Projektowanie efektywnych algorytmów

Autor:
Jacek Wieczorek (181043)

Prowadzący:
mgr inż. Karolina Mokrzysz

Wydział Elektroniki
III rok
Pn TP 11.15 - 13.00

13 listopada 2011

1 Opis problemu

Jednoprocesorowy problem szeregowania zadań przy kryterium minimalizacji ważonej sumy opóźnień zadań.

Danych jest n zadań (o numerach od 1 do n), które mają być wykonane bez przerwań przez pojedynczy procesor, mogący wykonywać co najwyżej jedno zadanie jednocześnie. Każde zadanie j jest dostępne do wykonania w chwili zero, do wykonania wymaga $p_j > 0$ jednostek czasu oraz ma określoną wagę (priorytet) $w_j > 0$ i oczekiwany termin zakończenia wykonywania $d_j > 0$. Zadanie j jest spóźnione, jeżeli zakończy się wykonywać po swoim terminie d_j, a miarą tego opóźnienia jest wielkość $T_j = max(0, C_j - d_j)$, gdzie C_j jest terminem zakończenia wykonywania zadania j. Problem polega na znalezieniu takiej kolejności wykonywania zadań (permutacji) aby zminimalizować kryterium $TWT = \Sigma_{j=1}^{n} w_j T_j$.

**Opis zapożyczony z opracowania dr Krysiaka.*

2 Opis algorytmu

2.1 Branch and Bound

Branch and Bound jest algorytmem służącym do znajdowania optymalnych rozwiązań dla różnych problemów optymalizacyjnych, szczególnie dla optymalizacji dyskretnej i kombinatorycznej. Polega na systematycznej iteracji po wszystkich kandydujących solucjach porblemu, z których większość jest odrzucana po niespełnienu kryterium dolnego lub górnego ograniczenia.

2.2 Użyte algorytmy

W celu rozwiązania problemu szeregowania zadań przy kryterium minimalizacji ważonej sumy opóźnień zadań zaimplementowałem 3 metody eliminacji permutacji.

2.2.1 Pierwsza metoda eliminacyjna

Pierwsza metoda eliminacyjna polega na określeniu parametru LB poprzez wyliczenie wartości początkowej kombinacji zadań. Gdy dla danej permutacji o k elementach $k < n$, zachodzi warunek $LB > \Sigma_{j=1}^{k} w_j T_j$, wtedy algorytm przechodzi o poziom wyżej w drzewie permutacji, w przeciwnym przypadku rozwiązanie jest odrzucane.

2.2.2 Druga metoda eliminacyjna

Druga metoda eliminacyjna polega na zamianie k-tego zadnia w permutacji k-elementowej, $k < n$ na początku z zadaniem $k-1$, następnie $k-2$. Jeżeli w którymś z tych przypadków suma opóźnień bedzie mniejsza niż w przypadku pierwotnej permutacji, rozwiązanie to jest odrzucane.

2.2.3 Trzecia metoda eliminacyjna

Trzecia metoda eliminacyjna polega na zamianie k-tego zadnia w permutacji k-elementowej, z elementem i-tym, $i < k$, jeżeli $p_i > p_k$. Jeżeli suma opóźnień będzie mniejsza niż w pierwotnej permutacji, to odrzucamy to rozwiązanie.

3 Implementacja i środowisko testowe

Językiem implementacji algorytmów jest $C\#$, .NET Framework 4.0, środowisko programistyczne Microsoft Visual Studio 2010. W celu maksymalnego przyśpieszenia wykonywanych algorytmów, porgram został gruntownie zbadany w $dotTrace'ie$ służacym do profilowania kodu. Wszystkie funkcje języka $C\#$ powodujące zbyt duży narzut czasowy, zostały usunięte.

Mimo, że porgram wyposażony został w interfejs graficzny, testy zostały przeprowadzone w aplikacji konsolowej (co przyśpiesza rozwiązywanie problemów). Testy dla przeglądu zupełnego i pierwszej metody eliminacyjnej zostały wykonane dla zestawów, zawierających do 11 zadań, natomiast druga i trecia metoda dla wszytskich zestawów. Dla każdego zestawu i każdej metody wykonane zostały po 3 testy, a prezentowany wynik jest ich średnią arytmetyczną. Jednostką czasu są ms.

4 Analiza wyników

lz	PZ	1E	2E	3E
5	0	0	0	0
6	0	0,66	0	0
8	27,66	58,66	1	1
9	263	561,33	5	5
10	2958,66	4880	21	16
11	34995,66	40462,66	63	46
12	∞	∞	230	143
13	∞	∞	592,66	359,66
15	∞	∞	4898,66	2533,33
17	∞	∞	61490,33	23592,66
18	∞	∞	152486,66	58963

Testy dla przegladu zupełnego i pierwszej eliminacji nie były przeprowadzone dla kompletów zawierających więcej niż 11 zadań z powodu zbyt długie czasu ich wykonywania. Na pierwszy rzut oka zaskoczeniem może wydawać się, iż pierwsza metoda eliminacyjna jest wolniejsza od przeglądu zupełnego. Dzieje się tak iż, liczony przez nas LB nie jest optymalną wartością. Powoduje to wielekrotne używanie funkcji liczącej aktualny koszt, która z moich analiz w profilerze kodu okazałą się najbardziej czasochłonna (około 60-70% czasu). Poprawe działania algorytmu można by było osiagnąć poprawiając funkcję obliczającą kryterium LB. Problem znika, gdy spojrzymy na drugą część tabeli, gdzie prezentowane są wyniki dla dwóch następnych metod, dające bardzo dobre rezultaty w przyśpieszaniu rozwiązywania algorytmu. Funkcja korzystająca ze wsyztskich 3 metod eliminacyjnych jest zdecydowanie najszybsz i najbardziej efektywna.

5 Wnioski

Algorytm $B\&B$ daje w wiekszości przypadków wymierne korzyści w przyśpieszaniu rozwiązywania problemu $sNPh$, jednak wymaga on wielu zabiegów, metod i często skomplikowanych algorytmów w celu przeprowadzania procesów eliminacji. Nie mamy też gwarancji, że układ zadań, obliczone LB czy UP, rzeczywiście przyśpieszą rozwiązanie problemu. W pesymistycznym przypadku, algorytm może dawać gorsze wyniki niż przegląd zupełny, a zakres jego używalności wynosi do około 30 zadań.