

Algorytm Neh z akceleracją

**Autor:** Jan Ploska

**Kierunek**: Informatyczne Systemy Automatyki

**Semestr**: VI

**Prowadzący**: dr. hab. inż. Mariusz Makuchowski

**Opis problemu**: Algorytm NEH (Nawaz-Enscore-Ham) jest heurystyczną metodą planowania produkcji, używaną do rozwiązywania problemów harmonogramowania przepływu prac w systemach produkcyjnych typu flow shop. W takim systemie zadania muszą przejść przez określoną sekwencję maszyn w tym samym porządku, a celem jest zminimalizowanie całkowitego czasu wykonania wszystkich zadań, nazywanego czasem przepływu.

**Cel algorytmu Neh**: Problem harmonogramowania przepływu prac (flow shop scheduling problem), polega na znalezieniu optymalnego porządku wykonywania zadań na zestawie maszyn, tak aby zminimalizować całkowity czas produkcji (makespan). Ten problem jest trudny do rozwiązania optymalnie (należy do problemów NP-trudnych), zwłaszcza gdy mamy do czynienia z dużą liczbą maszyn i zadań.

**Działanie algorytmu Neh**:

1. Sortowanie zadań: Na początku wszystkie zadania są sortowane malejąco według sumarycznego czasu pracy na wszystkich maszynach. A computer code with colorful text

   Description automatically generated
2. Konstrukcja harmonogramu: Zadania są dodawane jedno po drugim do harmonogramu początkowego, próbując je umieszczać w każdej możliwej pozycji. A computer screen shot of a program code

   Description automatically generated
3. Wybór pozycji: Dla każdego nowo dodanego zadania wybierana jest taka pozycja w harmonogramie, która minimalizuje aktualny czas przepływu.
4. A computer screen with text

   Description automatically generated

**Algorytm Neh z akceleracją(QNeh):** tawersja algorytmu NEH, która wprowadza dodatkowe optymalizacje w celu przyspieszenia procesu obliczeniowego. Bazuje na podstawowej zasadzie NEH, czyli sortowaniu zadań według ich sumarycznego czasu pracy i sekwencyjnym dodawaniu ich do harmonogramu w celu minimalizacji maksymalnego czasu zakończenia (makespan). QNEH wprowadza jednak dodatkowe tablice obliczeniowe, które pozwalają na szybsze ocenienie wpływu wstawienia nowego zadania w każdą możliwą pozycję.

**Działanie algorytmu QNeh**:

1. Sortowanie zadań: Na początku wszystkie zadania są sortowane malejąco według sumarycznego czasu pracy na wszystkich maszynach.
2. Zadania są dodawane jedno po drugim do wstępnego harmonogramu. Przy każdym dodaniu korzystam z dwóch pomocniczych tablic (cmax\_tab z przodu i reversed\_cmax\_tab z tyłu), które są generowane tylko raz dla aktualnego stanu harmonogramu. Te tablice reprezentują akumulacyjne czasy zakończenia pracy na każdej maszynie dla kolejnych zadań, obliczane od początku i od końca harmonogramu. Pozwala to na szybką ocenę wpływu wstawienia nowego zadania na każdą możliwą pozycję w harmonogramie, bez potrzeby wielokrotnego obliczania tych samych wartości. Dla każdego nowo dodanego zadania obliczany jest maksymalny czas zakończenia (cmax) dla każdej potencjalnej pozycji w harmonogramie, korzystając z wcześniej wspomnianych tablic, co znacząco przyspiesza i usprawnia proces wyboru optymalnej pozycji zadania. A computer code on a black background

   Description automatically generated
3. Wybór pozycji: Dla każdego nowo dodanego zadania wybierana jest taka pozycja w harmonogramie, która minimalizuje aktualny czas przepływu.

**Porównanie czasów działania algorytmu Neh i Qneh:** czasy podane są w sekundach.

A table with numbers and a green border

Description automatically generated

**Wnioski:**

* Moja implementacja QNeh znacząco przyśpieszyła działanie algorytmu
* Kod napisany jest w pythonie i czas działania dla wszystkich 120 instancji jest powyżej 12 sekund
* Niepotrzebnie za każdym razem buduje harmonogram od samego początku.