Grupa: 01.04.2019   
Patrycja Bugaj 235276  
Kamil Goś 235184

Sprawozdanie z zajęć laboratoryjnych: Sterowanie procesami dyskretnymi - zajęcia 2.

1. Zadanie do wykonania

Celem zajęć było zapoznanie się z popularną metodą rozwiązywania problemu przepływowego – algorytmem NEH oraz rozwinięcie zdolności analitycznych - porównanie wyników algorytmu NEH z podejściem z poprzednich zajęć.

1. Przebieg ćwiczenia

Po zaimplementowaniu podstawowej wersji algorytmu NEH przetestowaliśmy go z pomocą wszystkich instancji ta\_\_\_ i wyniki Cmax zgadzały się z oczekiwanymi wynikami. Poniżej   
w tabeli przedstawiamy porównanie wyników uzyskanych z naszego programu oraz   
z programu NehDemo dla 10 losowych instancji.

|  |  |  |
| --- | --- | --- |
| Instancja | Cmax uzyskane z naszego programu | Cmax uzyskane z programu NehDemo |
| ta001 | 1286 | 1286 |
| ta008 | 1223 | 1223 |
| ta017 | 1562 | 1562 |
| ta029 | 2320 | 2320 |
| ta038 | 2721 | 2721 |
| ta056 | 3914 | 3914 |
| ta074 | 5929 | 5929 |
| ta095 | 10645 | 10645 |
| ta103 | 11852 | 11852 |
| ta112 | 27232 | 27232 |

Tabela 1: Porównanie Cmax otrzymanego z naszego programu i z programu NehDemo

Następnie porównaliśmy czas wykonywania się przeglądu zupełnego, algorytmu Johnsona oraz algorytmu NEH, czego wyniki widać w poniższej tabeli.

|  |  |  |  |
| --- | --- | --- | --- |
| Ilość maszyn | Przegląd zupełny | Algorytm Johnsona | Algorytm NEH |
| 2 | 0.002024s | 0.000203s | 0.000167s |
| 3 | 0.002126s | 0.000434s | 0.000199s |

Tabela 2: Porównanie rzeczywistego czasu wykonywania się algorytmów dla 2 i 3 maszyn

Podobne porównywanie możemy przeprowadzić ze względu na Cmax

|  |  |  |  |
| --- | --- | --- | --- |
| Ilość maszyn | Przegląd zupełny | Algorytm Johnsona | Algorytm NEH |
| 2 | 20 | 20 | 20 |
| 3 | 32 | 32 | 32 |

Tabela 2: Porównanie czasu Cmax

Jak widać, dla takiej małej ilości zadań i maszyn wszystkie algorytmy radzą sobie tak samo dobrze.

Wykorzystane dane w powyższych porównaniach:

* 2 maszyny:

|  |  |  |
| --- | --- | --- |
|  | Maszyna 1 | Maszyna 2 |
| Zadanie 1 | 5 | 1 |
| Zadanie 2 | 6 | 2 |
| Zadanie 3 | 4 | 3 |
| Zadanie 4 | 2 | 6 |
| Zadanie 5 | 1 | 7 |

* 3 maszyny

|  |  |  |  |
| --- | --- | --- | --- |
|  | Maszyna 1 | Maszyna 2 | Maszyna 3 |
| Zadanie 1 | 1 | 3 | 8 |
| Zadanie 2 | 9 | 3 | 5 |
| Zadanie 3 | 7 | 8 | 6 |
| Zadanie 4 | 4 | 8 | 7 |

1. Akceleracja algorytmu NEH

W tym kroku zaimplementowano akcelerację algorytmu neh, która ma za zadanie skrócić czas wykonywania algorytmu przy jednoczesnym utrzymaniu jakości wyników. Algorytm został przetestowany dla wszystkich instancji. Poniżej przedstawiono 10 przykładowych wyników.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instancja** | **Rozmiar instancji** | **Algorytm** | **Cmax** | **Czas wykonywania [s]** | **Przyśpieszenie [krotność]** |
| data.012 | 20x10 | Klasyczny | 1729 | 0,023 | 2,55 |
| Akceleracja | 1729 | 0,009 |
| data.056 | 50x10 | Klasyczny | 3277 | 0,342 | 5,6 |
| Akceleracja | 3277 | 0,058 |
| data.080 | 100x10 | Klasyczny | 5918 | 2,533 | 12,35 |
| Akceleracja | 5918 | 0,205 |
| data.096 | 100x10 | Klasyczny | 10458 | 18,956 | 24,9 |
| Akceleracja | 10458 | 0,761 |
| data.100 | 200x10 | Klasyczny | 10807 | 19,166 | 25,08 |
| Akceleracja | 10807 | 0,764 |
| data.105 | 200x20 | Klasyczny | 11685 | 37,881 | 26,6 |
| Akceleracja | 11685 | 1,424 |
| data.117 | 500x20 | Klasyczny | 26797 | 568,852 | 59,47 |
| Akceleracja | 26797 | 9,565 |

Jak widać, algorytm działa tak jak powinien. We wszystkich przypadkach czas został diametralnie zmniejszony. Warto również zauważyć, że pomimo zmniejszenia czasu wykonywania, czas Cmax nie uległ zmianie. Można również zauważyć, że przyśpieszenie rośnie wraz ze wzrostem wielkości instancji.

1. Modyfikacja algorytmu NEH

W tym kroku należało zaimplementować jedną z modyfikacji algorytmu. Wybrano modyfikację pierwszą, która polega na tym, aby na ścieżce krytycznej w grafie znaleźć operację o największym czasie wykonywania (wyłączając operację ostatnio dodaną). Następnie zadanie, w którego skład wchodzi ta operacja należy usunąć z grafu i wstawić we wszystkie możliwe miejsca jednocześnie sprawdzając, czy przyniosło to jakieś pozytywne efekty. Poniżej została przedstawiona tabela porównująca algorytm bez modyfikacji z tym z modyfikacją ze względu na Cmax.

|  |  |  |
| --- | --- | --- |
| **Instancja** | **Algorytm** | **Cmax** |
| data.080 | Bez modyfikacji | 5918 |
| Z modyfikacją | 5920 |
| data.100 | Bez modyfikacji | 10807 |
| Z modyfikacją | 10787 |
| data.101 | Bez modyfikacji | 11594 |
| Z modyfikacją | 11628 |
| data.116 | Bez modyfikacji | 26992 |
| Z modyfikacją | 26950 |

Jak widać wyniki są zróżnicowane. W niektórych przypadkach modyfikacja dała lepszy wynik, a w niektórych gorszy.

1. Wnioski

* Przegląd zupełny jest zarówno najlepszym jak i najgorszym algorytmem. Najlepszy, ponieważ zawsze zwraca najlepszą sekwencję, ale jednocześnie najgorszy, ponieważ ma złożoność wykładniczą co dyskwalifikuje ten algorytm przy większej ilości danych (zadań/maszyn).
* Algorytm Johnsona jest dobrym zamiennikiem przeglądu zupełnego dla 2 i 3 maszyn. Dla problemu trzymaszynowego działa wolniej niż dla dwumaszynowego, gdyż przy problemie trzymaszynowym w algorytmie Johnsona dodawana jest nowa operacja – tworzenie z problemu trzymaszynowego problemu dwumaszynowego, poprzez zsumowanie czasu wykonywania zadań dla pierwszej i drugiej oraz drugiej i trzeciej maszyny. Złożoność czasowa tego algorytmu wynosi
* algorytm NEH jest najszybszy z uwzględnianych algorytmów. Dla naiwnej implementacji złożoność wynosi Przy implementacji QuickNEH złożoność wynosi