Sterowanie procesami dyskretnymi

Temat Laboratorium	**Zajęcia**
Metoda rozwiązywania problemu przepływowego - algorytm NEH	*2*

Skład grupy laboratoryjnej

Filip Dyba, Agata Smoląg

Prowadzący	**Data**
Mgr inż. Teodor Nižyński	*30 marca 2019*

1 Opis problemu

Zadaniem programu jest rozwiązanie problemu przepływowego. Mamy dany zbiór zadań n które wykonując się na m maszynach pod warunkiem, że wykonują się nieprzerwanie, w takiej samej kolejności. Rozwiązanie jest optymalne jeżeli czas zakończenia zadań jest najmniejszy.

2 Metoda rozwiązania

2.1 Algorytm NEH

Do rozwiązania problemu przepływowego stosujemy algorytm NEH. Składa się on z czterech kroków:

- wyznaczenie priorytetów (suma wszystkich operacji danego zadania)
- sortowanie nierosnąco zadań
- wybór zadania o największym priorytecie
- wstawienie zadań na każdej pozycji i wybór tego o najmniejszej wartości C_{max}

2.2 Algorytm NEH - modyfikacja

W celu uzyskania lepszych wyników zaimplementowano algorytm NEH z akceleracją oraz wprowadzono kilka modyfikacji. Modyfikacje polegają na wyborze elementu oraz dokonaniu wstawienia go na wszystkich możliwych miejscach. Wybieramy to dla którego wartość C_{max} jest najmniejsza. Różnią się one sposobem wyboru tego elementu. Wykonano 4 modyfikacje, wybór:

- Zadanie zawierające najdłuższą operacje na ścieżce krytycznej.
- Zadanie zawierające największą sumę operacji wchodzących w ścieżkę krytyczną.
- Zadanie zawierające największą liczbę operacji wchodzących w ścieżkę krytyczną
- Zadanie, którego usunięcie spowoduje największe zmniejszenie wartości C_{max}

Listing 1: Algorytm NEH

```
1      liczba_maszyn = Maszyny[1]
2      Kolejnosc = []
3      Kolejnosc_naj = []
4      Cmaxmin = 9999999
5      ListaZadan = [liczba_zadan, liczba_maszyn]
6      Priorytety = ObliczPriorytet(Maszyny)
7      KolejnoscPriorytetow = SortujPriorytetem(Priorytety)
8      for i in range(liczba_zadan):
9          Cmaxmin = 999999
10         zadania = WezZadanie(Maszyny, KolejnoscPriorytetow[i])
11         ListaZadan.extend(zadania)
12         ListaZadan[0] = i+1
13         for j in range (i+1):
14             Kolejnosc.insert(j, i+1)
15             dlugosc_trwania=Cmax(ListaZadan, Kolejnosc)
16             if dlugosc_trwania<Cmaxmin:
```

```
17                  Kolejnosc_naj = Kolejnosc[:]
18                  Cmaxmin = dlugosc_trwania
19              Kolejnosc.pop(j)
20          Kolejnosc = Kolejnosc_naj[:]
21      for i in range(len(Kolejnosc_naj)):
22          Kolejnosc_naj[i] = KolejnoscPriorytetow[Kolejnosc[i]-1]
23
24      return Cmaxmin, Kolejnosc_naj
```

3 Eksperymenty obliczeniowe

Celem testów jest zbadanie, na ile rozwiązania wygenerowane przez Algorytm NEH są lepsze od jego modyfikacji oraz porównanie jego działania z algorytmem Johnsona oraz przeglądem zupełnym.

Jako miarę jakości algorytmu przyjęto względną poprawę wartości C^{NEH} funkcji celu otrzymanej za pomocą algorytmu NEH przez zastawanie modyfikacji generującej wartość funkcji celu C^A.

$$\Phi^A = 100\%(C^{NEH} - C^A)/C^{NEH} \tag{1}$$

Wszystkie wyniki porównujące modyfikacje algorytmu NEH zostały zebrane i przedstawiono w tabeli 2. W tabeli 1 znajdują się względne wartości poprawy dla modyfikacji algorytmu NEH.

mod4	mod1	mod2	mod3	
-0,075471698	-0,301886792	0,679245283	-0,301886792	Ta004 – 20/5
-0,963523744	0,688231246	1,032346869	1,032346869	Ta016 20/10
2,211097205	1,752190238	1,293283271	1,293283271	Ta025 20/20
0	0	0	0	Ta031 50/5
0,334896182	-0,870730074	-0,803750837	0,435365037	Ta043 50/10
1,587301587	-0,07558579	-0,025195263	0,100781053	Ta054 50/20
-0,517340487	0,038321518	0,076643035	0	Ta063 100/5
-0,055015588	0,275077939	-0,293416468	0,421786173	Ta072 100/10
0,687968201	-0,642103654	0,107017276	-0,183458187	Ta081 100/20
-0,181405896	-0,371882086	-0,272108844	-0,43537415	Ta093 200/10
1,317928969	0,513478819	0,145485665	-0,051347882	Ta105 200/20

Tablica 1: Porównanie względnych wartości poprawy

	algorytm neh		z akceleracja		z mod4		mod1		mod2		mod3	
Ta004 – 20/5	0,007164001	1325	0,001497984	1325	0,008350611	1326	0,016020298	1329	0,00413847	1316	0,009791136	1329
Ta016 20/10	0,011402607	1453	0,002572298	1453	0,016391277	1467	0,008483887	1443	0,006673098	1438	0,013319492	1438
Ta025 20/20	0,149764538	2397	0,029443502	2397	0,086084366	2344	0,016011238	2355	0,015580654	2366	0,012372255	2366
Ta031 50/5	0,146720648	2733	0,009747744	2733	0,109884739	2733	0,025779247	2733	0,026609898	2733	0,046226978	2733
Ta043 50/10	0,187574387	2986	0,016611814	2986	0,19841671	2976	0,066185713	3012	0,074067831	3010	0,053500652	2973
Ta054 50/20	0,51346302	3969	0,065459967	3969	0,4847157	3906	0,108616352	3972	0,132687807	3970	0,080178738	3965
Ta063 100/5	0,710914373	5219	0,034420252	5219	0,750704527	5246	0,116067648	5217	0,125869274	5215	0,08395648	5219
Ta072 100/10	1,454129457	5453	0,068973541	5453	1,510807991	5456	0,232845068	5438	0,186050653	5469	0,165387154	5430
Ta081 100/20	3,068160295	6541	0,137150526	6541	3,047127724	6496	0,350482225	6583	0,325977087	6534	0,344335318	6553
Ta093 200/10	11,07620621	11025	0,270633459	11025	11,63428903	11045	0,674937487	11066	0,676392078	11055	0,675784349	11073
Ta105 200/20	21,62185144	11685	0,534049988	11685	23,00531292	11531	1,319253445	11625	1,337118387	11668	1,343390226	11691

Tablica 2: Porównanie modyfikacji

Porównanie wyników dla losowych danych dla wszystkich algorytmów zostały umieszczone w tabeli 3.

liczba zadań	2	3	4	5	6	7	8	9	10	
Algorytm johnsona	3,03E-05	1,81E-05	1,26E-05	1,60E-05	3,86E-05	2,88E-05	2,86E-05	3,58E-05	3,70E-05	czas
	251	339	236	374	350	509	536	600	537	cmax
Przegląd zupełny	3,86E-05	3,29E-05	5,39E-05	0,000389814	0,00315547	0,017202616	0,120973349	1,237499475	12,86993837	czas
	251	339	236	374	350	509	536	600	537	cmax
Algorytm NEH	0,000139475	8,61E-05	8,56E-05	0,000175476	0,000195265	0,000367641	0,000318527	0,000655651	0,000549078	czas
	251	339	236	374	350	509	536	621	537	cmax
z akceleracja	6,03E-05	6,75E-05	5,75E-05	0,000136852	0,000130892	0,000236988	0,000185251	0,000375509	0,00027585	czas
	251	339	236	374	350	509	536	621	537	cmax
z modyfikacja	8,01E-05	0,000150681	0,000148773	0,00036335	0,000647783	0,000823736	0,000659704	0,001363516	0,001081944	czas
	251	339	236	374	350	509	536	600	537	cmax

Tablica 3: Tabela porównania algorytmów

4 Wnioski

Algorytm NEH zwracał dla problemu 3-maszynowego takie same wartości C_{max} jak algorytm Johnsona. Algorytm Johnosona wykonywał się w nieznacznie krótszym czasie niż algorytm NEH. Dla małej liczadań algorytm NEH z akceleracją wykonywał się nieco dłużej niż wersja podstawowa. Podobnie miała się rzecz z algorytmem z modyfikacją nr 4.

Dla większej liczby zadań algorytm NEH z akceleracją wykonywał się zdecydowanie szybciej od podstawowej wersji. Porównanie czasów obliczeń przez poszczególne algorytmy pokazała, że złożoność obliczeniowa podstawowej wersji algorytmu wynosi $O(n^3m)$, natomiast wersja z akceleracją posiada złożoność obliczeniową $O(n^2m)$. Potwierdza to opis teoretyczny algorytmów.

Do algorytmu NEH z akceleracją wprowadzono modyfikacje, które czasami pozwalały uzyskać krótsze czasy C_{max}. Nie działo się tak jednak dla każdego przypadku - niekiedy uzyskiwany wynik był gorszy. Uzyskiwane zmiany wartości funkcji kryterialnej nigdy nie przekroczyły 2%, zatem potencjalne zyski były bardzo niewielkie. Ponadto często były okupione większym kosztem poświęconego czasu, zwłaszcza w przypadku modyfikacji nr 4. Wnioskuje się, że zaproponowane modyfikacje nie poprawiają wydatnie jakości szeregowania zadań, więc ich stosowanie jest nieopłacalne.