Zadanie 2 - permutacyjny problem przepływowy (ang. *flowshop)*

Dominik Żelazny, mgr inż.

26 luty 2013

1 Opis zadania

Zadanie dotyczy permutacyjnego problemu przepływowego (ang. *Permutation Flow Shop Scheduling Problem - PFSSP), który opisany został w trakcie laboratoriów. Do zadania dostępne są materiały (dział Materiały) w formacie PDF oraz dodatkowe aplikacje (dział Zadania), pozwalające na przetestowanie pliku wynikowego.*

1.1 Podział zadania

Zadanie podzielone zostało na dwie części. W celu otrzymania maksymalnej liczby punktów, należy wykonać obie z poniższych. Wykonanie wyłącznie części pierwszej da co najwyżej połowę punktów.

1.1.1 Część 1 - algorytm NEH

Implementacja opisanego na pierwszych zajęciach laboratoryjnych z algorytmu NEH (akronim od nazwisk twórców: Nawaz, Enscore, Ham) dla problemu PFSSP. Czas działania powinien być zbliżony do czasu generowanego przez aplikację umieszczoną na stronie.

1.1.2 Część 2 - akceleracja

*Opisany zostanie podczas drugich zajęć laboratoryjnych dotyczących problemu PFSSP. Podobnie jak w przypadku części pierwszej, czas działania powinien być zbliżony do aplikacji porównawczej umieszczonej na stronie prowadzącego (aplikacja **algorytm_NEH.exe**).*

2 PARAMETRY I FUNKCJE CELU

W przypadku podstawowym, którym zajmiemy się na zajęciach, operacje w zadaniu przypisany mają czas wykonania $p_{j,i}$ oraz kolejność wykonywania na maszynach. Ta ostatnia jest stała i taka sama dla wszystkich zadań, tzn. numer operacji odpowiada numerowi maszyny na której jest wykonywana. W przypadku problemu PFSSP posłużymy się wyłącznie jedną funkcją celu, opisaną poniżej.

2.1 C_{max} - KRYTERIUM MAKSYMALNEGO CZASU ZAKOŃCZENIA WYKONYWANIE (ANG. *makespan*).

Kryterium to jest najczęściej pojawiającym się kryterium w literaturze, ze względu na zbieżność modelu z problemami spotykanymi w przemyśle. Reprezentowane jest wzorem:

$$C_{max} = \max_{1 \leq j \leq m} C_{j,\pi_n}. \tag{2.1}$$

Dla problemu PFSSP można go również zapisać w postaci:

$$C_{max} = C_{m,\pi_n}, \tag{2.2}$$

gdzie:

$$C_{j,\pi_i} = max(C_{j-1,\pi_i}, C_{j,\pi_{i-1}}) + p_{j,\pi_i}, \tag{2.3}$$

$$C_{0,\pi_i} = 0, \tag{2.4}$$

$$C_{j,0} = 0. \tag{2.5}$$

Dla $i \in \{1,...,m\}$ oraz $j \in \{1,...,n\}$. Dobrą praktyką jest w tym wypadku wyzerowanie brzegów macierzy (gdy wartość x lub y jest równa zero) i wprowadzanie danych od 1 do n, zamiast od 0 do $n-1$.

3 Pliki wejściowe i wyjściowe

Dane do zadań umieszczono na stronie.

3.1 Dane wejściowe

*Umieszczone w pliku **bench_fs.txt** w następującej postaci (pierwsza instancja Taillarda dla problemu PFSSP):*

```
ta001 - nazwa instancji
20 5 - rozmiar instancji postaci: liczba_zadań liczba_maszyn
   1 54    2 79    3 16    4 66    5 58  - dane postaci: nr_maszyny czas_wykonania
   1 83    2  3    3 89    4 58    5 56
   1 15    2 11    3 49    4 31    5 20
   1 71    2 99    3 15    4 68    5 85
   1 77    2 56    3 89    4 78    5 53
   1 36    2 70    3 45    4 91    5 35
   1 53    2 99    3 60    4 13    5 53
   1 38    2 60    3 23    4 59    5 41
   1 27    2  5    3 57    4 49    5 69
   1 87    2 56    3 64    4 85    5 13
   1 76    2  3    3  7    4 85    5 86
   1 91    2 61    3  1    4  9    5 72
   1 14    2 73    3 63    4 39    5  8
   1 29    2 75    3 41    4 41    5 49
   1 12    2 47    3 63    4 56    5 47
   1 77    2 14    3 47    4 40    5 87
   1 32    2 21    3 26    4 54    5 58
   1 87    2 86    3 75    4 77    5 18
   1 68    2  5    3 77    4 51    5 68
   1 94    2 77    3 40    4 31    5 28
```

3.2 Dane wyjściowe

*Plik wyjściowy (**output.txt**) powinien zawierać 120 kolejnych uszeregowań, wygenerowanych przez algorytm NEH. Zadania w uszeregowaniu oddzielić należy biały znakiem, kolejne permutacje znakiem nowej linii. Np. dla 3 instancji i 5 zadań w każdej plik wyjściowy wyglądał by jak następuje:*

```
1 2 3 4 5
2 1 3 5 4
5 4 3 2 1
```