ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1982

Seria: AUTOMATYKA z. 64 Nr kol. 736

Krzysztof ZIELIŃSKI, Andrzej GOŚCIŃSKI
Instytut Informatyki
Akademia Górniczo-Hutnicza

HEURYSTYCZNY ALGORYTM SZEREGOWANIA ZADAŃ W JEDNOPROCESOROWYM KOMPUTEROWYM SYSTEMIE STEROWANIA

Streszczenie. W pracy zaproponowano heurystyczny algorytm szeregowania zadań scharakteryzowanych przez czas wykonania p, wartość linii krytycznej d i funkcję kary o postaci $c(t) = \alpha \max\{0, t-d\}$. Pokazano, że złożoność obliczeniowa tego algorytmu jest O(n). Porównano rozwiązania uzyskiwane za pomocą tego algorytmu z rozwiązaniami uzyskiwanymi przez zastosowanie algorytmu z priorytetami dynamicznymi, stosowanego w komputerach do sterowania w czasie rzeczywistym. Stwierdzono, że proponowany algorytm heurystyczny daje wyniki lepsze w sensie minimum sumy kar ważonych. Wskazuje to na celowość dostowania heurystycznego algorytmu szeregowania w jednoprocesorowych komputerach do sterowania.

1. WPROWADZENIE

Przedmiotem naszych rozważań jest problem szeregowania n zadań (algorytmów sterowania) realizowanych w jednoprocesorowym komputerze sterującym, pracującym w czasie rzeczywistym. Jak wynika z [3], celowe jest, aby każde z wykonywanych zadań P_i, i = 1,...,n było scharakteryzowane przez średni czas jego realizacji p_i oraz funkcje kary $c_i(t)$.

Funkcja $c_i(t)$ jest dana w postaci

$$c_i(t) = \alpha_i \max\{0, t-d_i\}, \tag{1}$$

gdzie:

α_i - jest znanym współczynnikiem,

d_i - natomiast wartością linii krytycznej zadania P_i.

Ponadto założymy, że zadania są realizowane bez przerwań oraz rozpoczęcie wykonania pierwszego z zadań następuje w chwili $t = t_0$.

Problem szeregowania polega na określeniu kolejności wykonania zadań, tzn. znalezienia uszeregowania π, który minimalizuje sumę

$$c = \sum_{i=1}^{n} c_{\pi(i)}(t_{\pi(i)}), \tag{2}$$

gdzie:

$$t_{\pi(1)} = t_o + P_{\pi(1)}, \tag{2}$$

$$t_{\pi(i)} = t_{\pi(i-1)} + P_{\pi(i)}, \quad i = 2,\ldots,n \tag{3}$$

oraz $\pi(i)$ oznacza i-te zadanie w planie π. Tak sformułowany problem szeregowania jest, jak pokazano w pracy [1], problemem NP-zupełnym.

W tej pracy sformułowano heurystyczny algorytm dla postawionego wyżej problemu szeregowania i wykazano, że posiada on złożoność $O(n)$ dla przypadku ustalonego zbioru zadań realizowanych w pewnych odstępach czasu. Należy podkreślić, że jest to przypadek najczęściej występujący w systemach komputerowego sterowania. Porównano jego efektywność w sensie jakości rozwiązań ze znanym algorytmem z priorytetami dynamicznymi. Pokazano wyższość proponowanego algorytmu heurystycznego. Należy podkreślić, że nie dokonano porównań z algorytmami dającymi optymalne rozwiązania (co byłoby interesujące z punktu widzenia jakości rozwiązania) ze względu na długie czasy ich przetwarzania, co wynika z ich złożoności obliczeniowej. Przedstawiono natomiast rozszerzenie prowadzonych rozważań na przypadek szeregowania dynamicznego zbioru zadań. Pokazano, że proponowany algorytm heurystyczny posiada w tym przypadku złożoność $O(n^2)$ i wskazano, że może on być użyty w trybie off-line w przypadku modyfikacji zbioru zadań realizowanego przez komputerowy system sterowania.

2. HEURYSTYCZNY ALGORYTM SZEREGOWANIA

Koncepcja proponowanego algorytmu heurystycznego polega na wprowadzeniu w zbiorze zadań P_i, $i = 1,\ldots,n$ relacji $\prec$ częściowego porządku zdefiniowanej następująco:

$$(P_j \prec P_i) \Leftrightarrow (d_j \leq \max\{d_i, P_i\} \wedge \sigma_j > \sigma_i \wedge P_j \leq P_i)$$

Jak pokazano w [3] w optymalnym planie wykonania zadań ich uporządkowanie jest zgodne z tą relacją, tzn. jeśli $P_i \prec P_j$, to także w optymalnym planie zadanie P_i poprzedza zadanie P_j.

Wprowadzenie powyższej relacji pozwala na ograniczenie zbioru zadań mogących znaleźć się na pierwszym miejscu poszukiwanego optymalnego planu wykonania zadań, do elementów maksymalnych danego zbioru zadań względem relacji $\prec$. Zadanie P_k jest elementem maksymalnym ze względu na rozważaną relację poprzedzania wtedy, gdy

$$\exists\, P_i \in \left(\{P_i\}_{i=1}^{\bar{n}} \setminus P_k\right) : P_i \prec P_k$$

Po znalezieniu zadania maksymalnego i ustawieniu go na pierwszym miejscu planu, można tę relację zastosować ponownie do pozostałego zbioru zadań. Zagadnieniem otwartym jest wybór jednego spośród kilku elementów maksymalnych, stanowiących kandydatów na dane miejsca planu. Ten krok stanowi istotny heurystyczny element rozważanego algorytmu i został rozwiązany za pomocą reguły o postaci

$$\max_{k \in M} \left\{ c_k(t) \right\}$$

gdzie M stanowi zbiór indeksów zadań będących elementami maksymalnymi w aktualnie jeszcze nie uporządkowanym zbiorze zadań.

2.1. Formalizacja algorytmu

W celu bardziej efektywnej implementacji algorytmu wykorzystano macierzowy sposób reprezentacji relacji poprzedzania określonej w zbiorze zadań P_i, $i = 1,\ldots,n$.

Para $(\{P_i\}_{i=1}, \prec)$ jest reprezentowana przez macierz A_{nxn}, której każdy element a_{ij}, $i,j = 1,\ldots,n$ jest zdefiniowany jak następuje:

$$a_{ij} = \begin{cases} 1 & \text{gdy } P_j \prec P_i, \quad i \neq j, \\ 0 & \text{w przeciwnym wypadku.} \end{cases}$$

Z przyjętych definicji wynikają następujące własności macierzy A:

(i) Zadanie P_k, które nie posiada żadnego poprzednika (może być rozpatrywane jako pierwsze w optymalnym planie wykonania zadań) posiada wszystkie elementy odpowiadającego mu wiersza równe zero, czyli

$$a_k = \sum_{i=1}^{n} a_{ki} = 0$$

(ii) Wykonanie lub dołączenie zadania P_k do zbioru zadań stwarza konieczność odpowiednio usunięcia lub dołączenia w macierzy A kolumny i wiersza o numerze k.

Powyższe własności pozwalają na wyciągnięcie następujących wniosków:

1. W procesie szeregowania wystarczy znajomość sum a_k, $k = 1,\ldots,n$.

2. Dołączenie zadań w przypadku dynamicznym nie stwarza konieczność budowy całej macierzy A od początku, lecz sprowadza się do skonstruowania jednego wiersza i kolumny.

3. W systemach sterowania, dla których zbiór zadań jest ściśle określony, macierz A wystarczy wyznaczyć tylko raz podczas generacji systemu.

Potrzebę realizacji danego zadania można określić przez wprowadzanie wektora flagactive $_k$ $k = 1,\ldots,n$ oraz odpowiednią modyfikację wektora s_k, $k = 1,\ldots,n$. Będziemy przyjmować w dalszym ciągu, że:

$$\text{active}_k = \begin{cases} 1 & \text{gdy zadanie } P_k \text{ jest aktywne,} \\ 0 & \text{w przeciwnym wypadku.} \end{cases}$$

Operacja aktywziacji zadania P_i jest określona przez następujący algorytm:

```
Algorithm      ACTIVE (i);
  begin
        active_i ⟵ 1;
        for k=1 to n do
                        if a_ik=1 then s_k ⟵ s_k+1
  end
```

Natomiast operacja dezaktywizacji tego zadania przebiega w sposób następujący:

```
Algorithm      PURGE (i);
  begin
        active_i ⟵ 0;
        for k=1 to n do
                        if a_ki=1 then s_k ⟵ s_k-1
  end
```

Dla realizacji powyższych algorytmów oraz algorytmu szeregowania konieczne jest konstrukcja macierzy A oraz odpowiednie ustawienie wektorów s_k, active $_k$ $k = 1,\ldots,n$. Operacje te są realizowane przez algorytm:

```
Algorithm    CONSTRUCTION;
  begin
        begin
          for   i=1   to   n  do
                for j=1 to n do
                if d_j ≤ max{d_i, p_i}  adn  α_j > α_i
                and p_j ≤ p_i
                then a_ij = 1   else   a_ij = 0

        end;
```

```
        begin
            for k=1 to n do
                active_k ← 1;
                for i=1 to n do
                if a_ij=1 then s_i ← s_i+1
        end;
end.
```

Wykorzystując wcześniej wprowadzone procedury oraz zakładając,że odpowiednie macierze zostały ustawione, problemy wyznaczenia do wykonania kolejnego zadania P_{ftask} przez proponowany w tym artykule algorytm szeregujący, można zapisać jak następuje:

```
Algorithm   SCHEDULER;
   begin
        maxnum ←   -1
        for i=1 to n do
            if active_i = 1  and  s_i=0  and
            maxnum < heuristic (d_i,σ_i,p_i) then
            begin
              ftask ← i;
              maxnum ← heuristic (d_i,σ_i,p_i)
            end;
end.
```

Funkcja heuristic ma postać:

```
function heuristic (d_i,σ_i,p_i)
  begin
      heuristic ← σ_i max{0,t-d_i}
  end
```

Analizując zaprezentowane algorytmy można zauważyć, że algorytmy, które dotyczą szeregowania zadań w komputerowym systemie sterowania,dla którego zbiór tych zadań jest stały, posiadają złożoność O(n). Najbardziej skomplikowanym algorytmem jest bowiem algorytm konstrukcji macierzy A.Posiada on złożoność $O(n^2)$. Mając na uwadze fakt, że konstrukcja macierzy A jest realizowana tylko na etapie generacji systemu komputerowego dla ustalonego zbioru zadań lub też w trakcie jego modyfikacji,co jest realizowane w trybie off-line, złożoność ta nie stanowi istotnego ograniczenia dla realizacji szeregowania zadań w czasie rzeczywistym.

2.2. Badania symulacyjne

Badania symulacyjne zostały zorientowane na porównanie zaproponowanego algorytmu heurystycznego ze znanym i stosowanym w komputerach sterujących algorytmem z priorytetami dynamicznymi. Porównania dokonano w sensie jakości szeregowania określonej wskaźnikiem jakości (2).

Przyjęto, że parametry α, d, p, są losowane, zgodnie z rozkładami danymi w tablicy 1. Przypadek 1 odpowiada równomiernemu rozłożeniu zadań w horyzoncie sterowania, przy czym zadania mają czasy wykonania z pewnego określonego przedziału czasowego, co jest zgodnie z charakterem zadań dla komputerów sterujących. Przypadek 2 uwzględnia spiętrzenia wykonywania zadań i istotnie charakteryzuje dobroć algorytmu ze względu na wymagania stawiane komputerom sterującym. Przypadki 3 i 4 odpowiadają odpowiednio przypadkom 1 i 2, z tym że zadania mogą posiadać dowolny czas wykonania z przewagą zadań krótszych.

Tablica 1

Parametr / Przypadek	Rozkład		
	α	d	p
1	równomierny w przedziale (A,B)	równomierny w przedziale (0,T)	równomierny w przedziale (K,L)
2	równomierny w przedziale (A,B)	gaussowski z parametrami E,E2	równomierny w przedziale (K,L)
3	równomierny w przedziale (A,B)	równomierny w przedziale (0,T)	wykładniczy o wartości średniej S
4	równomierny w przedziale (A,B)	gaussowski z parametrami E,E2	wykładniczy o wartości średniej S

Porównanie algorytmów prowadzono dla wskaźników jakości Q_H (heurystycznego) i Q_{PD} (z priorytetami dynamicznymi) wyznaczanymi w oparciu o wskaźnik jakości (2). W celu porównania obydwu rozważanych algorytmów wprowadzono zmienną losową $F = (Q_{PD}-Q_H)/Q_{PD}$, dla której obliczano dwa pierwsze momenty E(F) oraz V(F) odpowiednio z zależności:

$$E(F) = \frac{1}{R}\sum_{i=1}^{R} F_i, \qquad V(F) = \frac{1}{R-1}\sum_{i=1}^{R} (F_i - E(F))^2,$$

gdzie R jest liczbą realizacji.

Dla obliczeń przyjęto następujące wartości parametrów n = 20, R = 50, A = 0, B = 100, T = 100, K = 1, L = 15, E = 50, E2 = 2, S = 8.

Uzyskane wyniki przedstawiono w tablicy 2.

Tablica 2

Przypadek	Wartość średnia E(F)	Wariancja V(F)	Wskaźnik określający liczbę realizacji dla których E(F) jest dodatni w %
1	0,463	0,043	98
2	0,348	0,043	94
3	0,373	0,052	98
4	0,263	0,036	84

Analiza wyników wskazuje, że proponowany algorytm heurystyczny jest znacznie lepszy od stosowanego dotąd algorytmu z priorytetami dynamicznymi. Analiza ostatniej kolumny tablicy 2 wskazuje, że dla przeważającej liczby realizacji, czego należało oczekiwać z wartości liczby średniej E(F), algorytm heurystyczny daje lepsze wyniki. Jest to ważne, ponieważ ocena ta jest bardzo istotna z punktu organizacji pracy komputerów sterujących.

3. PODSUMOWANIE

Zaproponowano algorytm heurystyczny, który posiada złożoność obliczeniową O(n), tzn. taką, jaką posiadają stosowane obecnie w komputerach sterujących algorytmy: cykliczny oraz z priorytetami dynamicznymi. Badania symulacyjne pozwoliły stwierdzić, że algorytm heurystyczny jest znacznie lepszy w sensie jakości określonej sumą ważoną kar, aniżeli algorytm z priorytetami dynamicznymi. Wskazuje to na możliwość praktycznego wykorzystania tego algorytmu w komputerach sterujących.

W tym kierunku prowadzone są pewne prace implementacyjne.

LITERATURA

[1] COFFMAN E.G.Jr.: Computer and Job-shop Scheduling Theory (John Wiley and Sons, Inc., New York, 1976).

[2] GOŚCIŃSKI A., ZIELIŃSKI K.: A Technique and a System Structure for Online Activation Moments Determination for Steady-State Optimization, International Journal of System Science, Vol. 12, (1981), 263-176.

[3] GOŚCIŃSKI A., ZIELIŃSKI K.: The Determination of Direct Control Algorithms Activation Moments Using a Real-Time Scheduling Algorithm, Kybernetyka (in press).

[4] HELD M., KARP R.M.: A Dynamic Programming Approach to Sequencing Problems, Journal of the Society fo Industrial and Applied Mathematics, Vol. 10 (1962).

[5] LAWLER E.: The Scheduling of Single Machine Systems. A Review,The International Journal of Production Research, Vol. 3 (1969), 177-199.

[6] RINNOY A.H.G., LAGENEG B.J., LENSTRA J.J.: Minimizing Total Costs in One-Machine Scheduling, Opns. Res. 23 (1975), 908-928.

[7] SCHWIMER J.: On the N-job One-Machine, Sequence-Independent Scheduling Problem with Tardiness Penalties; A Branch-Bound Solution, Management Science, Vol. 18 (1972), B-301-B-312.

ЕВРИСТИЧЕСКИЙ АЛГОРИТМ РАСПИСАНИЯ ЗАДАЧ В ОДНОПРОЦЕССОРНОЙ КОМПЮТЕРНОЙ СИСТЕМЕ

Р е з ю м е

В статье представлен евристический алгоритм расписания задач охарактеризованных функцией штрафа из-за задержки в вычислительных системах управления процессами. Этот субоптимальный алгоритм обладает вычислительной сложностью O(n)?

HEURISTIC SCHEDULING ALGORITHM FOR SINGLE PROCESSOR CONTROL COMPUTER

S u m m a r y

Heuristic scheduling algorithm which minimizes weighted tardiness of jobs processing has been presented. An analysis of the proposed algorithm shows that it has computational complexity O(n).

The efficiency of this algorithm has been compared with the well-know algorithm with dynamic priorities.