Schemat Aproksymacyjny dla $P_2||C_{max}$

Autorzy:
Jacek Wieczorek (181043)

Prowadzący : mgr inż. Karolina Mokrzysz
Poniedziałek TP 11.00 - 13.00

Wydział Elektroniki
III rok

8 stycznia 2012

1 Opis problemu

Celem laboratorium jest zaimplementowanie algorytmu szeregowania n zadań na 2 równolegle działających procesorach - $P_2||C_{max}$ za pomocą wilomianowego schematu aproksymacyjnego *PTAS*.

2 PTAS

Wielomianowy schemat aproksymacji (ang. Polynomial-time approximation scheme, w skrócie PTAS) to algorytm aproksymacyjny, który pozwala na uzyskanie dowolnie dobrego rozwiązania przybliżonego danego problemu optymalizacyjnego, i którego złożoność czasowa jest wielomianowa dla każdej żądanej dokładności.

Algorytm A jest wielomianowym schematem aproksymacji dla problemu Π jeśli spełnione są następujące warunki:

- dla każdego odpowiedniego ε A jest algorytmem ε-aproksymacyjnym dla Π,
- dla każdego odpowiedniego ε złożoność czasowa A jest wielomianowa ze względu na rozmiar instancji problemu podanej na wejściu A.

3 Opis algorytmu

- Krok 1 : Posortuj zadania malejąco wg czasu ich wykonania
- Krok 2 : Przyporządkuj k-pierwszych zadań w sposób optymalny
- Krok 3 : Pozostałe $n - k$ zadań przyporządkuj za pomocą algorytmu *LPT*

4 Dowód wielomianowości

- Sortowanie: $O(nlogn)$
- Przyporządkowanie 1 : przegląd zupełny przy pomocą wektora binarnego - 2^k

- Przyporządkowanie $LPT : O(n-k)$

Złożóność obliczeniowa : $O(nlogn + 2^k)$

Dokładność : $\varepsilon = \frac{1}{1+\lfloor k/2 \rfloor}$

Wynika z tego, iż k = $O(\frac{1}{\varepsilon})$

Co daje nam złożoność obliczeniową algorytmu : $O(nlogn + 2^{\frac{1}{\varepsilon}})$

5 Implementacja

```
1
    public void ptasFunction()
    {
        bool [] tmp = new bool[tasksCount];

        Array.Sort(tasks);
        Array.Reverse(tasks);

        if (!(eps > 1.0/3.0))
        {
11          best = getTime(tmp, k);
            while (getPermutation(tmp))
            {
                int tmpTime = getTime(tmp, k);
                if (best > tmpTime)
                {
                    best = tmpTime;
                    Array.Copy(tmp, optimal, k);
                }
            }
21      }
        else
        {
            k = 0;
        }
        for (int i = k; i < tasksCount; i++)
        {
            if (!freeCpu(i + 1))
            {
                optimal[i] = true;
31          }
        }
    }


    private bool getPermutation(bool[] tab)
    {
        for(int i=0; i < k; i++)
        {
            if (tab[i])
41          {
```

```
                tab[i] = false;
            }
            else
            {
                tab[i] = true;
                return true;
            }
        }
        return false;
51  }
```

6 Testy

W celu przeprowadzenia dokłądnej analizy wykonanych zostało szereg testów badających czas wykonania algorytmu w zależności od parametru n i ε. W przypadku badania zcasu wykonania kazdej instancji wykonano po 3 testy, a prezentowany wynik jest ich średnią arytmetyczną. Mimo iż program posiada graficzny interfejs użytkownika, moduł testowy wykonany został jako aplikacja konsolowa.