Spis treści

1 Wprowadzenie **1**

2 Postać problemu **1**
2.1 Maszyny . . . 1
2.2 Zadania . . . 2
2.3 Parametry zadań . . . 2
2.4 Uszeregowanie . . . 2
2.4.1 Parametry uszeregowania . . . 2
2.4.2 Kryteria optymalizacji . . . 2
2.5 Notacja Trójpolowa . . . 2

3 Problemy na jednej maszynie **2**
3.1 $1||C_{max}$ . . . 3
3.2 $1|r_j|C_{max}$ . . . 3
3.3 $1||\sum C_j$ . . . 3
3.3.1 $1||\sum w_jC_j$ . . . 3
3.3.2 $1|pmtn, r_j|\sum C_j$ . . . 3

4 Problemy kolejnościowe **3**
4.1 $1|\text{prec}|\sum C_j$ . . . 3
4.2 $1|\text{prec}|f_{max}$ . . . 3
4.3 $1|\text{out-tree}|\sum w_jC_j$ . . . 3
4.4 $1|\text{in-tree}|\sum w_jC_j$ . . . 3

1 Wprowadzenie

Teoria szeregowania zadań zajmuje się problemami polegającymi na przydzieleniu pewnych zadań do dostępnych maszyn w taki sposób, aby pewne kryterium było optymalizowane. Będziemy się zajmować deterministycznymi problemami, czyli takimi, w których wszystkie dane są znane z góry.

2 Postać problemu

Standardowo, problem jest skonstruowany z następujących składowych:

- zadania $\mathcal{J} = \{J_1, \ldots, J_n\}$
- maszyny $\mathcal{P} = \{P_1, \ldots, P_m\}$
- zasoby $\mathcal{R} = \{R_1, \ldots, R_s\}$ dostępnych w $m_1, \ldots, m_s$ jednostkach

2.1 Maszyny

W problemie w zależności od wykonywanego zadania i maszyn mogą występować różne ograniczenia i różnice między maszynami. Jeśli mamy do czynienia z kilkoma maszynami równoległymi to te maszyny mogą być:

- P - identycznościowe czyli z jednakową szybkością
- Q - jednorodne czyli z różną szybkością między maszynami
- R - dowolne czyli z różniącą się szybkością między zadaniami i maszynami

Jeśli mamy do czynienia z maszynami dedykowanymi, gdzie każde zadanie składa się z operacji wykonywanych na różnych maszynach to maszyny mogą być:

- F - system przepływowy czyli każde zadanie przechodzi przez maszyny w tej samej kolejności
- O - system otwarty czyli kolejność wykonywania operacji jest dowolna
- J - system gniazdowy czyli każde zadanie ma ustaloną własną kolejność przechodzenia przez maszyny

2.2 Zadania

Zadanie J opisują następujące atrybuty:

- p_j - czas wykonania zadania J_j
- r_j - czas przygotowania zadania J_j
- d_j - pożądany czas zakończenia zadania J_j
- w_j - waga zadania J_j

2.3 Parametry zadań

Zbiór zadań $\mathcal{J}$ jako całość opisują ograniczenia kolejnościowe (acykliczny graf skierowany), oraz podzielność czyli czy zadania można przerywać i wznawiać.

2.4 Uszeregowanie

Uszeregowaniem nazywamy przypisanie każdemu zadaniu maszyny i zasobów w czasie. Koniecznym jest aby następujące warunki były spełnione:

- w każdej chwili maszyna wykonuje tylko jedno zadanie
- w każdej chwili każde zadanie jest wykonywane przez jedną maszynę
- Każde zadanie jest wykonywane w całości
- Spełnione są ograniczenia kolejnościowe
- Jeśli zadania są podzielne to są one przerywane skończoną ilość razy

2.4.1 Parametry uszeregowania

- moment rozpoczęcia S_j
- moment zakończenia C_j
- czas przepływu $F_j = C_j - S_j$
- opóźnienie $L_j = C_j - d_j$
- spóźnienie $T_j = \max(0, L_j)$
- przyspieszenie $E_j = \max(0, d_j - C_j)$

2.4.2 Kryteria optymalizacji

Typowo w szeregowaniu optymalizujemy jakąś funkcję składającą się z parametrów uszeregowania. Przykładowe funkcje to: $C_{max} = \max(C_j)$, $C_{sum} = \sum C_j$ czy $T_{sum} = \sum T_j$.

2.5 Notacja Trójpolowa

$$\alpha|\beta|\gamma$$

gdzie α określa ograniczenia maszyn, β określa ograniczenia zadań, a γ określa kryterium optymalizacji.

3 Problemy na jednej maszynie

Poniżej są opisane raczej trywialne problemy z minimalnymi utrudnieniami.

3.1 $1||C_{max}$

Jest jedna maszyna i n zadań z czasami przetwarzania p_j. Każde uszeregowanie bez przestojów jest optymalne. $C_{max} = \sum_{j=1}^{n} C_j$.

3.2 $1|r_j|C_{max}$

Jest jedna maszyna i n zadań z czasami przetwarzania p_j i czasami, od których są dostępne r_j. Tutaj możliwe, że przestoje są nieuniknione. Aby rozwiązać ten problem, sortujemy zadania po r, po czym kolejno je szeregujemy.

3.3 $1||\sum C_j$

Jest jedna maszyna i n zadań z czasami przetwarzania p_j. Chcemy minimalizować sumę ich czasów zakończeń, więc lepiej najpierw wykonać najkrótsze zadania. Nazywamy to SPT.

3.3.1 $1||\sum w_j C_j$

Jest jedna maszyna i n zadań z czasami przetwarzania p_j i wagami w_j. Chcemy minimalizować ważoną sumę ich zakończeń, co za tym idzie najwyżej ważone zadania chcemy wykonywać jako pierwsze. Aby to osiągnąć wystarczy posortować zadania po $\frac{p_j}{w_j}$. Nazywamy to WSPT i jest to ogólniejszy przypadek SPT.

3.3.2 $1|pmtn, r_j|\sum C_j$

Jest jedna maszyna i n zadań z czasami przetwarzania p_j, z czasami gotowości r_j, które **są podzielne**. Tutaj ponownie stosujemy algorytm SPT, ale w momencie, gdy skończymy zadanie, lub zadanie stanie się dostępne, zmieniamy wykonywane zadanie na to o najkrótszym czasie przetwarzania. Nazywamy to SRPT.

4 Problemy kolejnościowe

W momencie dodania ograniczeń kolejnościowych, z reguły opisanych grafem kolejności, złożoność problemu znacznie rośnie. Załóżmy, że graf jest reprezentowany przez listę sąsiedztwa.

4.1 $1|\textbf{prec}|\sum C_j$

W tym problemie mamy jedną maszynę i n zadań nieprzerywanlnych z czasami przetwarzania p_j. Próba rozwiązania tego problemu tylko przy pomocy DFS nie wystarczy. Konieczny jest algorytm Kahna, który sortuje topologicznie graf. Algorytm Kahna jest bardzo podobny do DFS, ale zamiast używania stosu, używamy kolejki priorytetowej, czyli najpierw zwiedzamy te wierzchołki o najkrótszym czasie przetwarzania.

4.2 $1|\textbf{prec}|f_{max}$

W tym problemie mamy jedną maszynę i n zadań nieprzerywalnych z czasami przetwarzania p_j. Wystarczy rozważać szeregowania bez przestojów. Problem ten rozwiązuje algorytm Lawler'a, w którym budujemy uszeregowanie od końca. W każdym kroku algorytmu rozważamy te zadania, bez uszeregowanych następników, i wybieramy te które w danym momencie generuje najmniejszy koszt.

4.3 $1|\textbf{out-tree}|\sum w_j C_j$

W tym problemie mamy jedną maszynę i n zadań nieprzerywalnych z czasami przetwarzania p_j oraz wagami. Trik aby rozwiązać ten problem, to łączyć kolejno zadania o najmniejszej wartości kryterium $(\frac{w_j}{p_j})$ z swoimi poprzednikami, pamiętając w poprzednikach o kolejności łączenia.

4.4 $1|\textbf{in-tree}|\sum w_j C_j$

Ten problem mapuje się 1-1 do problemu poprzedniego, wystarczy tylko odwrócić krawędzie i nadać zadaniom wagi przeciwne.