**Projektowanie Efektywnych Algorytmów**

Jednoprocesorowy problem szeregowania zadań

Metoda Tabu Search

Autor:

Nr indeksu:

Termin oddania:

Prowadzący:

Termin zajęć:

Kierunek: Informatyka

1. **Wstęp teoretyczny**

Pełna nazwa problemu to „Jednoprocesorowy problem szeregowania zadań przy kryterium minimalizacji ważonej sumy opóźnień zadań”.

Danych jest *n* zadań, które mają być wykonane bez przerwań przez pojedynczy procesor, mogący wykonywać co najwyżej jedno zadanie jednocześnie. Każde zadanie jest dostępne do wykonania w chwili zero, do wykonania wymaga pi > 0 jednostek czasu, ma określoną wagę wi > 0 i oczekiwany termin zakończenia wykonywania di > 0. Zadanie *i* jest spóźnione, jeżeli zakończy się wykonywać po swoim terminie d*i*, miarą tego opóźnienia jest Ti = max(0, Ci – di), gdzie Ci to termin zakończenia zadania *i*. Problem polega na znalezieniu takiej kolejności zadań aby zminimalizować kryterium .

1. **Sposób przeprowadzenia eksperymentu**
   1. **Metoda**

Metoda Tabu Search (TSM) wykorzystywana jest do otrzymywania wyników optymalnych, lub niewiele różniących się od optymalnych dla problemów optymalizacyjnych. Ideą algorytmu jest przeszukiwanie przestrzeni możliwych rozwiązań, stworzonej za pomocą sekwencji ruchów swap (zamiana miejscami dwóch elementów), zawierających ruchy niedozwolone (tabu).

Aby uniknąć zakleszczenia w minimum lokalnym, dokonywana jest dywersyﬁkacja poprzez sprawdzenie, czy w trakcie ostatnich k iteracji wystąpiło lepsze rozwiązanie niż dotychczasowe minimum. W przeciwnym wypadku losujemy nowe rozwiązanie. Kluczowe jest też odpowiednie wygenerowanie rozwiązania od którego zaczynamy przeszukiwanie.

Wersja wykonana w poniższym programie jako pierwsze rozwiązanie wybiera kolejność zadań na podstawie funkcji sortującej (WSPT) definiowanej jako:

wi – waga zadania o indeksie i

ci – czas wykonania zadania o indeksie i

Algorytm też sprawdza czy długość listy Tabu nie przekracza założonej przy rozpoczęciu przeszukiwania. Jeśli dochodzi do przepełnienia, usuwany jest element który znajdował się na samym jej początku.

Na podstawie wygenerowaych możliwych ruchów można sprawdzić czy ktoś z sąsiadów jest kandydatem do tego aby funkcja celu (wyliczenie sumy opóźnień) dla niego dawała lepszy rezultat. Jeżeli jest ona faktycznie lepsza, staje się wtedy maksimum globalnym.

* 1. **Założenia**
     1. Program testowy jest wykonany jako aplikacja konsolowa
     2. Aby móc porównać z wynikami uzyskanymi za pomocą algorytmu programowania dynamicznego (DPA), wykonano badanie dla takich samych wielkości problemu (badanie 1)
        + Ilość iteracji jest taka sama jak ilość zadań
        + Kryterium dywersyfikacji to *ceil(ilość zadań/5)*
        + Długość listy tabu to *ceil(ilość zadań/5)*
        + Badania zostały wykonane dla 7 wartości N zadań odpowiadającym tym wykorzystanym w przypadku DPA
        + Dla każdej wartości N wygenerowano 50 losowych instancji problemu (w sprawozdaniu podano tylko wyniki uśrednione)
     3. Implementacja została wykonano obiektowo
     4. Funkcja celu w tym algorytmie jest wyliczenie kosztu sumy opóźnień ważonej zadań. Im mniejszy koszt, tym wyższa wartość funkcji.
     5. Sprawdzono też jak bardzo odbiegał wynik TSM od wyniku właściwego dla dużych instancji (badanie 2)
        + Instancje mają wielkości 40, 50 i 100
        + Dla każdej wielkości instancji wykonano po 10 testów dla 3 różnych zestawów
        + Wynik został uśredniony
        + Parametry zostały dopasowane jak w badaniu 1

1. **Wyniki**
2. Porównanie TSM i DPA

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ilość zadań [n] | 6 | 9 | 12 | 15 | 18 | 21 | 24 | Algorytm |
| Średni czas wykonania [ms] | 0,08096 | 0,88695 | 8,94367 | 91,4503 | 920,358 | 9260,12 | 89113,9 | DPA |
| 0,05071 | 0,14537 | 0,34673 | 0,69347 | 1,04194 | 1,55352 | 2,35199 | TSM |

Tabela 1. Porównanie średniego czasu wykonania dla ilości zadań od 6 do 24 dla TSM i DPA

Wykres 1. Porównanie średniego czasu wykonania dla ilości zadań od 6 do 24 dla TSM i DPA

1. Sprawdzenie różnicy znalezionych wyników od wyników właściwych

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instancja | Wynik właściwy | Średni wynik obliczony | Błąd względny | Śr. czas [ms] |
| wt40a | 913 | 930 | 0,0186 | 13,9485 |
| wt40b | 1225 | 1230 | 0,0041 | 13,5250 |
| wt40c | 537 | 573 | 0,0670 | 13,6862 |
| wt50a | 2134 | 2167 | 0,0155 | 28,8084 |
| wt50b | 2011 | 1996 | 0,0075 | 29,2038 |
| wt50c | 2583 | 2691 | 0,0418 | 30,0161 |
| wt100a | 5988 | 6294 | 0,0511 | 311,3190 |
| wt100b | 6170 | 6182 | 0,0019 | 315,7750 |
| wt100c | 4267 | 4350 | 0,0195 | 311,2910 |

Tabela 2. Porównanie wyniku właściwego i średniego wyniku wygenerowanego

1. **Wnioski**

Na czas wykonywania przeszukiwania metodą TabuSearch najbardziej wpływa ilość iteracji, wielkość instancji nie jest więc aż tak decydująca. Algorytm wykonuje się praktycznie w każdym przypadku szybciej niż algorytm programowania dynamicznego, jak i jego wymagania pamięciowe są o wiele niższe.

Na podstawie badań można też wywnioskować, że wielkość instancji nie wpływa tak bardzo na różnicę miedzy uzyskanymi wynikami a wynikiem najbardziej optymalnym (właściwym). Wpływają na to jednak najbardziej składowe tych instancji, ponieważ generowanie pierwszego rozwiązania korzysta z tej instancji aby wybrać punkt startowy. Generator wybierze oczywiście najlepszy możliwy punkt startowy, ale nawet najlepsze ze zbioru słabych wciąż pozostaje słabym.

Mimo szybkości i niewielkich wymagań największą wadą pozostaje to co jest wspólne dla metod metaheurystycznych – znalezione rozwiązanie bardzo rzadko jest najlepszym globalnym. Algorytm ten nie bez powodu jest nazywany przeszukiwaniem – on tylko oscyluje wokół najlepszego rozwiązania, szuka go ale równie dobrze może do niego nie dotrzeć. Blokowanie ruchów w liście tabu pomaga odseparować chociaż część rozwiązań których nie warto sprawdzać, co jest pomocne w trakcie przeszukiwania.

1. **Literatura/źródła**

<http://www.lifl.fr/~derbel/ueOC/4/refs/BilKurKir07.pdf>

<http://marek.piasecki.staff.iiar.pwr.edu.pl/dydaktyka/isa/2007/Studzinski_Grzegorz.pdf>

<http://www.cs.put.poznan.pl/mradom/resources/labs/OptKomb/dziamski_4.pdf>

<http://zio.iiar.pwr.wroc.pl/pea/w5_ts.pdf>