

### Zadanie nr 3 i 4

Opracowanie i implementacja algorytmów dla problemu  $FP||C_{\max}$ . Zgodnie z notacją trójpolową  $F$  oznacza **system przepływowy (taśmowy)** (ang. *flow shop*) – każde zadanie wymaga wykonania takiej samej liczby operacji w kolejnych stadiach; kolejność wykonywania poszczególnych operacji jest taka sama dla każdego zadania; w każdym stadium jest jeden procesor. Jeżeli kolejność wykonywania zadań na każdym z procesorów jest taka sama, wówczas system nazywamy przepływowym *permutacyjnym* i dodajemy literkę „P” (tak jak w analizowanym problemie).

#### Sformułowanie problemu:

Dany jest zbiór  $n$  zadań  $J = \{1, \dots, n\}$  i  $m$  procesorów  $M = \{M_1, \dots, M_m\}$ . Każde zadanie  $j$  składa się ze zbioru  $m$  operacji  $O = \{O_{1,j}, \dots, O_{m,j}\}$ . Każda operacja  $O_{z,j}$  musi zostać wykonana na procesorze  $M_z$  ( $z=1, \dots, m$ ). Ponadto, zakłada się, że operacja  $O_{z+1,j}$  może rozpocząć się dopiero w momencie, kiedy operacja  $O_{z,j}$  jest zakończona. W przypadku, gdy każdy z procesorów musi wykonywać operacje w takiej samej kolejności problem nazywamy permutacyjnym i w niniejszym rozdziale skoncentrujemy się wyłącznie na problemach permutacyjnych. Założono również, że każdy procesor może w danym momencie wykonywać co najwyżej jedną operację i nie istnieją żadne ograniczenia kolejnościowe pomiędzy zadaniami. Operacje są niepodzielne i dostępne w chwili 0 na procesorze  $M_1$ . W dalszej części instrukcji będziemy używać określenia zadanie  $j$  na procesorze  $M_z$  zamiast operacja  $O$ . Czas wykonywania zadania  $j$ , tj.  $p_j^{(z)}$ , wykonywanego na procesorze  $M_z$  ( $z = 1, \dots, m$ ) opisany jest liczbą rzeczywistą. Dla  $m$ -procesorowego *permutacyjnego* problemu przepływowego uszeregowanie zadań na procesorach może zostać jednoznacznie zdefiniowane jako permutacja  $\pi$ . Zatem dla każdego zadania  $\pi(i)$ , tj. uszeregowanego na pozycji  $i$  w permutacji  $\pi$  możemy określić czas zakończenia jego wykonywania  $C_{\pi(i)}^{(z)}$  na procesorze  $M_z$ :

$$C_{\pi(i)}^{(z)} = \max \left\{ C_{\pi(i)}^{(z-1)}, C_{\pi(i-1)}^{(z)} \right\} + p_{\pi(i)}^{(z)}$$

Gdzie  $C_{\pi(1)}^{(0)} = C_{\pi(0)}^{(z)} = 0$  dla  $z = 1, \dots, m$  i  $C_{\pi(i)}^{(1)} = \sum_{l=1}^i p_{\pi(l)}^{(1)}$  jest czasem zakończenia wykonywania zadania uszeregowanego na pozycji  $i$  w permutacji  $\pi$  na  $M_1$ .

Celem jest znalezienie takiego uszeregowania zadań  $\pi$  na procesorach, które minimalizuje kryterium długości uszeregowania  $C_{\max} \triangleq \max_{j \in J} \{C_j\}$ . W analizowanym problemie:

$$C_{\max} = C_{\pi(n)}^{(m)}.$$

Problem  $FP||C_{\max}$  jest problemem silnie NP-trudnym dla  $m \geq 3$  (dla liczby maszyn równej co najmniej 3).

Benchmarki (instancje testowe) są dostępne np. tu: <http://staff.iiar.pwr.wroc.pl/wojciech.bozejko/benchmarks.htm> (pierwszy link od góry)

Zadania do zrobienia w ramach zadania nr 3:

Na ocenę 3.0	Przegląd zupełny + NEH
Na ocenę 3.5	Algorytm Johnsona dla $m=2$
Na ocenę 4.0	FNEH (NEH z akceleracją)
Na ocenę 4.5	Wersja podstawowa algorytmu podziału i ograniczeń (z prostym LB)
Na ocenę 5.0	Wersja podstawowa dwóch wybranych algorytmów metaheurystycznych opartych o

Uwaga, oceniam też sprawozdanie (jego zawartość), w tym sposób wykonania eksperymentu numerycznego. Ze względu na to, że część zajęć odpadła, to punkty 4.5 i 5.0 są zamienne na 4.5, na 5.0 należy zrobić oba.

Przykładowe algorytmy: symulowane wyżarzanie, tabu search, przeszukiwanie ze zmiennym sąsiedztwem, algorytm akceptacji progu.

Uwaga, oceniam też sprawozdanie (jego zawartość), w tym sposób wykonania eksperymentu numerycznego.

Algorytm Johnsona jest algorytmem optymalnym dla problemu z dwoma maszynami [1]. Jego złożoność obliczeniowa to  $O(n \log n)$ . Opis znajduje się poniżej.

Algorytm NEH (technika wstawień/wcięć), zaproponowany w [2] jest najlepszym konstrukcyjnym algorytmem heurystycznym dla klasycznego problemu przepływowego. Dotychczas nie udało się skonstruować innego algorytmu, który w tym samym czasie działania dostarcza rozwiązania charakteryzującego się porównywalną jakością. Rozpoczyna on działanie od pewnego rozwiązania początkowego  $\pi_{init}$ , które określa kolejność zadań. W wersji dla klasycznego problemu, zadania są uszeregowane zgodnie z niemalejącą sumą czasów wykonywania operacji. Następnie zadania z  $\pi_{init}$  są kolejno wstawiane do rozwiązania końcowego  $\pi^i$ , na taką pozycję, aby zminimalizować wartość kryterium  $C_{\max}(\pi^i)$ . Złożoność obliczeniowa klasycznej wersji algorytmu to  $O(n^3m)$ . Istnieje metoda szybkiego liczenia kryterium po zamianie pozycji dwóch zadań, przy której uwzględnieniu mamy złożoność obliczeniową  $O(n^2m)$ .

Literatura:

[1] S. M. Johnson. Optimal two-and-three-stage production schedules. *Naval Research Logistic*, 1:61–68, 1954

[2] M. Nawaz, Jr E. E. Enscore, I. A. Ham. A heuristic algorithm for  $m$ -machine,  $n$ -jobs Flow-shop sequencing problem. *Omega*, 11:91–95, 1983

---

### Algorytm 1 Johnson

---

- 1:  $C_{\max}^* = \infty$ ,  $\pi^* = \emptyset$
  - 2: Skonstruuj zbiory:  $J_L = \{j : j \in J \wedge p_j^{(1)} < p_j^{(2)}\}$  i  $J_R = \{j : j \in J \wedge p_j^{(1)} \geq p_j^{(2)}\}$
  - 3: Skonstruuj uszeregowanie  $\pi_L$  z zadań  $J \in J_L$  według niemalejących wartości  $p_j^{(1)}$
  - 4: Skonstruuj uszeregowanie  $\pi_R$  z zadań  $J \in J_R$  według nierosnących wartości  $p_j^{(2)}$
  - 5: Konkatenacja  $\pi^* = \pi_L \cap \pi_R$  jest optymalnym rozwiązaniem.
- 

---

### Algorytm 2 NEH

---

- 1:  $C_{\max}^* = \infty$ ,  $\pi^* = \emptyset$
  - 2: Ustal kolejność zadań  $\pi_{init}$
  - 3: Pobierz pierwsze zadanie  $j$  z  $\pi_{init}$
  - 4: Wstaw  $j$  na taką pozycję w  $\pi^*$  dla której wartość  $C_{\max}(\pi^*)$  jest minimalna
  - 5: Usuń  $j$  z  $\pi_{init}$
  - 6: Jeżeli  $\pi_{init} \neq \emptyset$  Idź do Krok 4
  - 7: Permutacja  $\pi^*$  jest danym rozwiązaniem.
-