$

2. Szereguj według EDD dla nowych d_j^* z wywłaszczaniem zadania, gdy pojawia się nowe, wolne, z mniejszym zmodyfikowanym terminem zakończenia
3. Powtarzaj 2 aż do uszeregowania wszystkich zadań.

Przykład - osobne slajdy.

Minimalizacja $L_{\max}$ - zadania zależne, niepodzielne

Trochę faktów

- Problem $P|p_j = 1, out - tree|L_{\max}$ jest NP-trudny.

Minimalizacja $L_{\max}$ - zadania zależne, niepodzielne

Trochę faktów

- Problem $P|p_j = 1, out - tree|L_{\max}$ jest NP-trudny.
- algorytm wielomianowy dla $P2|prec, p_j = 1|L_{\max}$

Minimalizacja $L_{\max}$ - zadania zależne, niepodzielne

Trochę faktów

- Problem $P|p_j = 1, out - tree|L_{\max}$ jest NP-trudny.
- algorytm wielomianowy dla $P2|prec, p_j = 1|L_{\max}$
- **algorytm Bruckera** dla $P|p_j = 1, in - tree|L_{\max}$ $O(n \log n)$

Algorytm Bruckera - $P|p_j = 1, in - tree|L_{\max}$

$next(j)$ - bezpośredni następnik zadania Z_j

1. wylicz zmodyfikowane terminy zakończenia zadań:

$$d^*_{root} = 1 - d_{root}$$

$$d^*_k = \max\{1 + d^*_{next(k)}, 1 - d_k\}$$

2. szereguj zadania dostepne podobnie jak w alg. Hu (tu: lista tworzona jest wg nierosnących wartości d^*_j)

Przykład - osobne slajdy

Szeregowanie zadań na procesorach dedykowanych - kolejne wykłady