ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2006
Seria: AUTOMATYKA z. 143 Nr kol. 1726

Dariusz OKOŁOWSKI
Uniwersytet im. A.Mickiewicza, Poznań

O PEWNYCH PROBLEMACH SZEREGOWANIA ZADAŃ Z EFEKTEM UCZENIA SIĘ

Streszczenie. Praca dotyczy wybranych problemów wielomaszynowego szeregowania zadań z efektem uczenia się oraz kryterium minimalizacji czasu zakończenia wykonywanych zadań. Przedstawiono przykłady, gdy nie jest możliwe zastosowanie dla tych problemów algorytmów szeregowania znanych dla problemów klasycznych. Podano modyfikacje wybranych klasycznych algorytmów szeregowania, pozwalające na ich zastosowanie dla problemów szeregowania jednostkowych zadań z efektem uczenia się.

ON SOME SCHEDULING PROBLEMS WITH LEARNING EFFECT

Summary. In the paper selected problems of minimum makespan, parallel machine scheduling with learning effect are considered. It has been shown by examples that some classical scheduling algorithms are not applicable in the case of scheduling with learning effect. There have been given modifications of some classical scheduling algorithms which allow to apply them to scheduling unit processing time jobs with learning effect.

1. Wstęp

Deterministyczna (klasyczna) teoria szeregowania zadań ([2]) powstała w połowie XX wieku jako narzędzie do rozwiązywania problemów spotykanych w produkcji przemysłowej. Podstawy tej teorii stanowią następujące założenia:
(Z1) w danej chwili zadanie może być wykonywane przez co najwyżej jedną maszynę oraz każda maszyna może wykonywać nie więcej niż jedno zadanie,
(Z2) prędkości maszyn mogą być różne, lecz podczas wykonywania zadań nie zmieniają się,
(Z3) czasy wykonywania zadań są stałymi, znanymi z góry wielkościami.

Zmieniając jakiekolwiek z założeń (Z1) - (Z3) otrzymujemy modele nieklasycznej teorii szeregowania zadań. Jednym z takich modeli jest szeregowanie zadań z efektem uczenia się.

Pierwszą pracą, dotyczącą efektu uczenia się w kontekście szeregowania zadań zależnych od pozycji w uszeregowaniu, jest praca Biskupa ([3]). Przedstawił on model efektu uczenia się, który zdominował dalsze badania nad tą klasą problemów. Również w tej pracy używać będziemy wprowadzonego przez niego modelu. Opis innych modeli szeregowania zadań z efektem uczenia się można znaleźć np. w [1], [7] lub [14].

Najszerzej badanymi zagadnieniami w dotychczasowych pracach dotyczących szeregowania zadań z efektem uczenia się są problemy jednomaszynowe. Rozważano kryteria takie, jak: czas zakończenia wykonywanych zadań ([10]), suma czasów zakończenia wykonywanych zadań ([3]), suma ważonych czasów wykonywanych zadań ([1]) oraz kryteria dotyczące opóźnień ([10], [13]). Rozważano również problemy, gdzie kryteriami są kombinacje wypukłe innych kryteriów ([3]). Mniejszą grupę stanowią prace, dotyczące środowisk wielomaszynowych. Prace te dotyczą głównie szeregowania zadań na identycznych maszynach równoległych dla kryterium sumy czasów zakończenia wykonywania zadań ([11]) oraz problemów szeregowania w systemach przepływowych ([9], [14]).

2. Sformułowanie problemu

W pracy rozważamy następujący problem szeregowania zadań. Danych jest m równoległych identycznych maszyn oraz zbiór n zadań takich, że czas wykonywania j-tego zadania ze zbioru zadań T na r-tej pozycji w uszeregowaniu wyraża się wzorem $p_{j,r} = p_j * r^a$, gdzie $a \leqslant 0$ jest indeksem uczenia się, a p_j jest początkowym czasem wykonania zadania. Na zbiorze zadań zdefiniowano ograniczenia kolejnościowe zadane w postaci acyklicznego digrafu $G_p = (T, E_p)$, gdzie E_p jest zbiorem łuków (T_i, T_j) takich, że w grafie G_p istnieje skierowana ścieżka od T_i do T_j wtedy i tylko wtedy, gdy wykonywanie T_i musi się zakończyć przed rozpoczęciem wykonywania T_j (symbolicznie $T_i \prec T_j$). Stosowanym kryterium optymalności uszeregowania jest czas zakończenia wykonywanych zadań

$$C_{max} = \max_{i \in \{1,\ldots,m\}} \{p^i_{[1]} * 1^a + p^i_{[2]} * 2^a + \ldots + p^i_{[n]} * n^a\},$$

gdzie $p^i_{[j]}$ oznacza początkowy czas wykonywania j-ego zadania na maszynie P_i.

Do opisu omawianych problemów będziemy używali notacji $\alpha|\beta|\gamma$ ([8]).

3. Omówienie uzyskanych wyników

Wobec NP-zupełności problemu $Pm||C_{max}$ (w wersji decyzyjnej), problem $P2|p_{j,r} = p_j * r^a|C_{max}$ w wersji decyzyjnej także jest NP-zupełny ([10]).

3.1. Dowolne czasy wykonywania zadań

W ciągu kilku ostatnich dekad dla problemu $Pm||C_{max}$ wprowadzono wiele heurystyk i algorytmów aproksymacyjnych, generujących rozwiązania przybliżone

o pewnych gwarancjach jakości ([2]).

Jedną z nich jest algorytm LPT (od ang. *Longest processing time first*, [2]). W czasie $t = 0$ przydziela ona m największych zadań do wolnych maszyn. Następnie, w każdym momencie czasu, w którym następuje zwolnienie maszyny, przydzielamy tej maszynie największe dostępne zadanie.

Znane jest twierdzenie ([2]), dotyczące analizy najgorszego przypadku dla LPT, mówiące, że $\frac{C_{max}(S_{LPT})}{C_{max}(S_{OPT})} \leqslant \frac{4}{3} - \frac{1}{3m}$, gdzie S_{LPT} to uszeregowanie uzyskane za pomocą LPT, a S_{OPT} to uszeregowanie optymalne. W przypadku problemu uwzględniającego efekt uczenia się, twierdzenie to nie zachodzi.

Przykład 1
Dana jest instancja problemu $P2|p_{j,r} = p_j * r^a|C_{max}$ taka, że indeks $a = -0.322$, zbiór zadań $T = \{T_1, T_2, \ldots, T_{10}\}$, a początkowe czasy wykonania są dane wektorem $\vec{p} = [10, 10, 1, 1, 1, 1, 1, 1, 1, 1]$. Wartość kryterium dla uszeregowania zgodnego z regułą LTP $S_{LPT} = ((T_1, T_3, T_5, T_7, T_9), (T_2, T_4, T_6, T_8, T_{10}))$ wynosi:

$$C_{max}(S_{LPT}) = 10 * 1^{-0.322} + 1 * 2^{-0.322} + 1 * 3^{-0.322} + 1 * 4^{-0.322} + 1 * 5^{-0.322}.$$

Wartość kryterium dla uszeregowania optymalnego (uzyskanego metodą pełnego przeglądu) $S_{OPT} = ((T_3, T_5, T_7, T_9, T_1), (T_4, T_6, T_8, T_{10}, T_2))$ wynosi:

$$C_{max}(S_{OPT}) = 1 * 1^{-0.322} + 1 * 2^{-0.322} + 1 * 3^{-0.322} + 1 * 4^{-0.322} + 10 * 5^{-0.322}.$$

Stąd mamy, że $\frac{C_{max}(S_{LPT})}{C_{max}(S_{OPT})} \approx \frac{12.73}{9.1} = 1.4 > \frac{7}{6}$.

Podobny przykład można wygenerować również dla dowolnego $m > 2$. Wystarczy w zbiorze T umieścić m „dużych" zadań i $4m$ zadań „małych".

Innym algorytmem aproksymacyjnym dla problemu $Pm||C_{max}$ jest algorytm listowy (ang. *list scheduling*, w skr. LS). Przydziela on wolnym maszynom pierwsze dostępne zadanie z listy będącej dowolną permutacją zadań.

Znane jest twierdzenie ([2]), dotyczące analizy najgorszego przypadku dla LS, zgodnie z którym $\frac{C_{max}(S_{LS})}{C_{max}(S_{OPT})} \leqslant 2 - \frac{1}{m}$, gdzie S_{LS} oznacza uszeregowanie będące wynikiem działania algorytmu LS.

Przykład 2
Dana jest instancja problemu $P2|p_{j,r} = p_j * r^a|C_{max}$ taka, że $a = -0.55$, zbiór $T = \{T_1, \ldots, T_{10}\}$ taki, że $\vec{p} = [30, 30, 1, 1, 1, 1, 1, 1, 1, 1]$.

Wartość kryterium dla uszeregowania otrzymanego z algorytmem LS (w przypadku, gdy zadania na liście dane są w porządku nierosnącym względem parametru p_j) $S_{LS} = ((T_1, T_3, T_5, T_7, T_9), (T_2, T_4, T_6, T_8, T_{10}))$ wynosi:

$$C_{max}(S_{LS}) = 30 * 1^{-0.55} + 1 * 2^{-0.55} + 1 * 3^{-0.55} + 1 * 4^{-0.55} + 1 * 5^{-0.55}.$$

Wartość kryterium dla uszeregowania optymalnego (uzyskanego metodą pełnego przeglądu) $S_{OPT} = ((T_3, T_5, T_7, T_9, T_1), (T_4, T_6, T_8, T_{10}, T_2))$ wynosi:

$$C_{max}(S_{OPT}) = 1 * 1^{-0.55} + 1 * 2^{-0.55} + 1 * 3^{-0.55} + 1 * 4^{-0.55} + 30 * 5^{-0.55}.$$

Stąd mamy, że $\frac{C_{max}(S_{LS})}{C_{max}(S_{OPT})} \approx \frac{32.1}{15.07} = 2.13 > \frac{3}{2}$.

Przykład 2 pokazuje, że twierdzenie to nie jest prawdziwe w przypadku modelu
z efektem uczenia się dla $m = 2$ (dla przypadku , gdy $m > 2$ można wskazać podobny przykład, zob. komentarz po przykładzie 1).

3.2. Jednostkowe czasy wykonywania zadań

Relaksacja $Pm||C_{max}$ prowadzi do kolejnego problemu, $Pm|p_j = p|C_{max}$, który jest wielomianowo rozwiązywalny ([2]). Problem $Pm|p_{j,r} = p * r^a|C_{max}$ jest również rozwiązywalny w czasie wielomianowym.

Przedstawimy algorytm A1, rozwiązujący problem $Pm|p_j = p * r^a|C_{max}$. W pierwszym kroku algorytm A1 przydziela $\left\lfloor \frac{n}{m} \right\rfloor$ zadań każdej z m maszyn. Jeśli $d_T = n - m * \left\lfloor \frac{n}{m} \right\rfloor > 0$, to należy przydzielić d_T maszynom po jednym dodatkowym zadaniu. Długość uszeregowania dla n-zadaniowej instancji problemu wynosi $C_{max} = \sum_{r=1}^{n_m} p * r^a$, gdzie $n_m = \left\lceil \frac{n}{m} \right\rceil$.

Twierdzenie 1. *Algorytm A1 generuje optymalne uszeregowanie dla problemu* $Pm|p_j = p * r^a|C_{max}$ *w czasie* $O(n)$.

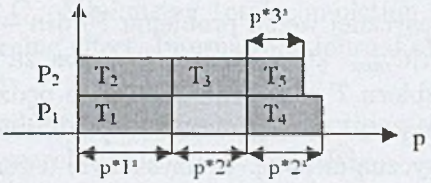
Dowód. (Szkic) Załóżmy, że uszeregowanie generowane przez algorytm A1 nie jest optymalne, czyli istnieje uszeregowanie krótsze w sensie kryterium C_{max}. Oznacza to, że w uszeregowaniu optymalnym (niezgodnie z A1) musimy co najmniej jedno z zadań przydzielonych i-tej maszynie ($1 \leqslant i \leqslant m$) przez algorytm A1 przydzielić innej maszynie. Wtedy należy rozpatrzeć dwa przypadki. Jeśli przydzielimy je maszynie, która ma przydzielonych $\left\lfloor \frac{n}{m} \right\rfloor$ zadań (gdy $\left\lfloor \frac{n}{m} \right\rfloor \neq \left\lceil \frac{n}{m} \right\rceil$), to długość uszeregowania nie zmieni się. Jeśli przydzielimy je maszynie, która ma przydzielonych $\left\lceil \frac{n}{m} \right\rceil$ zadań, to $C_{max} = \sum_{r=1}^{n_m+1} p * r^a$, co jest wartością większą od uzyskanej przez algorytm A1. Sprzeczność.

□

Innym klasycznym problemem, dla którego istnieje dokładny algorytm wielomianowy A2, jest problem $P2|p_j = p, prec|C_{max}$ ([2]). W pierwszej fazie algorytm A2 przydziela zadaniom etykiety liczbowe. Następnie, dostępne zadania o najniższej etykiecie są przydzielane do wolnych maszyn w każdej jednostce czasu. Pokażemy teraz, że algorytm ten nie znajduje zastosowania dla modeli uwzględniających efekt uczenia się.

Przykład 3
Dana jest następująca instancja problemu $P2|p_j = p * r^a, prec|C_{max}$: $a < 0$, zbiór zadań $T = \{T_1, T_2, T_3, T_4, T_5\}$ oraz zbiór ograniczeń kolejnościowych jest dany przez częściowy porządek $O_T = \{T_1 \prec T_3, T_2 \prec T_3, T_3 \prec T_4, T_3 \prec T_5\}$.

Jeżeli uszeregujemy zbiór T w porządku przedstawionym na rys. 1 (zgodnie z algorytmem A2), to wynikiem jest sytuacja, w której „przedziały czasu" począwszy od pozycji



Rys. 1. Przykład sytuacji nierozpatrywanej przez algorytm A2

3 mają na każdej maszynie inną szerokość.

Kolejnym problemem, dla którego istnieje wielomianowy algorytm A3, jest problem $Pm|p_j = p, in - tree|C_{max}$ ([2]).

Algorytm A3 dla problemu $Pm|p_j = p, in - tree|C_{max}$ działa w dwóch krokach. Dla każdego zadania ze zbioru T należy obliczyć odległość danego zadania od korzenia drzewa. Następnie, w każdej jednostce czasu wolnym maszynom należy przydzielać zadania w kolejności od najwyższego poziomu. Algorytm ten można zaimplementować w czasie $O(n)$ ([2]).

W przypadku problemu $Pm|p_{j,r} = p * r^a, in - tree|C_{max}$ algorytm A3 wymaga modyfikacji. Zmodyfikowany algorytm A3m, w momencie, gdy w aktualnym przedziale czasu dostępnych zadań jest mniej niż dostępnych maszyn, przydziela zadania do maszyn o największej liczbie wykonanych zadań.

Twierdzenie 2. *Algorytm A3m generuje optymalne uszeregowanie dla problemu* $Pm|p_{j,r} = p * r^a, in - tree|C_{max}$ *w czasie* $O(n)$.

Dowód. (Szkic) Optymalność algorytmu A3m wynika bezpośrednio z optymalności algorytmu A3. Wystarczy pojęcie „jednostki czasu" zastąpić pojęciem „przedziału czasu" oraz zapewnić wykonywanie się zadań na maszynach o jak największej liczbie przetworzonych zadań. W przypadku, gdy chcielibyśmy przydzielić zadanie maszynie, która wykonała mniejszą liczbę zadań, to czas wykonania zadania wzrasta, ponieważ czynnik r^a nie jest minimalny.

Co z sytuacją, gdy dostępnych zadań w przedziale $r+1$ jest więcej niż maszyn, które do r-tego przedziału wykonały r zadań (por. uszeregowanie na rys. 1)? Taka sytuacja nie może zajść, gdyż ograniczenia kolejnościowe są dane w postaci drzewa wstępującego, a przez to dostępnych zadań w i-tym przedziale nigdy nie będzie mniej niż w przedziale $i + 1$.

□

W klasycznej wersji problem $Pm|p_j = p, in - tree|C_{max}$ jest istotnie związany z problemem $Pm|p_j = p, out - tree|C_{max}$. Do rozwiązania wykorzystuje się ten sam algorytm A3 (odwracając łuki w wejściowym digrafie i czytając wyjściowe uszeregowanie "wspak"). Niestety, w szeregowaniu zadań z efektem uczenia się algorytm ten zawodzi, ponieważ czytając uszeregowanie "wspak", należałoby wziąć pod uwagę skracanie się zadań.

Rozszerzeniem klasycznej wersji problemu $Pm|in - tree|C_{max}$ jest problem $Pm|p_j = p, in - forest|C_{max}$, który można rozwiązać za pomocą tego samego algorytmu, dodając do zbioru T jedno zadanie, które będzie następnikiem korzeni wszystkich drzew należących do tego lasu. W problemie z efektem uczenia się można stosować indentyczną metodę. W uzyskanym uszeregowaniu należy odrzucić ostatni przedział zadań, ponieważ będzie się w nim znajdowało tylko dodatkowe sztuczne zadanie.

4. Podsumowanie

W pracy podano wstępną analizę wybranych problemów związanych z szeregowaniem zadań z efektem uczenia się.

Kolejnym krokiem, dotyczącym analizy zagadnień wielomaszynowych, są problemy uwzględniające inne modele efektu uczenia się wprowadzone w [1], [4], [7] lub [12]. Przedmiotem dalszych badań mogą być problemy wielomaszynowe dla innych kryteriów optymalności, np. kryteriów związanych z pożądanym czasem zakończenia wykonywania zadań ([10], [13]). Można również zbadać własności uszeregowań, dla których kryterium jest kombinacją wypukłą kryteriów prostych (zob. [3], [10]).

LITERATURA

1. Bachman A., Janiak A.: Scheduling jobs with position dependent processing times. Journal of the Operational Research Society 55, 2004, p. 257–264.
2. Błażewicz J., Ecker K.H., Schmidt G., Węglarz J.: Scheduling in Computer and Manufacturing Systems. Springer-Verlag, Berlin-Heidelberg 1996.
3. Biskup D.: Single-machine scheduling with learning considerations. European Journal of Operational Research 115, 1999, p. 173–178.
4. Cheng T.C.E., Wang G.: Single Machine Scheduling with Learning Effect Considerations. Annals of Operation Research 98, 2000, p. 273–290.
5. Janiak A., Śnieżyk A.: Wielomaszynowe problemy szeregowania zadań z efektem uczenia się. Badania operacyjne i systemowe 1, 2004, s. 167–186.
6. Janiak A., Rudek R.: On a general model of the learning effect. Proceeddings of the 11th IEEE International Conference on Methods and Models in Automation and Robotics MMAR 2005, p. 1115–1119.
7. Kou W.H., Yang D.L.: Minimizing the makespan in single machine scheduling problem with a time-based learning effect. Information Processing Letters 97, 2006, p. 64–67.
8. Leung J.Y.T.: Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press 2004.

9. Lee W.C., Wu C.C.: Minimizing total completion time in a two-machine flowshop with learning effect. Internationl Journal of Production Economics 88, 2004, p. 85–93.
10. Mosheiov G.: Scheduling problems with a learning effect. European Journal of Operational Research 132, 2001, p. 687–693.
11. Mosheiov G.: Parallel machine scheduling with a learning effect. Journal of the Operational Research Society 52, 2001, p. 1165—1169
12. Mosheiov G., Sidney J.B.: Scheduling with general job-dependent learning curves. European Journal of Operational Research 147, 2003, p. 665–670.
13. Mosheiov G., Sidney J.B.: Note on scheduling with general learning curves to minimize the number of tardy jobs. Journal of the Operational Research Society 56, 2005, p. 110–112.
14. Wang J.B., Xia Z.Q.: Flow-shop scheduling with learning effect. Journal of the Operational Society 56, 2005, p. 1325–1330.

Recenzent: Prof. dr hab. inż. Adam Janiak

Abstract

In the paper selected scheduling problems with sequence-dependent learning effect are considered. In some of these problems precedence constraints may exist. The chosen job model is $p_{j,r} = p_j * r^a$, where $p_{j,r}$ is the processing time of job sequenced on r-th place in some schedule, p_j is the initiate processing time, and $a \leqslant 0$ is a learning index. All the problems concern identical parallel machines with the makespan criterium.

It has been shown by counter-examples that some classical approximation algorithms are not useful when learning considerations have been taken into account. There have been also proposed some modifications of algorithms for scheduling unit processing times jobs with in-tree precedence constraints for the case of learning effect. It has been proved that some of these modified algorithms are optimal.