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

Krzysztof MAIK, Eugeniusz TOCZYŁOWSKI
Instytut Automatyki Politechniki Warszawskiej

ALGORYTM HARMONOGRAMOWANIA PRODUKCJI PORCJAMI Z POJEDYNCZYM OGRANICZENIEM ZASOBOWYM

Streszczenie: W referacie opracowano efektywną metodę harmonogramowania produkcji porcjami w problemie wieloetapowym, w którym rozważa się występowanie pojedynczego ograniczenia zasobowego w pierwszym etapie. Metoda wykorzystuje właściwości algorytmu Wagelmansa-Hoesela-Kolena w celu T-krotnej redukcji złożoności obliczeniowej algorytmu, gdzie T jest liczbą etapów.

ALGORITHM FOR LOT-SIZE SCHEDULING WITH A SINGLE RESOURCE

Summary: The paper presents an efficient method for the dynamic lot-size scheduling when a single resource is limited in the first period. The method uses particular properties of the Wagelmans-Hoesel-Kolen algorithm to reduce the complexity of algorithm T times, where T is the number of periods.

EIN ALGORITHMUS DER HARMONOGRAMMBILDUNG FÜR DIE PRODUKTION IN DEN PARTIEN BEIM EIZELNEN VORRÄTLICHEN BESCHRÄNKEN

Zusammenfassung: Im Referat wurde eine effektive Methode der Harmonogrammbildung für die Produktion im Partien in einem Problem mit vielen Etappen, in dem wird einzelne vorrätliche Beschränken im ersten Etappe erwogen, bearbeitet. Diese Methode benutzt eigenartige Eigenschäfte des Algorithmen WHK zwecks der Reduktion der Rechnenkompliziertheit T mal, wo T ist die Zahl der Etappen.

1. Sformułowanie problemu

W klasycznym problemie Wagnera-Whitina harmonogramowania produkcji porcjami [5] jest poszukiwany plan produkcji pojedynczego wyrobu na horyzoncie czasu złożonym z T etapów, przy znanym zapotrzebowaniu w każdym z etapów. Zakłada się, że niezerowa produkcja w danym etapie wiąże się z pewnymi niezerowymi kosztami wznowienia

produkcji, niezależnymi od jej wielkości. Zakłada się przy tym, że możliwa jest w danym etapie produkcja o wielkości przekraczającej bieżące zapotrzebowanie i magazynowanie nadwyżki, przy czym uwzględnia się koszt magazynowania jednostek wyrobu.

Problem ten można zapisać jak następuje:

$$\min \sum_{t=1}^{T} (s_t v_t + c_t x_t + h_t^+ I_t^+) \tag{1}$$

przy ograniczeniach:

$$\begin{aligned} I_{t-1}^+ + x_t - I_t^+ &= d_t \\ 0 \leqslant x_t &\leqslant M_t v_t \\ I_t^+ &\geqslant 0 \\ I_0 = I_T &= 0 \\ v_t &\in \{0,1\} \end{aligned}$$

gdzie zmiennymi decyzyjnymi zadania są: x_t – wielkość produkcji w okresie t, v_t – zmienna binarna $= 1$ jeżeli wznowiono produkcję w t, I_t^+ – stan zapasu wyrobu na koniec etapu t. Parametrami zadania są: s_t – koszt wznawiania produkcji, c_t – jednostkowy koszt produkcji w etapie t, h_t^+ – koszt magazynowania jednostki wyrobu w etapie t, d_t – zapotrzebowanie na wyrób w etapie t, M_t – dostatecznie duża stała.

W wielu zagadnieniach planowania produkcji rozważa się produkcję wielu rodzajów wyrobów (np. N) wykorzystujących wspólne zasoby konieczne do produkcji. W modelu rozważanym w niniejszym opracowaniu uwzględniamy istnienie jednego, zagregowanego zasobu dodatkowego. Zasób ten jest wymagany w ilości e_{kt} dla wznowienia produkcji wyrobu k-tego w etapie t, przy jednostkowym zużyciu p_{kt} (porcja zasobu wymagana do produkcji jednostki wyrobu). Dostępność zasobu w etapie t jest ograniczona przez Q_t.

Formalny zapis rozważanego zadania planowania produkcji przedstawia się następująco:

$$\min \sum_{k=1}^{N} \sum_{t=1}^{T} (s_{kt} v_{kt} + c_{kt} x_{kt} + h_{kt}^+ I_{kt}^+)$$

przy ograniczeniach:

$$\begin{aligned} I_{k,t-1}^+ + x_{kt} - I_{kt}^+ &= d_{kt} && \text{dla } t = 1,\ldots,T \text{ i } k = 1,\ldots,N \\ x_{kt} &\leqslant M v_{kt} \\ I_{kt}^+ &\geqslant 0 \\ \textstyle\sum_{k=1}^{N} (e_{kt} v_{kt} + p_{kt} x_{kt}) &\leqslant Q_t && \text{dla } t = 1,\ldots,T \\ v_{kt} &\in \{0,1\} \end{aligned}$$

gdzie oznaczenia są identyczne do opisanych poprzednio, rodzaje wyrobów są indeksowane przez k, a etapy przez t.

Warto zwrócić uwagę, że jesteśmy zainteresowani rozwiązywaniem powyższego zadania w układzie sterowania repetycyjnego, gdzie w jednym kroku algorytmu repetycyjnego podejmujemy decyzję o asortymencie i wielkości produkcji w najbliższym etapie, a następnie powtarzamy algorytm w kolejnych krokach dla dalszych etapów. W tym kontekście jest interesująca możliwość przeformułowania ograniczeń zasobowych w kolejnych etapach, tak by można je było sprowadzić, przynajmniej w pewnym stopniu, do ograniczenia etapu

pierwszego. W ten sposób możemy mieć do czynienia tylko z jednym, skalarnym ograniczeniem zasobowym. Dalej pokażemy, że rozwiązywanie tak otrzymanego zadania może być zrealizowane szczególnie efektywnie.

Rozważmy zatem, jak oszacować od dołu zużycie zasobu w etapie pierwszym,tak aby uniknąć trudności z dostępnością zasobu w etapach następnych. Chcemy zagwarantować warunki, dzięki którym dla dowolnych stanów zapasów na koniec etapu pierwszego będzie możliwa produkcja zaspokajająca zapotrzebowanie w etapach następnych. Z punktu widzenia wykorzystywania zasobów najbardziej krytyczny przypadek zachodzi dla zerowego stanu zapasów na koniec etapu pierwszego.

Zużycie zasobu w etapie t przy produkcji równej bieżącemu zapotrzebowaniu wyraża się wzorem:

$$C_{dt} = \sum_{i=1}^{N}(e_{it}v_{it} + p_{it}d_{it})$$

Sumaryczna ilość zasobu brakującego do zaspokojenia bieżącego zapotrzebowania w etapach $t = 2, ..., m$ jest równa:

$$\sum_{t=2}^{m}(C_{dt} - Q_t)$$

W ten sposób powstaje dolne ograniczenie na zużycie zasobu w etapie pierwszym:

$$\max_{2\leqslant m\leqslant T}\{\sum_{t=2}^{m}(C_{dt} - Q_t)\} \leqslant \sum_{i=1}^{N}(e_{i1}v_{i1} + p_{i1}x_{i1})$$

W niniejszym opracowaniu przedstawiono efektywną metodę harmonogramowania produkcji porcjami w sformułowanym powyżej problemie wieloetapowym, w którym rozważa się występowanie pojedynczego, zagregowanego ograniczenia zasobowego w pierwszym etapie. Metoda wykorzystuje relaksację Lagrange'a do dekompozycji problemu. Wielokrotne, efektywne rozwiązywanie podproblemów jest możliwe dzięki wykorzystaniu specyficznych właściwości problemu i cech algorytmu Wagelmansa-Hoesela-Kolena. Prowadzi to do T-krotnej redukcji złożoności obliczeniowej algorytmu, w stosunku do stosowanych dotychczas metod. W rozdziale 4 omówiono zaimplementowane dwie wersje algorytmu WHK rozwiązywania podproblemów zdekomponowanych i przebadano ich numeryczne właściwości.

2. Relaksacja Lagrange'a obustronnego ograniczenia zasobowego

Przyjmijmy $q = \max_{2\leqslant m\leqslant T}\{\sum_{t=2}^{m}(C_{dt} - Q_t)\}$ i załóżmy, że $0 \leqslant q < Q_1$. Obustronne ograniczenie zasobowe w etapie pierwszym można sprowadzić do równości wprowadzając dodatkową, ograniczoną zmienną dopełniającą s:

$$\sum_{i=1}^{N}(e_{i1}v_{i1} + p_{i1}x_{i1}) + s - Q_1 = 0 \quad \text{dla } 0 \leqslant s \leqslant Q_1 - q$$

Jeżeli dokonamy relaksacji Lagrange'a tego ograniczenia równościowego, mnożnik będzie nieograniczony co do znaku, natomiast dualna funkcja Lagrange'a względem pojedynczego mnożnika $\mu = \mu_1$ przybierze postać:

$$L_D(\mu) = \min_{x,v,s} \left[\sum_{i=1}^{N} \sum_{t=1}^{T} (s_{it} v_{it} + c_{it} x_{it} + h_{it} I_{it}^{+}) + \mu \Big(\sum_{i=1}^{N} (e_{i1} v_{i1} + p_{i1} x_{i1}) + s - Q_1 \Big) \right]$$

Można się teraz pozbyć pomocniczej zmiennej s dokonując wstępnie minimalizacji po s. Widać, że jeżeli $\mu \geqslant 0$, to minimum osiągane jest dla $s = 0$, natomiast dla $\mu < 0$ $s = Q_1 - q$. Zatem funkcja $L_D(\mu)$ będzie miała postać:

$$L_D(\mu) = \min_{x,v} \left[\sum_{i=1}^{N} \sum_{t=1}^{T} (\bar{s}_{it} v_{it} + \bar{c}_{it}(\mu) x_{it} + h_{it} I_{it}^{+}) \right] - \mu Q(\mu)$$

przy ograniczeniach:

$$\begin{aligned} I_{i,t-1}^{+} + x_{it} - I_{it}^{+} &= d_{it} \qquad \text{dla } t = 1, \ldots, T \text{ i } i = 1, \ldots, N \\ x_{it} &\leqslant M v_{it} \\ I_{it}^{+} &\geqslant 0 \\ v_{it} &\in \{0, 1\} \end{aligned}$$

gdzie:

$$\bar{s}_{it}(\mu) = \begin{cases} s_{it} + \mu e_{it} & t = 1 \\ s_{it} & t \geqslant 2 \end{cases}, \bar{c}_{it}(\mu) = \begin{cases} c_{it} + \mu p_{it} & t = 1 \\ c_{it} & t \geqslant 2 \end{cases}, Q(\mu) = \begin{cases} Q_1 & \mu \geqslant 0 \\ q & \mu < 0 \end{cases},$$

Funkcja $L_D(\mu)$ jest funkcją wklęsłą i kawałkami liniową. Funkcję $L_D(\mu)$ można zapisać także w innej postaci, uwidoczniającej możliwą dekompozycję zadania:

$$L_D(\mu) = \sum_{i=1}^{N} L_i(\mu) - \mu Q(\mu) \tag{2}$$

gdzie:

$$L_i(\mu) = \min \sum_{t=1}^{T} (\bar{s}_{it} v_{it} + \bar{c}_{it} x_{it} + h_{it} I_{it}^{+}) \tag{3}$$

Zadanie relaksacji Lagrange'a polega na maksymalizacji $L_D(\mu)$ względem nieograniczonego μ. Ze względu na szczególną postać zadania dualna funkcja Lagrange'a może być maksymalizowana szczególnie efektywnie metodą iteracyjnej aproksymacji subgradientowej funkcji L_D złożonej w każdej iteracji z dwóch odpowiednio dobieranych kawałków liniowych. Dla dwóch wartości mnożnika, dla których subgradienty mają przeciwne znaki, wyznaczany jest punkt przecięcia aktywnych odcinków wykresu L_D w tych punktach. W uzyskanym nowym punkcie μ obliczamy wartość $L_D(\mu)$. Jeżeli w otrzymanym punkcie wartość $L_D(\mu)$ jest poniżej aproksymacji, to nowy punkt wchodzi do bieżącej pary punktów, zastępując jeden z dotychczasowych punktów, tak aby w pozostawionych punktach nachylenie funkcji było w dalszym ciągu różnych znaków. W przeciwnym przypadku, czyli jeżeli wartość $L_D(\mu)$ jest równa wartości aproksymacji, to w znalezionym punkcie znajduje się maksimum funkcji L_D, co kończy działanie algorytmu.

Rozwiązanie zadania relaksacji Lagrange'a umożliwia wyznaczenie przybliżonego dopuszczalnego rozwiązania zadania pierwotnego w wyniku zastosowania prostych heurystycznych procedur korekcyjnych [2].

3. Przyrostowy algorytm obliczania $L_D(\mu)$

Dla ustalonych wartości mnożnika μ obliczenie $L_D(\mu)$ sprowadza się do N-krotnego rozwiązania podzadań postaci klasycznego problemu WW dla pojedynczych wyrobów [4]. Należy się liczyć z faktem, że liczba iteracji doboru mnożników Lagrange'a może być znaczna. Dlatego też sprawą istotną jest jak najbardziej efektywne obliczanie wartości $L_D(\mu)$. Przykładowo, wykorzystanie klasycznego algorytmu Wagnera-Whitina daje złożoność $O(NT^2)$, natomiast wykorzystanie najefektywniejszych algorytmów [1, 4, 3] daje złożoność $O(NTlogT)$.

Zauważmy jednak, że zmiana wartości mnożnika będzie wpływać jedynie na parametry podzadań w etapie pierwszym. Można zatem spodziewać się, że jest możliwe uniknięcie powtarzania wielu kosztownych obliczeń związanych z etapami dalszymi. Chcemy zatem opracować odpowiedni algorytm przyrostowy, w którym będziemy mogli wykorzystywać rezultaty z poprzednich iteracji uruchamiając tylko jeden krok algorytmu wieloetapowego, krok związany z etapem pierwszym. Właściwości takie zapewnia regresywny algorytm WHK Wagelmansa-Hoesela-Kolena [4]. W rezultacie, jak pokażemy, zredukujemy złożoność obliczeniową jednego kroku algorytmu do $O(N \log T)$.

3.1. Opis algorytmu WHK

Dla uproszczenia rozważań pomijamy indeks pojedynczego wyrobu oraz zależność współczynników s_i, c_i od μ. Można też pokazać, że problem ogólny można przetransformować do problemu równoważnego z zerowymi kosztami magazynowania[1] [4]. Rozważamy zatem problem decyzyjny (1) z $h_t = 0$. Wprowadzamy wielkość zapotrzebowania zakumulowanego: $d_{ij} = \sum_{t=i}^{j} d_t$.

Podstawową rolę w algorytmie WHK odgrywa funkcja $G(t)$ oznaczająca minimalny koszt dojścia od etapu t do etapu końcowego T, przy założeniu zerowego stanu zapasów $I_t = 0$. Jest ona zdefiniowana następująco:

$$G(t) = \begin{cases} \min\limits_{t<i\leqslant T+1} (s_t + c_t d_{t,i-1} + G(i)) & \text{jeżeli } d_t > 0 \\ \min\left[G(t+1), \min\limits_{t+1<i\leqslant T+1} s_t + c_t d_{t,i-1} + G(i)\right] & \text{jeżeli } d_t = 0 \end{cases}$$

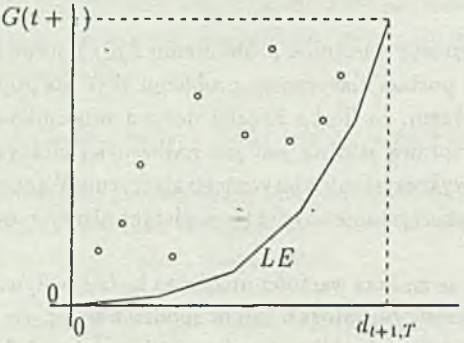
dla celów algorytmu dodaje się sztuczny etap $T+1$. Zachodzi oczywiście: $G(T+1) = 0$.

Bezpośrednie wyznaczenie wartości minimum we wzorze na $G(t)$ wymagałoby $T-t+1$ porównań, co prowadziłoby do całkowitej złożoności tej operacji równej $O(T^2)$. Może ona być zredukowana do $O(\log T)$.

Na rysunku 1 pokazano wykres wielkości $G(t)$ w zależności od zapotrzebowań zakumulowanych d_{tT}. Zaznaczone punkty oznaczają właśnie pary liczb $(d_{tT}, G(t))$ wyznaczone dla kolejnych etapów, poczynając od etapu $T+1$, a kończąc na etapie 1.

Linią ciągłą, oznaczoną jako LE, zaznaczono uwypuklenie dolne zbioru tych punktów. Linia LE jest wykresem pewnej funkcji g: $g(z) = w \leftrightarrow (z, w) \in LE$. Jak widać, musi być

[1] Podstawiając: $c_t := c_t + \sum_{i=t}^{T} h_i$. Wartość funkcji celu ulega zmianie o stałą $-\sum_{t=1}^{T} h_t \sum_{i=1}^{t} d_i$.



Rys. 1. Minimalny koszt względem zapotrzebowania zakumulowanego
Fig. 1. Minimum-cost function with respect to accumulated demand

ona funkcją wypukłą. Punktom, w których funkcja g zmienia swoje nachylenie, odpowiadają pewne etapy harmonogramu. Niech r oznacza liczbę punktów zalamania wykresu g poczynając od etapu $t+1$. Wówczas przez $t+1 = t(1) < \ldots < t(r) = T+1$ oznaczymy etapy odpowiadające tym zalamaniom i nazwiemy je *etapami znaczącymi* (ang. *efficient periods*).

Dla każdego etapu znaczącego (punktu zalamania wykresu) wyznaczamy punkt przecięcia prostej pionowej przechodzącej przez punkt $(d_{tT}, 0)$ z prostą o nachyleniu c_t przechodzącą przez odpowiedni punkt zalamania wykresu g. Punkt przecięcia o najmniejszej pionowej współrzędnej wyznacza $\min_{1 \leqslant p \leqslant r}\{c_t d_{t,t(p)-1} + G(t(p))\}$. Niech $t(q)$ oznacza etap znaczący, dla którego osiągane jest to minimum:

$$q := \min\left[r, min\left\{p : 1 \leqslant p < r \wedge \frac{G(t(p)) - G(t(p+1))}{d_{t(p),t(p+1)-1}} < c_t\right\}\right]$$

$t(q)$ ma oczywisty sens następnego etapu wznowienia produkcji po etapie t. Warto zauważyć, że w momencie gdy poszukujemy wartości funkcji optymalnego kosztu dojścia dla etapu t, mamy dany zbiór etapów znaczących będący podzbiorem etapów rozważanych poprzednio: $t+1, \ldots, T$ oraz że są one w sposób naturalny uporządkowane względem nachyleń odcinków prostoliniowych:

$$\frac{G(t(p)) - G(t(p+1))}{d_{t(p),t(p+1)-1}}, \quad p = 1, \ldots, r-1.$$

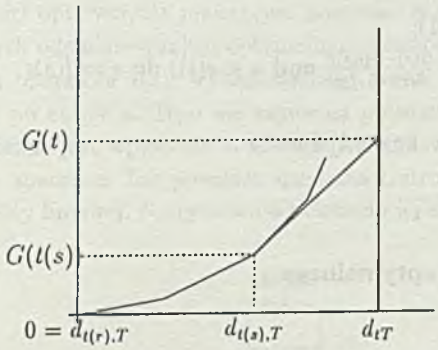
Wniosek jest więc taki, iż możliwe jest wyznaczenie $t(q)$ w czasie $O(\log T)$.

Załóżmy, że wyznaczona została wartość funkcji $G(t)$. Algorytm musi teraz przejść do rozważenia etapu $t-1$, uaktualnić zbiór etapów znaczących i wyznaczyć $G(t-1)$. Dla celów ilustracyjnych możemy zastosować następującą procedurę geometryczną:

- dodaj punkt $(d_{tT}, G(t))$,

- znajdź najmłodszy etap znaczący $t(s)$,taki że nachylenie odcinka łączącego $(d_{tT}, G(t))$ z $(d_{t(s),T}, G(t(s)))$ jest większe niż nachylenie odcinka łączącego $(d_{t(s+1),T}, G(t(s+1)))$ z $(d_{t(s),T}, G(t(s)))$,
- nowy zbiór etapów znaczących składa się z etapu t i etapów od $t(s)$ do $t(r)$.

Przebieg procedury pokazano na rysunku 2.



Rys. 2. Uaktualnianie zbioru etapów znaczących
Fig. 2. Upgrading of the set of efficient periods

By dokładniej zdefiniować etap $t(s)$, należy jeszcze rozważyć przypadek, gdy $d_t = 0$. Wówczas etap znaczący $t+1$ jest zastępowany przez etap t. W tym wypadku jest więc: $s = 2$. Ogólnie: $s \leqslant q$. Definiujemy więc:

$$s := \min\left[q, \min\left\{p : t+1 \leqslant p < q, d_{t,t(p)-1} > 0 \wedge \frac{G(t) - G(t(p))}{d_{t,t(p)-1}} > \frac{G(t(p)) - G(t(p+1))}{d(t(p), t(p+1)-1}\right\}\right]$$

Ostatecznie, algorytm WHK można zapisać w sposób następujący:

Inicjalizacja
Policz c_t, d_{tT} dla $t = 1, \ldots, T$ oraz COR;
Wstaw etap $T+1$ do L;

Iteracje
for $t := T$ **downto** 1 **do**
begin
Szukaj $q(t) := \min\left[T+1, \min\left\{p \in L : p < T+1 \wedge \frac{G(p)-G(l(p))}{d_{pT}-d_{l(p),T}} < c_t\right\}\right]$;
$G(t) := s_t + c_t[d_{tT} - d_{q(t),T}] + G(q(t))$;
if ($d_t = 0$ **and** $G(t+1) < G(t)$) **then**
begin

$G(t) := G(t+1)$;
$s := l(t+1)$;
end
else
begin
if $d_t > 0$ then
$s := t+1$
else
$s := l(t+1)$;
while $\left(\frac{G(t)-G(s)}{d_{tT}-d_{sT}} \leqslant \frac{G(s)-G(l(s))}{d_{sT}-d_{l(s),T}}\right.$ and $\left.s < q(t)\right)$ do $s := l(s)$;
end
Usuń z L wszystkie p, takie że $t+1 \leqslant p < s$;
Wstaw t do L;
end.

Odtworzenie harmonogramu optymalnego
$t := 1$;
while $t \leqslant T$ do
if ($d_t = 0$ and $G(t) = G(t+1)$) then
$t := t+1$
else
begin
$x_t := d_{tT} - d_{q(t),T}$;
$s_t := 1$;
$t := q(t)$;
end.

W powyższym opisie L oznacza specjalną strukturę danych, w której przechowywane są etapy znaczące, która, jak wiemy, uporządkowana jest z definicji względem nachyleń odcinków wykresu funkcji g. Operator $l(p)$ wyznacza następny po p etap znaczący znajdujący się w L.

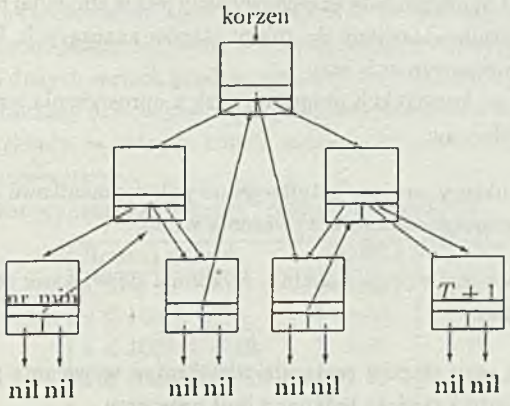
3.2. Implementacja algorytmu

Opracowano dwie wersje algorytmu WHK różniące się między sobą strukturą operacyjną i szczegółami algorytmicznymi. Pierwsza, podstawowa wersja jest zgodna z oryginalnym opisem algorytmu i matematycznym oszacowaniem złożoności algorytmu, druga zaś jest modyfikacją opierającą się na wynikach eksperymentów i specyficznych własnościach działania algorytmu.

Wersja podstawowa. Aby całkowita złożoność algorytmu była prawie liniowa (tzn. $O(T \log T)$), dwie kluczowe operacje w każdej iteracji algorytmu muszą być wykonane w czasie logarytmicznym. Te dwie operacje to wstawianie nowego etapu znaczącego do

L-struktury lub usuwanie etapów już nie będących znaczącymi oraz wyszukiwanie w tej strukturze etapu $q(t)$ o możliwie najmniejszym indeksie, dla którego nachylenie funkcji g będzie mniejsze od kosztu produkcji c_t. Spełnienie tego warunku jest możliwe tylko pod warunkiem zrealizowania wstawiania i wyszukiwania binarnego, co w oczywisty sposób prowadzi do zastosowania drzewa binarnego jako L-struktury.

Etapy znaczące w drzewie są uporządkowane względem swoich numerów oraz, co za tym idzie, względem wielkości nachyleń odcinków liniowych funkcji g. Jak zostało pokazane w części opisowej, dla malejących numerów etapów znaczących rosną nachylenia prostoliniowych odcinków funkcji optymalnych kosztów. Wylania się jednak konieczność zrealizowania operatora $l(s)$, wyznaczającego numer etapu znaczącego występującego bezpośrednio po etapie s. Tego nie zapewnia podstawowa struktura drzewa binarnego, dlatego konieczne jest wplecenie w drzewo listy liniowej, w której uszeregowane byłyby kolejno etapy znaczące. Tak powstała specjalna L-struktura będąca połączeniem drzewa binarnego i listy liniowej. Na rysunku 3 pokazano jej schemat.



Rys. 3. Schemat L-struktury
Fig. 3. L-structure scheme

Najbardziej skrajne elementy w drzewie to oczywiście sztuczny etap $T + 1$ oraz etap znaczący o najmniejszym dotąd indeksie – oznaczony na schemacie jako "nr min". Zmienna **korzen** przechowuje adres korzenia drzewa. Każdy element drzewa posiada trzy pola wskaźnikowe: dwa wskazujące na lewe i prawe poddrzewo oraz jedno na etap chronologicznie następny w strukturze. Utrzymanie wysokiej efektywności obliczeniowej, jaką oferuje drzewo binarne, wymaga jednak zachowania wyważenia drzewa. Tak więc po każdym wstawieniu i usunięciu elementu należy, o ile jest to konieczne, przywracać wyważenie drzewa uruchamiając specjalne procedury rekurencyjne.

Modyfikacja algorytmu. Omówiony powyżej algorytm wymaga dość zaawansowanych zabiegów programistycznych skupiających się jedynie wokół obsługi L-struktury, która za-

pewnia najmniejszą teoretycznie złożoność obliczeniową algorytmu. Implementacja oparta na strukturze drzewiastej jako podstawowej strukturze operacyjnej w algorytmie nie jest jednak łatwa. Prowadzi ponadto do stosunkowo dużego zużycia pamięci operacyjnej, co nie jest bez znaczenia, gdy pomyśli się nie tyle może o wielkiej liczbie etapów horyzontu planowania, co o dużym zestawie wyrobów powiązanych ze sobą wspólnymi zasobami produkcyjnymi.

Doświadczenie wskazuje, że dla realnych danych produkcja "na zapas" dokonuje się najwyżej na kilka etapów naprzód, gdyż przy większych partiach dominować zaczynają koszty obciążenia magazynu i zamrożenia produkcji. W takim razie, jeżeliby w poszukiwaniu $q(t)$ przeszukiwać etapy znaczące kolejno, poczynając od ostatnio wstawionego do L-struktury, to sprawdzanych etapów będzie kilka. Zważywszy, iż aby dokonać tego samego w drzewie, musieliśmy dojść w poszukiwaniach aż do liścia, strata będzie niewielka, bądź nawet osiągniemy zysk.

Nasuwa się zatem zastosowanie prostej struktury danych, jaką jest zwykła lista jednokierunkowa pełniąca rolę L-struktury. Pola jej rekordów zawierają więc tylko po jednym polu wskaźnikowym. Adres początku listy przechowywany jest w zmiennej `poczatek_listy` i wskazuje on na etap ostatnio wstawiony do zbioru etapów znaczących. Po zakończeniu działania algorytmu jest nim oczywiście etap 1.

Z punktu widzenia samej konstrukcji programu zysk z uproszczenia struktury danych jest oczywisty i bardzo widoczny:

- Pola rekordów L-struktury zawierają tylko jedno pole wskaźnikowe oraz nie muszą posiadać pola mieszczącego znacznik wyważenia węzła.

- Procedura wstawiania jest wprost banalna, wykonuje się w czasie stałym i zawiera tylko 28 linii programu.

- Procedura usuwania serii etapów zastąpiła wielokrotne wywołania procedury usuwania jednego elementu i zawiera jedynie 8 linii programu.

3.3. Implementacja algorytmu dla wielu rodzajów wyrobów

W momencie gdy w miejsce jednego wyrobu będziemy mieć ich pewną liczbę oraz będziemy chcieli wykorzystywać możliwie dużo informacji z poprzednich iteracji poszukiwań optymalnego mnożnika, muszą ulec zwielokrotnieniu wszystkie struktury danych używane w algorytmie podstawowym. Informacje o tych strukturach przechowywane są w liście liniowej jednokierunkowej, której element zawiera pola:

- numer wyrobu,

- koniec listy parametrów etapów dla danego wyrobu,

- wskaźnik na etap pierwszy na tej liście,

- wskaźnik na korzeń L-struktury dla danego wyrobu,

- adres szczytu stosu dla harmonogramu najlepszego niedopuszczalnego,
- adres szczytu stosu dla harmonogramu dopuszczalnego,
- adres elementu ostatnio wstawionego do L-struktury,
- adres wyrobu następnego.

Ponadto listy parametrów etapów rozszerzono o wielkości zużycia zasobu przy wznowieniu i produkcji oraz o pola przechowujące pierwotne wartości kosztów wznowienia i produkcji, co ma znaczenie dla etapu pierwszego, w którym parametry te zmieniają się przy każdej zmianie mnożnika Lagrange'a.

4. Wyniki eksperymentalne

Dla porównania szybkości obu realizacji algorytmu WKH przeprowadzono szereg eksperymentów dla przesadnie dużej liczby etapów $T = 3000$. Ponieważ podejrzewano, że druga realizacja może okazać się wolna w sytuacjach, gdy z danych zadania będą wynikały harmonogramy o dużych seriach produkcyjnych (wtedy procedury usuwania i wyszukiwania etapu q będą musiały prawdopodobnie przejrzeć duże liczby etapów), szczególną uwagę zwrócono na przykłady, w których koszty wznowienia produkcji dominują znacznie nad kosztami magazynowania.

Oto wyniki eksperymentów — czasy[2] podane w sekundach:

Rodzaj danych	WHK1	WHK2
$s, h \leqslant 10$	1.49	0.65
$s \leqslant 100, h \leqslant 10$	1.60	0.83
$s \leqslant 1000, h \leqslant 10$	1.76	1.10
$s \leqslant 10000, h \leqslant 10$	1.70	1.15
$s \leqslant 100000, h \leqslant 10$	1.65	1.21
$s \leqslant 100000, h = 1$	1.70	1.21
$s \leqslant 1000000, h \leqslant 10$	1.70	1.31
$s \leqslant 10000000, h \leqslant 10$	1.43	1.26

Można zauważyć, iż w przypadku realizacji drzewiastej czas obliczeń pozostaje na pewnym, niezmiennym poziomie niezależnie od proporcji między kosztami wznowienia i magazynowania. Dla realizacji listowej natomiast czas ten, gdy weźmiemy dwa skrajne przykłady, wzrósł dwukrotnie. Jednakże i on, jak się wydaje, podlega pewnej stabilizacji i mimo wszystko pozostaje krótszy. Dla danych z przykładu pierwszego, gdzie koszty losowane były z jednakowego przedziału, czas obliczeń wykonywanych przez implementację listową był ponad dwukrotnie krótszy! Warto też zauważyć, że liczba etapów rozważana w przykładach (3000) jest bliska górnej granicy dopuszczalnego przez implementację drzewiastą rozmiaru zadania. Wniosek jest więc taki, że dla dopuszczanych przez implementację rozmiarów zadania nie uwidoczni się jej przewaga. Obie wersje algorytmu

[2]Mierzono czas wykonania zasadniczej części algorytmu, bez wczytywania danych z pliku, odtwarzania harmonogramu i zapisywania go w pliku (mikrokomputer IBM PC/386, 33 MHz).

porównywano również szczegółowo na wielu seriach bardziej realistycznych klas zadań o zróżnicowanych wartościach współczynników [2] dla $T = 10, 20, 50, 100, 200$. Wyniki obliczeń potwierdzają przewagę algorytmu zmodyfikowanego dla wszystkich klas zadań.

Omówione algorytmy mogą być wykorzystywane w algorytmie przyrostowym przy rozwiązywaniu zadań dla wielu wyrobów. Podczas wielokrotnego obliczania $L_D(\mu)$ koszt rozwiązania podproblemu dla pojedynczego wyrobu jest skracany T razy i trwa tylko pojedyncze milisekundy (na IBM PC/386). W przypadku zastosowania przedstawionego w pracy algorytmu w systemach planowania produkcji (MRP II) będzie możliwe rozwiązywanie zadań praktycznych o znacznie większych rozmiarach niż dotychczas.

LITERATURA

[1] Awi Federgruen, Michal Tzur, A Simple Forward Algorithm to Solve General Dynamic Lot Sizing Models with n Periods in $O(n \log n)$ or $O(n)$ Time, *Management Science*, Vol.37, No.8, August 1991.

[2] Krzysztof Maik, *Wybrane algorytmy harmonogramowania produkcji porcjami*, praca magisterska, Instytut Automatyki PW, Warszawa, 1994.

[3] Eugeniusz Toczyłowski, Algorytmy harmonogramowania produkcji porcjami o niskiej złożoności obliczeniowej, *Zeszyty Naukowe Politechniki Śląskiej*, nr kol. 1175, 1992

[4] Albert Wagelmans, Stan van Hoesel, Antoon Kolen, Economic Lot Sizing: An $O(n \log n)$ Algorithm that Runs in Linear Time in the Wagner-Whitin Case, *Operation Research*, Vol.40, Supp., No.1, January-February 1992

[5] Wagner, M. H., T. M. Whitin, 'Dynamic version of the economic lot-sizing model', *Management Science.*, Vol. 5 (1958), 89–96.

Recenzent: Prof.dr hab.inż. Konrad Wala

Wplynęło do Redakcji do 30.04.1994 r.

Abstract

The paper presents an efficient scheduling method for the multi-item dynamic lot-size scheduling problem, in which a single limited resource is considered.

The limitations of the resource in subsequent periods are transformed into a lower bound for the resource limitation in the first period. As a result, an approximate relaxed problem, with doubly constrained limitation in the first period is formulated. This problem may be efficiently solved by a specialized method, based on Lagrange decomposition. The method uses particular properties of the problem and particular features of the Wagelmans-Hoesel-Kolen algorithm for solving decomposed subproblems. In comparison to the best existing methods, this one allows to reduce the complexity of the algorithm T times, where T is a number of periods.

Two distinct variants of the WHK algorithm for solving decomposed subproblems are implemented. Extensive numerical experiments show a superiority of the modified algorithm over the original WHK algorithm.