ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994
Seria: AUTOMATYKA z.114 Nr kol. 1250

Adam JANIAK, Wiesław GRYGO, Krzysztof CHUDZIK
Politechnika Wrocławska.

SZEREGOWANIE ZADAŃ NA JEDNEJ MASZYNIE Z UWZGLĘDNIENIEM KARY ZA NIETERMINOWOŚĆ I ILOŚĆ ZUŻYTYCH ZASOBÓW

Streszczenie: Rozważamy problemy szeregowania zadań na jednej maszynie z czasami realizacji zadań zależnymi od przydzielonych zasobów. Każde zadanie ma określony termin dostępności i powinno być wykonane na określony termin, z karą za nieterminowość. Celem jest znalezienie harmonogramowania, które minimalizuje łączne koszty za nieterminowość i ilość zużytych zasobów. Do rozwiązania problemu proponujemy algorytm heurystyczny o złożoności $O(n^4)$.

SINGLE MACHINE SCHEDULING WITH PENALTIES FOR EARLINESS AND TARDINESS AND AMOUNT OF RESOURCES

Summary: We consider problems of single machine scheduling of tasks with processing times dependent on allotted resources. For each task accesing time is defined. The task should be completed in due date with penalty for earliness and tardiness. The objective is to find a schedule (permutation and resource allocation) which minimizes the cost of resources and penalties. We propose a heuristic algorithm with complexity $O(n^4)$.

EINORDNUNG VON AUFGABEN AUF EINER MASCHINE MIT BERÜCKSICHTIGUNG DER STRAFE FÜR DIE UNFÄLLIGKEIT UND DIE MENGE DER VERBRAUCHTEN VORRÄTE

Zusammenfassung: Die Probleme der Einordnung von Aufgaben auf einer Maschine mit den von zugeteilten Vorräten abhängigen Zeiten der Aufgabenrealisierung werden betrachtet. Jede Aufgabe hat die bestimmte Zugänglichkeitsfrist und soll für die bestimmte Frist mit der Strafe für die Unfälligkeit realisiert werden. Das Ziel ist, die Einordnung von Aufgaben zu finden, die Gesamtkosten für die Unfälligkeit und die Menge der verbrauchten Vorräte auf das Minimum bringt. Für die Lösung des Problems schlagen wir den heuristischen Algorithmus mit der Kompliziertheit: $O(n^4)$ vor.

1.Wstęp

Praca niniejsza dotyczy problemów szeregowania zadań na jednej maszynie. Czas wykonywania zadania zależy od ilości przydzielanych zasobów np: energii, paliwa, określonych środków materiałowych itp. Przydział różnego rodzaju zasobów często występuje w określonych proporcjach i dlatego wygodniej jest interpretować ilości przydzielonych zasobów jako nakład finansowy na wykonanie określonego zadania. Realizacja zadania jest możliwa od pewnego terminu, przy czym dla różnych zadań jest on inny. Zadanie powinno być wykonane na określony termin. Jeżeli termin wykonania jest późniejszy niż podany przez

zleceniodawcę, płacimy karę umowną, tym większą, im większe jest spóźnienie. Jeżeli ukończymy wykonywanie zadania wcześniej, ponosimy koszty, np. magazynowania.

Zadanie nasze polega na takim wyznaczeniu terminów rozpoczęcia i zakończenia wykonywania zbioru zadań, aby minimalizować globalne koszty ponoszone ze względu na:

- nakłady finansowe (zasobowe, zależne od czasu wykonywnia zadań),
- kary za nieterminowość (przedwczesne lub spóźnione wykonanie zadań).

Problem należy do klasy problemów silnie NP-trudnych. Do jego rozwiązania proponujemy algorytm heurystyczny. Dokonujemy rozbicia go na dwa zagadnienia.

Pierwsze z nich polega na znalezieniu terminów rozpoczęcia i zakończenia wykonywania zadań, a co się z tym wiąże także ilości przydzielonych zasobów, przy zadanej z góry kolejności (permutacji, uszeregowaniu) zadań. Podany przez nas krok algorytmu znajduje dla tego zagadnienia rozwiązanie optymalne.

W oparciu o wyniki uzyskiwane przez ten krok, algorytm heurystyczny proponuje permutację wynikową, co jest rozwiązaniem drugiego zagadnienia.

W konsekwencji uzyskujemy poszukiwany harmonogram. Algorytm jako całość posiada złożoność obliczeniową rzędu $O(n^4)$.

Do oszacowania jakości opracowanego przez nas algorytmu proponujemy jego porównanie z dosyć prostym dolnym ograniczeniem funkcji celu (kosztów globalnych). Dla przykładowego zbioru danych wybranych losowo z zachowaniem zdroworozsądkowych wartości, stosunek wyników heurezy do dolnego oszacowania funkcji celu średnio waha się w granicach 1.7 do 2.0 dla 50 do 200 zadań.

2. Opis problemu wejściowego

Danych jest n niezależnych zadań J_i, i=1,2,...,n, które mają być wykonywane na pojedynczej maszynie. Zakładamy, że:

1. zadania są niepodzielne,
2. maszyna w danej chwili czasu nie może wykonywać więcej niż jedno zadanie,
3. każde zadanie J_i może być wykonywane po upływie jego terminu dostępności r_i,
4. dla każdego zadania dany jest jego pożądany termin wykonania d_i,
5. czas wykonywania każdego zadania p_i jest liniowo zależny od zasobu u_i niezbędnego do realizacji tego zadania (jest to model typu czas-zasób) tzn.:

$$p_i = p_i(u_i) = b_i - a_i u_i$$

gdzie a_i, b_i są znanymi parametrami większymi od zera, przy czym zachodzi:

$$\alpha_i \le u_i \le \beta_i$$

gdzie α_i, β_i są znanymi parametrami ograniczającymi ilość przydzielonego zasobu:

$$\alpha_i \le 0, \qquad \alpha_i \le \beta_i \le b_i/a_i .$$

Funkcja kryterialna ma następującą postać:

$$A \Sigma u_i + B \Sigma w_i T_i + B' \Sigma w_i' E_i$$

gdzie: - A, B, B' są wagami poszczególnych składników.

Zadanie ma parametry:

- p_i - czas trwania zadania,
- S_i - termin rozpoczęcia wykonywania,
- C_i - termin zakończenia wykonywania,
- $T_i = \max (0, C_i - d_i)$ - czas spóźnienia zadania i,
- $E_i = \max (0, d_i - C_i)$ - czas przedwczesnego wykonania zadania i,
- w_i, w_i' - wagi.

Ogólnie problem ten można zapisać następująco:

$$1 \mid r_i, d_i, p_i = b_i - a_i u_i, \alpha_i \le u_i \le \beta_i \mid A \Sigma u_i + B \Sigma w_i T_i + B' \Sigma w_i' E_i .$$

Podczas opracowywania rozwiązania tego problemu doszliśmy jednak do wniosku, że w tym przypadku wygodniejszy jest nieco zmieniony model matematyczny zadań i samego kryterium, szczególnie ze względów obliczeń numerycznych. Zamieniliśmy go na model równoważny, a przejścia między obydwoma modelami są jednoznaczne. Opiszmy więc dokładnie problem i model, którym będziemy się posługiwać. Po jego opisaniu omówimy dokładnie związki transformacyjne jednego opisu problemu w drugi.

3. Opis problemu

3.1 Model matematyczny zadań

Liczba zadań - n.

Każde zadanie opisane jest następującymi parametrami:

J_i - zadanie i, i = 1,2...n - indeks zadania.

Następne parametry dotyczą pojedynczego zadania J_i:

r_i - termin dostępności, tzn. termin, od którego można rozpocząć wykonywanie zadania,

d_i - pożądany temin zakończenia wykonywania zadania, tzn termin wyznaczony przez zleceniodawcę, na który należy wykonać zadanie,

S_i - termin rozpoczęcia wykonania zadania (do wyznaczenia),

C_i - termin zakończenia wykonywania zadania (do wyznaczenia),

$p_i = C_i - S_i$ - czas wykonywania zadania,

$p_{i\,max}$ - czas wykonywania zadania dla minimalnej ilości zasobu (czas maksymalny),

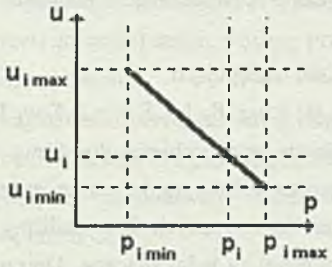
$p_{i\,min}$ - czas wykonywania zadania dla maksymalnej ilości zasobu (czas minimalny),

u_i - reprezentuje ilość przydzielonego zasobu , ponieważ dla $p_{i\ max}$ zużywamy $u_{i\ min}$ zasobu, a dla $p_{i\ min}$ zużywamy $u_{i\ max}$ zasobu to przy liniowej zależności czasu skrócenia zadania $p_{i\ max} - p_i$ od przydzielonego zasobu u_i:

$$u_i = u_{i\ min} + (p_{i\ max} - p_i)(u_{i\ max} - u_{i\ min})/(p_{i\ max} - p_{i\ min}),$$

gdzie $u_{i\ max}$ i $u_{i\ min}$ są z góry narzucone dla danego zadania np. przez wymogi technologiczne.

Wzór ten ilustruje poniższy rysunek:



Wartości $p_{i\ min}$ i $p_{i\ max}$ muszą być podane przez użytkownika.

b_i - koszt wykonania zadania gdy $p_i = p_{i\ max}$,

a_i - koszt skrócenia zadania o jednostkę czasu,

w_i - koszt spóźnienia zakończenia zadania o jednostkę czasu,

w_i' - koszt przedwczesnego zakończenia zadania o jednostkę czasu.

3.2.Sformułowanie zadania i opis kryterium

Zadanie sformułowane jest w następujący sposób:

Wyznaczyć takie harmonogramowanie zadań na jednej maszynie (ściślej - podać wszystkie pary S_i i C_i), aby minimalizować kryterium:

$$K = \Sigma[(p_{i\ max} - p_i)a_i + b_i] + \Sigma\, w_i T_i + \Sigma\, w_i' E_i$$

gdzie

- $T_i = \max(0, C_i - d_i)$ - czas spóźnienia zadania i,
- $E_i = \max(0, d_i - C_i)$ - czas przedwczesnego wykonania zadania i.

Uściślijmy co reprezentują sobą poszczególne składniki kryterium:

$\Sigma[(p_{i\ max} - p_i)a_i + b_i]$ - koszt wykonania zadań dla danych wartości p_i, (pamiętamy, że zadanie wykonywane krócej pochłania większą ilość zasobów),

$\Sigma\, w_i T_i$ - kary za spóźnienie terminów zakończenia wykonywania zadań,

$\Sigma\, w_i' E_i$ - kary za przedwczesne zakończenie wykonywania zadań.

Problem nasz możemy zapisać więc:

$$\mathbf{1 \mid r_i, d_i, p_{i\ min} \le p_i \le p_{i\ max} \mid \Sigma[(p_{i\ max} - p_i)a_i + b_i] + \Sigma\, w_i T_i + \Sigma\, w_i' E_i}$$

3.3 Transformacja problemu w postaci wejściowej do postaci, którą rozpatrujemy

Zadania mogą być sformułowane, w zależności od wygody, w jednej z wymienionych postaci. Algorytm posługuje się jednak drugim modelem zadań i kryterium, więc trzeba przeliczyć dane z postaci wejściowej do zasadniczej dla algorytmu:

Interpretacja następujących danych jest ta sama : r_i, d_i, p_i, S_i, C_i, u_i, T_i, E_i.

Aby zachować jednoznaczność opisu transformacji i nie zmieniać ogólnie przyjętych standardów oznaczeń, bedziemy indeksować dane, które nie są interpretowane tak samo indeksami:

- 0 - dla modelu wejściowego,
- 1 - dla modelu, którym posługuje się algorytm.

Wobec tego model zadania transformuje się następująco:

$p_{i\,max} = b_{i\,0} - \alpha_i u_i$, $\quad u_{i\,min} = \alpha_i$,

$p_{i\,min} = b_{i\,0} - \beta_i u_i$, $\quad u_{i\,max} = \beta_i$,

Zajmijmy się teraz funkcją kryterialną:

Aby nie powodować niepotrzebnego rozrostu wzorów:

$w_{i\,1} = B\, w_{i\,0}$,

$w_i'_{\,1} = B'\, w_i'_{\,0}$.

Dokładnie rozpiszmy część zasobową:

Wiemy, że: $b_{i\,1}$ - koszt wykonania zadania gdy $p_i = p_{i\,max}$, wobec tego $b_{i\,1} = A\alpha_i$.

$a_{i\,1}$ - koszt skrócenia zadania o jednostkę czasu, więc $a_{i\,1} = A / a_{i\,0}$ ponieważ: z modelu pierwotnego koszt za skrócenie o Δp_i zadania i-tego $A\Delta u_i = A\Delta p / a_{i\,0}$ (wyliczamy z przekształcenia $u_i = (b_{i\,0} - p_i)/a_{i\,0}$), a z modelu zasadniczego jest to $\Delta p_i\, a_{i\,1}$.

4. Algorytm optymalny wyznaczania terminów rozpoczęcia i zakończenia wykonywania zadań dla ustalonej kolejności zadań

Dane : Mamy zadaną permutację (uszeregowanie) zadań (dla ułatwienia opisu ustalmy, że kolejne zadania są ponumerowane kolejnymi liczbami naturalnymi od 1 do n).

Dla każdego zadania mamy podane r_i, d_i, b_i, a_i, w_i, w_i', $p_{i\,min}$, $p_{i\,max}$.

Wynik : S_i i C_i dla każdego zadania tak, aby funkcja kryterialna osiągała wartość minimalną.

Krok wstępny.

Ustawiamy zadania w kolejności od 1 do n.

Stawiamy $S_i = \max(C_{i-1}, r_i)$, $C_i = d_i$.

Jeżeli $C_i - S_i > p_{i\,max}$ to $S_i = d_i - p_{i\,max}$.

Ale jeżeli $w_i' = 0$ to $S_i = \max(C_{i-1}, r_i)$, $C_i = S_i + p_{i\,max}$.

Jeżeli $p_{i\,min} \le C_i - S_i \le p_{i\,max}$ to gdy

$w_i \ge a_i$ to S_i i C_i bez zmian; gdy $w_i < a_i$ to $C_i = S_i + p_{i\,max}$.

Jeżeli $C_i - S_i < p_{i\,min}$ to gdy

$w_i \ge a_i$ to $C_i = S_i + p_{i\,min}$ a gdy $w_i < a_i$ to $C_i = S_i + p_{i\,max}$.

Dla pełności przyjmujemy $C_0 = -\infty$.

Zwróćmy uwagę, że zadania są tak dostawiane, by wnosiły jak najmniejszy koszt. (W szczególności zadanie pierwsze jest ustawiane optymalnie jako zadanie odosobnione.) Ich terminy rozpoczęcia i zakończenia są na tyle przesunięte w prawo na osi czasu, że jakakolwiek modyfikacja, mająca na celu obniżenie wartości funkcji kryterialnej musi być przesunięciem tych terminów w lewą stronę. Ściślej, w rozwiązaniu optymalnym zachodzi $S_{i\,opt} \le S_i$ i $C_{i\,opt} \le C_i$ po kroku wstępnym.

Znalezienie rozwiązania optymalnego.

Dla każdego zadania od 1 do n obliczamy:

k_i^{sl} - koszt skrócenia i-tego zadania o jednostkę czasu w lewo (S_i bez zmiany, C_i w lewo):

a) ∞ dla $p_i = p_{i\,min}$

b) $a_i - w_i$ dla $p_i > p_{i\,min}$ i $C_i > d_i$

c) $a_i + w_i'$ dla $p_i > p_{i\,min}$ i $C_i \le d_i$

tk_i^{sl} - maksymalna modyfikacja (czas) C_i, dla której obowiązuje obliczony wyżej współczynnik:

a) ∞

b) $\min(C_i - d_i,\ p_i - p_{i\,min})$

c) $p_i - p_{i\,min}$

k_i^{ml} -koszt przesunięcia i-tego zadania o jednostkę czasu w lewo (S_i i C_i w lewo):

a) ∞ dla $S_i = r_i$

b) $-w_i$ dla $S_i > r_i$ i $S_i > C_{i-1}$ i $C_i > d_i$

c) w_i' dla $S_i > r_i$ i $S_i > C_{i-1}$ i $C_i \le d_i$

d) $\min(k_{i-1}^{ml},\ k_{i-1}^{sl}) - w_i$ dla $S_i > r_i$ i $S_i = C_{i-1}$ i $C_i > d_i$

e) $\min(k_{i-1}^{ml},\ k_{i-1}^{sl}) + w_i'$ dla $S_i > r_i$ i $S_i = C_{i-1}$ i $C_i \le d_i$

tk_i^{ml} - maksymalna modyfikacja C_i i S_i, dla której obowiązuje obliczony wyżej współczynnik:

a) ∞

b) $\min(S_i - r_i,\ C_i - d_i,\ S_i - C_{i-1})$

c) $\min(S_i - r_i,\ S_i - C_{i-1})$

d) $\min(S_i - r_i,\ C_i - d_i,\ tk_{i-1}^{sl})$ dla $k_{i-1}^{ml} \ge k_{i-1}^{sl}$

$\min(S_i - r_i,\ C_i - d_i,\ tk_{i-1}^{ml})$ dla $k_{i-1}^{ml} < k_{i-1}^{sl}$

e) $\min(S_i - r_i,\ tk_{i-1}^{sl})$ dla $k_{i-1}^{ml} \ge k_{i-1}^{sl}$

$\min(S_i - r_i,\ tk_{i-1}^{ml})$ dla $k_{i-1}^{ml} < k_{i-1}^{sl}$

k_i^{el} - koszt wydłużenia i-tego zadania o jednostkę czasu w lewo (S_i w lewo, C_i bez zmiany):

a) ∞ dla $p_i = p_{i\,max}$ lub $S_i = r_i$ lub $a_i = 0$

b) $-a_i$ dla $p_i < p_{i\,max}$ i $S_i > r_i$ i $S_i > C_{i-1}$

c) $\min(k_{i-1}^{ml}, k_{i-1}^{sl}) - a_i$ dla $p_i < p_{i\,max}$ i $S_i > r_i$ i $S_i = C_{i-1}$

tk_i^{el} - maksymalna modyfikacja S_i dla której obowiązuje obliczony wyżej współczynnik:

a) ∞

b) $\min(S_i - r_i, p_{i\,max} - p_i, S_i - C_{i-1})$

c) $\min(S_i - r_i, p_{i\,max} - p_i, tk_{i-1}^{sl})$ dla $k_{i-1}^{ml} \geq k_{i-1}^{sl}$

$\min(S_i - r_i, p_{i\,max} - p_i, tk_{i-1}^{ml})$ dla $k_{i-1}^{ml} < k_{i-1}^{sl}$

Mając policzone powyższe współczynniki, wybieramy do modyfikacji zadanie, które leży najbardziej na lewo i ma któryś ze współczynników ujemny. Gdy to zadanie ma więcej niż jeden ujemny współczynnik, wybieramy ten o najmniejszej wartości (uwzględniając znak). Następnie modyfikujemy zadanie tak jak wskazuje na to współczynnik k o odpowiedni czas tk. Gdy zachodzi konieczność (w przypadku k^{ml} i k^{el}) modyfikujemy o ten sam czas zadania poprzedzające w sposób w jaki były uwzględniane przy liczeniu kosztu modyfikacji bieżącego zadania.

Po dokonaniu modyfikacji liczymy na nowo współczynniki k i tk. Należy je policzyć od pierwszego zadania po lewej stronie, które było modyfikowane przy zmianie terminów wybranego przez nas zadania, aż do zadania, po którym występowała luka czasowa przed modyfikacją (zadania nie stykały się).

Następnie wybieramy zadanie do modyfikacji i modyfikujemy tak, jak opisano wyżej.

Postępowanie kontynuujemy aż do momentu, gdy wszystkie współczynniki k będą nieujemne.

Otrzymane rozwiązanie jest optymalnym wyznaczeniem S_i i C_i dla danego uszeregowania.

5. Heureza harmonogramowania zadań

Opis algorytmu.

Algorytm ten działa w oparciu o opisany wcześniej optymalny algorytm wyznaczania S_i i C_i zadań przy zadanym uszeregowaniu. Model zadań, jak i dane są te same.

Krok I - Tworzymy pomocniczą listę zadań.

- Dla każdego zadania J_i przypisujemy wagę W_i :

  jeżeli $r_i+p_{i\,min} \geq d_i$,to $W_i=w_i$, jeśli ten warunek nie zachodzi,to $W_i=\max(w_i,w_i')$.

- Ustawiamy zadania tak,aby $W_i \geq W_{i+1}$, dla i=1,2,...,n-1.

Krok II - Pobieramy kolejne zadania z pomocniczej listy i wstawiamy do nowej listy.

- Właściwą listę wynikową, obecnie pustą, inicjujemy przenosząc pierwsze zadanie z listy pomocniczej.

- Bierzemy kolejne zadanie z listy pomocniczej i ustawiamy je raz przed zadaniem na liście, i raz za nim. Dla obu przypadków uruchamiamy procedurę znajdowania optymalnych wartości S_i i C_i dla zadanej kolejności zadań. Jako rozwiązanie tego podproblemu przyjmujemy tę kolejność zadań, która dała mniejszy koszt.

- Dalej postępujemy analogicznie w sposób następujący:

Mamy ustaloną kolejność k zadań. Na liście pozostało ich n-k.

Bierzemy k+1 z listy pomocniczej i wstawiamy kolejno na początek listy właściwej, między każde dwa sąsiednie zadania, i na koniec. Dla każdego wstawienia uruchamiamy procedurę optymalnego wyznaczania S_i i C_i przy danej kolejności zadań. Wybieramy tę kolejność, która miała najmniejszy koszt po wyznaczaniu S_i i C_i. Mamy teraz kolejność k+1 zadań.

- Pobieramy następne zadanie z listy pomocniczej i powtarzamy procedurę aż do wyczerpania listy pomocniczej.

- Po wyczerpaniu listy pomocniczej, odtwarzamy wyznaczanie S_i i C_i dla najlepszej uzyskanej przez nas permutacji zadań (o najmniejszym koszcie).

6. Dowód opymalności algorytmu wyznaczania terminów rozpoczęcia i zakończenia wykonywania zadań dla ustalonej kolejności zadań

Dowód optymalności nie będzie zamieszczony w niniejszej pracy ze względu na ograniczenia na jej rozmiar i obszerność dowodu. Został on jednak przesłany organizatorom konferencji.

7. Złożoność obliczeniowa algorytmu wyznaczania terminów rozpoczęcia i zakończenia dla ustalonej kolejności zadań

Algorytm pracuje w dwóch krokach - wstępnym i zasadniczym. Krok wstępny ma złożoność rzędu O(n), ponieważ dla każdego zadania wykonywany jest algorytm o stałej złożoności obliczeniowej niezależnej od wartości n.

Krok zasadniczy znajdujący rozwiązanie optymalne dla danej kolejności zadań jest bardziej złożony i ma wyznaczyć terminy S_i i C_i dla n zadań.

Algorytm analizuje zadanie pierwsze, następnie sprawdza ujemność współczynnika modyfikacji i dołącza do niego zadanie drugie, potem trzecie, i tak dalej. Przyjmijmy za krok elementarny algorytmu realizację jednej modyfikacji na określonej grupie zadań.

Po kroku wstępnym zadanie k-te ma jakiś współczynnik modyfikacji ujemny i dokonujemy tej modyfikacji. Czas modyfikacji, czyli przedział czasowy, na którym ten współczynnik obowiązuje, wyznacza przekroczenie przez któreś z zadań od 1 do k terminu d, oparcie się o r, zetknięcie się z zadaniem porzedzającym, skrócenie go do p_{min} lub wydłużenie do p_{max}.

Zwróćmy uwagę na pewne własności:

1. Każdy parametr r_i i d_i dla i=1..k powoduje zmianę współczynnika modyfikacji zadania k i porzedzających go tylko jeden raz. Jeżeli spowodował on zmianę parametru modyfikacji przy uwzględnianiu zadania l (l<k), to w modyfikacji parametrów zadania k nie bierze on udziału. (np: dowolne zadanie przy modyfikacji tylko w lewo może tylko raz przekroczyć swój termin d i raz oprzeć się o r).

Ta własność jest dosyć oczywista. Następne już nieco mniej:

2. Zadanie, które raz zetknęło się z bezpośrednio go poprzedzającym pozostanie już z nim złączone tzn: $C_{l-1}=S_l$, ponieważ zadania przesuwają się wyłącznie w lewo i dla zadania l wykonujemy operację w lewo, gdy nie opłaci się wykonać tego dla poprzedzających go zadań.

3. Przeanalizujmy teraz, jak zadania będą zmieniać swoją długość. Załóżmy, że zadanie po kroku wstępnym ma minimalną długość. Załóżmy, że istnieje taka możliwość, że zostanie ono wydłużone i wydłużanie to skończy się na p_{max} (Gdy zadanie oprze się wcześniej o swój termin r, to koniec roszerzania w ogóle, a gdy o zadanie poprzednie, to może się ono dalej rozszerzać nawet do p_{max}, jeżeli koszt z odzyskanych zasobów jest większy od strat za modyfikację zadania poprzedniego). Dopiero pod wpływem następnych zadań może zostać ono skrócone (ściśnięte). Samo nie będzie się skracać, ponieważ następuje to tylko w przypadku, gdy leży ono poza swoim terminem d oraz ma czas wykonania większy od p_{min} i koszt za spóźnienie jest większy niż za włożone zasoby, a takiej sytuacji nie dopuszcza krok wstępny, który taki warunek sprawdza. Nie wydłuży się ono wiecej w trakcie dalszych modyfikacji pod wpływem nastepnych zadań, dlatego że wcześniej byłoby przesuwane a nie skracane. Przy skracaniu tracimy dodatkowy zasób, który powoduje spadek opłacalności modyfikacji, a który odzyskujemy dopiero przy wydłużaniu (pamiętajmy, że dla zadań poprzedzających wszystkie współczynniki są nieujemne, a przy wydłużaniu nic nie zyskujemy na kryterium na zadaniach następnych po rozpatrywanym, ponieważ nie są modyfikowane). Przypadek, gdy w sytuacji początkowej mamy $p>p_{min}$ lub wręcz maksymalną długość zadania zawiera się w wyżej wymienionym.

Tak więc w najgorszym przypadku zadanie może przejść do p_{max} a potem do p_{min}, co daje dwa punkty zmiany warunków wyliczania współczynników modyfikacji.

Zbierzmy więc razem wszystkie punkty zmiany wartości parametrów modyfikacji; są to

- r_k, d_k, C_{k-1}(jeden raz), p_{max}, p_{min},

Czyli jedno zadanie wnosi 5 punktów zmiany wartości parametrów. Ogółem jest 5n zmian.

Jeżeli po kroku wstępnym i po każdej modyfikacji przeliczymy wszystkie parametry, to będziemy je przeliczać 5n+1 razy.

Każdorazowe obliczenie współczynników ma złożoność n, jeżeli za 1 przyjmiemy wyliczenie wszytkich współczynników dla konkretnego zadania. Ogółem wykonamy $5n^2+n$ kroków obliczeniowych.

Algorytm wyznaczania terminów rozpoczęcia i zakończenia dla ustalonej kolejności zadań ma złożoność rzedu $O(5n^2)$ (po zaniedbaniu jako istotnie mniejszych kroku wstępnego i składnika n z sumy $5n^2+n$).

8. Złożoność obliczeniowa algorytmu jako całości

Jeżeli przyjmiemy za jeden krok algorytmu każdorazowe wywołanie procedury wyznaczania terminów rozpoczęcia i zakończenia dla ustalonej kolejności zadań o złożoności $O(k^2)$, (k-liczba zadań do ustawienia) to czas wykonania algorytmu będzie proporcionalny do:

$$1\cdot 1^2+2\cdot 2^2+\ldots+n\cdot n^2=\sum_{k=1}^{n}k\cdot k^2=\sum_{k=1}^{n}k^3=\frac{n^4+n^2}{4}.$$

Cały algorytm ma złożoność $O(n^4)$.

9. Dolne oszacowanie funkcji celu (LB)

Dolne oszacowanie polega na relaksacji problemu.
Sortujemy zadania wg. d_i. Następnie dzielimy je od początku na grupy po 7 zadań. Każdą siódemkę umieszczamy na osobnej maszynie, i dla każdej z nich dokonujemy przeglądu zupełnego wszystkich permutacji zadań. Do każdej permutacji (kolejności zadań) uruchamiamy algorytm wyznaczania terminów rozpoczęcia i zakończenia dla ustalonej kolejności zadań i wybieramy rozwiązanie optymalne. Liczba siedem wynika stąd, że komputer dokonuje przeglądu zupełnego jeszcze w rozsądnym czasie. Sumujemy rozwiązania dla wszystkich maszyn (kryterium jest sumacyjne) i w ten sposób otrzymujemy LB. LB ma złożoność typu O(7!·n), jest więc liniowy.
Przyjmijmy $x=(K_h-LB)/LB$. K_h - wynik heurezy.
Dla zdroworozsądkowych danych otrzymujemy wyniki:

liczba zadań	x_{min}	$x_{śr}$	x_{max}
50	0.12	0.78	1.59
100	0.39	0.92	1.82
200	0.70	0.92	1.33

Mimo zastosowania tak mało wyszukanego LB wyniki są dosyć dobre.
Algorytm wraz z oszacowaniem pracował na komputerze klasy PC 386/25MHz odpowiednio dla:
-50 zadań - 6.5 min, -100 zadań - ok 20 min, - 200 zadań - ok 80 min.
Podane czasy są wartościami średnimi. Algorytm wykazywał więc złożoność obliczeniową niższą niż wyznaczoną z szacowania złożoności obliczeniowej.

10. Wskazówki do implementacji

. Dla zadania są określone cztery koszty jednostkowe w_i, w'_i, a_i, b_i. Koszty k_i^{sl},k_i^{ml},k_i^{el} są wynikiem operacji addytywnych na kosztach.

Dla zadania jest określonych pięć czasów r_i, d_i, $p_{i\ max}$, $p_{i\ min}$. Czasy tk_i^{sl}, tk_i^{ml}, tk_i^{el}, C_i, S_i są wynikiem operacji addytywnych na czasach.

Jeśli na pewnym skończonym zbiorze wartości wymiernych wykonujemy wyłącznie operacje addytywne, to zbiór wszystkich możliwych wyników jest zbiorem wartości dyskretnych i każdą z wartości a_i z tego zbioru można wyrazić jako $a_i=k_i\cdot\delta$, gdzie k jest całkowite. Stąd wynika, że wartości a_i można reprezentować za pomocą k_i, pamiętając o δ. Oczywiście operacje addytywne na wartościach k_i są równoważne operacjom na a_i. Możemy więc reprezentować czasy i koszty jako wartości całkowitoliczbowe w czasie obliczeń i w razie potrzeby przeliczyć na wartości rzeczywiste. Zmiana reprezentacji pozwala na uniknięcie kłopotów z operacjami addytywnymi na liczbach zmiennopozycyjnych, dla których np: 5-3 jest różne od 1005-1003. Dodatkowo operacje całkowitoliczbowe są wykonywane wielokrotnie szybciej.

Dyskretyzujemy oddzielnie koszty (δk) i czasy (δt, t_0). Możemy to zrobić, bo są to różne wielkości i nie dodajemy ich razem ze sobą. Do dyskretyzacji wystarczy znaleźć δ i określić zakres reprezentowanych wartości i sposób ich reprezentacji. Przyjmujemy, że liczby całkowite są pamiętane na 32 bitach.

Dla kosztów wartości mieszczą się w przedziale $\pm\Sigma(a_i+\max(w_i, w'_i))$. Reprezentujemy je jako liczby całkowite ze znakiem z przedziału $\pm(2^{31}-2)$, a wartości z przedziału $\pm(2^{31}-1)$ reprezentuje $\pm\infty$.

Jeśli do czasów r_i i d_i dodamy jakąś stałą, to nie zmieni to problemu. Znajdujemy więc wartość $t_0=\min(\ldots, r_i, d_i, \ldots)$ i od wszystkich wartości r_i,d_i odejmujemy t_0. Otrzymane czasy będą miały wartości dodatnie. Dopiero dla takiego zbioru wartości r_i, d_i, $p_{i\ max}$, $p_{i\ min}$, b_i możemy wyznaczyć δt.

Reprezentujemy więc wartości z przedziału: $(0,\max(\max(r_i,d_i) - \min(r_i,d_i) + \Sigma(p_{i\ max})))$. Reprezentujemy wartości czasu jako liczby bez znaku z przedziału ($0, 2^{32}-2$), a wartość $2^{32}-1$ reprezentuje $\pm\infty$.

Przy reprezentacji $\pm\infty$ jako wartości całkowitych należy pamiętać, że poprawnie działają tylko operacje porównania, a przy odejmowaniu i dodawaniu należy pamiętać że wynik zależy od drugiego argumentu.

Żeby wyznaczyć δ wystarczy znaleźć największy wspólny podzielnik dla wartości, na których wykonujemy operacje addytywne. Można to zrobić za pomocą dzielenia z resztą na liczbach wymiernych. Przy liczeniu należy pamiętać o błędach wynikających z reprezentacji

liczb w komputerze. Można błędy wyeliminować przez zamianę wyniku każdej operacji na postać znakową z zaokrągleniem i przez ponowną zamianę na postać binarną co zapewni poprawne wyniki dla systemu dziesiętnego (10000.01, 0.6 to NWP=0.01). Jeśli tak otrzymana δ jest zbyt mała, aby można było przestawić wszystkie wartości z przedziału do dyskretyzacji to wyznaczamy δ na podstawie wzoru: δ=(długość przedziału do dyskretyzacji)/(długość przedziału po dyskretyzacji), co zapewnia nam maksymalną dokładność obliczeń.

Dla uzyskania wartości rzeczywistych należy koszty pomnożyć przez δk, dla czasów $p_{i\,max}$, p_i, $p_{i\,min}$, wystarczy te wartości przemnożyć przez δt, a dla r_i, d_i, C_i, S_i daną wartości należy pomnożyć przez δt i dodać t_0.

11. Uwagi

Można oczywiście w inny sposób wyliczać W_i np:

$$W_i = w_i + w_i',\ W_i = w_i + w_i' + a_i,\ W_i = w_i * w_i',\ \text{itp.}$$

Okazało się jednak, po symulacjach komputerowych, że kryterium $\max(w_i, w_i')$ daje najlepsze wyniki.

Odpowiada to następującemu rozumowaniu:

- gdy rozważamy ustawienie zadań w dużej odległości od ich położeń optymalnych, to na funkcję kosztów największy wpływ mają kary za nieterminowość.

- wybieramy wagi za nieterminowość jako max, ponieważ dla danego położenia zadania płacimy tylko jedną z kar za nieterminowość.

LITERATURA

[1] Błażewicz J.: Złożoność obliczeniowa problemów kombinatorycznych. WNT. Warszawa 1988.

[2] Błażewicz J.: Problemy optymalizacji kombinatorycznej - zlozoność obliczeniowa, algorytmy aproksymacyjne. WNT. Warszawa 1986.

[3] Błażewicz J., Cellary W., Słowiński R., Węglarz J.: Badania operacyjne dla informatyków. WNT. Warszawa 1983.

[4] Du J., Leung J.Y.-T.: Minimizing total tardiness on one machine is NP-hard. Math. Oper. Res. 1989.

[5] Janiak A.: Time - optimal control in a single machine problem with resource constraints. Automatica, vol. 22, No. 6, 1986.

[6] Janiak A.: Dokładne i przybliżone algorytmy szeregowania zadań i rozdziału zasobów w dyskretnych procesach przemysłowych. Prace Naukowe Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej. Wydawnictwo Politechniki Wrocławskiej. Wrocław 1991.

[7] Sawik T.: Optymalizacja dyskretna w elastycznych systemach produkcyjnych. WNT. Warszawa 1992.

[8] Smutnicki C.: Szeregowanie zadań z żądanym terminem zakończenia na jednej maszynie. Prace Naukowe Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej. Wrocław 1991.

Recenzent: Dr hab.inż. Ewa Dudek-Dyduch

Wpłynęło do Redakcji do 30.04.1994r.

Abstract

In this paper we consider problems of task scheduling on a single machine. The processing times of tasks depend on amount of resources (energy, fuel, etc.) allotted to them. The different kinds of resources in specified proportions are used so it is often possible to consider the amount of appropriated resources as the expenditure of money. The realization of each task is possible since specified moment and it should be completed in time (due date), with penalties for too early or too late completion.

The objective is to find a schedule (permutation and resource allocation) which minimizes a cost of:

- expenditure of money (dependent on resources - processing times),
- penalties for earliness and tardiness.

This problem is NP-hard in the strong sense. We propose heuristic algorithm. We split up the problem into two parts.

The first one is optimization of starting and completing times, and by that the allocation of resources for all tasks scheduled with given permutation.

Using the results, heuristic algorithm gives the permutation which is the solution of the second part.

In this way we obtain the solution of our problem. The complexity of the whole algorithm is $O(n^4)$.