ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1986
Seria: AUTOMATYKA z. 84 Nr kol.894

Eugeniusz Nowicki, Stanisław Zdrzałka
Politechnika Wrocławska
Instytut Cybernetyki Technicznej

PROBLEMY SZEREGOWANIA ZE ZMIENNYMI CZASAMI WYKONYWANIA ZADAŃ*

Streszczenie. W pracy sformułowano klasę problemów szeregowania zadań na maszynach, w której modelem zadania jest liniowa funkcja koszt/czas oraz występują dwie funkcje celu: wskaźnik jakości uszeregowania oraz koszt wykonania zadań. Dla klasy tej podano wyniki dotyczące złożoności obliczeniowej, algorytmów wielomianowych oraz przedstawiono systematyczny przegląd istniejących wyników.

1. Wstęp

W pracy przedstawiono klasę problemów szeregowania zadań na maszynach, w której tradycyjny dla teorii szeregowania model zadania /operacji/, podawany w postaci ustalonego czasu wykonywania, zastąpiony został przez liniową funkcję koszt/czas określoną na domkniętym i ograniczonym zbiorze dopuszczalnych czasów wykonywania. Dla tak sformułowanego modelu zadania pojawia się obok tradycyjnej zmiennej decyzyjnej /kolejności wykonywania zadań/, nowy typ zmiennej - czas wykonywania zadania. Ponadto, obok rozważanych w teorii szeregowania funkcji celu pojawia się nowa - łączny koszt wykonania wszystkich zadań. Patrząc na rozpatrywaną klasę problemów jako na zagadnienia planowania sieci czynności typu PERT/koszt - gdzie model zadania typu koszt/czas jest w powszechnym użyciu, znajdujemy je jako zagadnienia planowania sieci czynności z ograniczonymi zasobami odnawialnymi /maszyny/, podzielonymi w sposób dyskretny oraz liniowymi funkcjami koszt/czas.

Z uwagi na mieszany charakter zmiennych decyzyjnych, zmieniające się w sposób ciągły czasy wykonywania zadań oraz kolejność wykonywania zadań - zmiana typu kombinatorycznego, rozpatrywane problemy zaliczają się na ogół do klasy problemów trudnych, przy czym granica pomiędzy problemami o złożoności obliczeniowej wielomianowej a problemami NP-trudnymi leży "niżej" niż w klasie czystych problemów kolejnościowych.

Niniejsze opracowanie jest próbą stworzenia jednolitego spojrzenia na problemy z tej klasy oraz próbą podsumowania dotychczasowych fragmentarycznych wyników dotyczących złożoności obliczeniowej, algorytmów wielomianowych heurystycznych oraz dokładnych, typu podziału i ograniczeń.

* praca była częściowo finansowana przez RP.I.02 "Teoria sterowania i optymalizacja ciągłych układów dynamicznych i procesów dyskretnych"

2. Sformułowanie problemu

Niech $J = \{1,2,\ldots,n\}$ będzie zbiorem indeksów zadań a $M = \{1,2,\ldots,m\}$, zbiorem indeksów maszyn. Każde zadanie może być wykonywane w danej chwili na co najwyżej jednej maszynie oraz każda z maszyn może w danej chwili wykonywać co najwyżej jedno zadanie. Wykonywanie zadania j na maszynie i zajmuje $a_{ij} - x_{ij}$ jednostek czasu, przy czym $a_{ij} > 0$ jest dane, natomiast x_{ij} - zmienna decyzyjna przyjmująca wartości z przedziału $[0,u_{ij}]$, $u_{ij} \leq a_{ij}$. Zmienna x_{ij} określa liczbę jednostek czasu o jaką skrócono czas wykonywania a_{ij}; dalej nazywać ją będziemy czasem skrócenia zadania j na maszynie i. Jeżeli każde zadanie j składa się ze skończonego ciągu operacji $\langle O_{1j},\ldots,O_{m_j j}\rangle$ oraz dana jest funkcja μ_{ij} /$\mu_{ij} \neq i$, $m_j \neq m$/ przyporządkowująca każdej operacji O_{ij} indeks maszyny μ_{ij}, to $a_{ij} - x_{ij}$ określa czas trwania operacji O_{ij} na maszynie μ_{ij}. Dla uniknięcia nadmiernej no acji pozostajemy jednak dalej przy poprzedniej interpretacji czasu $a_{ij} - x_{ij}$. Zauważmy, że eliminujemy wtedy z naszych rozważań klasę problemów typu "job-shop"; rozważania te jednak można prosto przenieść na tę klas; problemów.

Koszt skrócenia czasu wykonywania zadania j na maszynie i wynosi $c_{ij}x_{ij}$, $c_{ij} > 0$. W dalszym ciągu koszt ten będziemy utożsamiać z kosztem wykonania zadania j na maszynie i; w rzeczywistości koszt wykonania jest sumą pewnego stałego składnika i kosztu skrócenia, jednakże w zagadnieniach optymalizacji stawianych w dalszym ciągu stały składnik nie ma wpływu na rozwiązanie optymalne i dlatego może być pominięty. Niech $X = \{x = (x_{11},\ldots,x_{1n},\ldots,x_{m1},\ldots,x_{mn}) : 0 \leq x_{ij} \leq u_{ij},\ i \in M,\ j \in J\}$, gdzie u_{ij} jest maksymalnym dopuszczalnym skróceniem. Koszt wykonania wszystkich zadań dla pewnego $x \in X$ wynosi

$$F_2(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}x_{ij} . \qquad /1/$$

Zakładać będziemy dalej, że wykonywanie każdego zadania na każdej z maszyn odbywa się bez przerwań. Oznaczmy przez φ funkcję przyporządkowującą każdej maszynie pewien podzbiór zadań oraz porządek liniowy określony na tym podzbiorze, który określa kolejność w jakiej zadania będą wykonywane na tej maszynie. W przypadku gdy każde zadanie wykonywane jest na każdej z maszyn i kolejność wykonywania zadań na każdej z maszyn jest jednakowa, wówczas w miejsce φ wykorzystywać będziemy π, permutację na zbiorze J; przez $\pi(i)$ oznaczamy indeks zadania stojącego na pozycji i w π. Niech Φ będzie zbiorem wszystkich dopuszczalnych φ; w przypadku π, zbiór ten oznaczamy przez Π.

W dalszym ciągu rozpatrywać będziemy klasę problemów, w której drugim obok F_2 wskaźnikiem jest wskaźnik jakości uszeregowania. Wskaźnik ten jest zazwyczaj definiowany następująco. Niech C_j oznacza moment zakończe-

nia wykonywania zadania j. Zauważmy, że przy założeniu o nieprzerywalności wykonywania zadań, φ oraz x w sposób jednoznaczny określają uszeregowanie /czyli także C_j/ - w tradycyjnym rozumieniu tego pojęcia [1]. Załóżmy dalej, że $f_j : R \to R$ będzie funkcją przyporządkowującą każdemu C_j karę $f_j(C_j)$ za zakończenie wykonywania zadania j w chwili C_j. Przyjmujemy, że f_j jest funkcją niemalejącą, $j \in J$. W literaturze, patrz np. [1], zwykło się definiować wskaźnik jakości uszeregowania oznaczany dalej przez F_1, jako $F_1(\varphi, x) \equiv \max_{1 \le j \le n} f_j(C_j)$ albo $F_1(\varphi, x) \equiv \sum_{j=1}^{n} f_j(C_j)$; wskaźniki te oznaczane są symbolicznie przez, odpowiednio, f_{max} i $\sum f_j$. Funkcje f_j definiowane są zazwyczaj jako $f_j(C_j) = C_j, L_j, T_j, U_j, w_j C_j, w_j L_j, w_j T_j, w_j U_j$, gdzie $w_j > 0$ oraz $L_j = C_j - d_j$ - nieterminowość wykonania zadania j względem zadanego pożądanego terminu wykonania d_j, $T_j = \max\{L_j, 0\}$ - opóźnienie, $U_j = 1$ jeżeli $L_j \geq 0$ oraz $U_j = 0$ w przeciwnym przypadku - jednostkowa funkcja kary.

Ogólny problem szeregowania przy zmiennych czasach wykonywania zadań stawiamy następująco. Niech Ω będzie zbiorem wszystkich uporządkowanych par $\langle F_1(\varphi, x), F_2(x)\rangle$ dla $\varphi \in \Phi$ i $x \in X$. Punkt $(\alpha, \beta) \in \Omega$ jest efektywny, jeżeli nie istnieje taki wektor $(\alpha', \beta') \in \Omega$, że $\alpha' \leq \alpha$ i $\beta' \leq \beta$ oraz przynajmniej jedna z nierówności jest ostra. Niech $P \subset \Omega$ będzie zbiorem wszystkich punktów efektywnych. Zauważmy, że każda para $(\alpha, \beta) \in P$ ma tę własność, że każde uszeregowanie, które daje koszt wykonania zadań mniejszy od β równocześnie powoduje, że wskaźnik jakości uszeregowania jest większy od α i na odwrót. Zatem P jest zbiorem punktów kompromisu pomiędzy kosztem wykonania a jakością uszeregowania.

/P1/ Problem 1. Znaleźć zbiór P oraz zbiór W par $(\varphi, x) \in \Phi \times X$ takich, że jeżeli $(\alpha, \beta) \in P$ to $\alpha = F_1(\varphi, x)$, $\beta = F_2(x)$.
Ponadto, stawiamy następujące zagadnienia.

/P2/ Problem 2. Dla zadanego T znaleźć parę $(\varphi^0, x^0) \in \Phi \times X$ taką, że $F_2(x^0) = \min_{\varphi \in \Phi, x \in X} F_2(x)$ oraz $F_1(\varphi^0, x^0) \leq T$.

/P3/ Problem 3. Dla zadanego K znaleźć parę $(\varphi^0, x^0) \in \Phi \times X$ taką, że $F_1(\varphi^0, x^0) = \min_{\varphi \in \Phi, x \in X} F_1(\varphi, x)$ oraz $F_2(x^0) \leq K$.

/P4/ Problem 4. Dla ustalonych $\omega_1, \omega_2 > 0$ znaleźć parę $(\varphi^0, x^0) \in \Phi \times X$ taką, że $\omega_1 F_1(\varphi^0, x^0) + \omega_2 F_2(x^0) = \min_{\varphi \in \Phi, x \in X} \omega_1 F_1(\varphi, x) + \omega_2 F_2(x)$.

Problemy te były badane już od dawna ale w kontekście zagadnień planowania sieci czynności, np. [3]. Na przykład Problem 1 odpowiada szukaniu krzywej koszt/czas wykonania projektu w zagadnieniach PERT/koszt. Przypominamy, że problemy rozpatrywane w pracy można uważać jako zagadnienia planowania sieci czynności, w których występują ograniczone zasoby odnawialne typu maszyn. Jak wiadomo, wprowadzenie takiego ograniczenia w istotny sposób wpływa na model matematyczny zagadnienia i w konsekwencji

na złożoność obliczeniową. Niektóre z powyższych problemów były rozpatrywane przy założeniu, że X jest zbiorem skończonym /tzw. sposoby wykonywania operacji/, na przykład w pracach [7],[8].

Problemy postawione w niniejszej pracy badane były przez nielicznych autorów poczęwszy od 1980 roku - inicjujące prace Vicksona [11],[12] - mimo, że już od dawna wskazywano na ich wagę. Interpretacja tych problemów jest podobna jak w zagadnieniach planowania sieciowego. Mianowicie, P1 odpowiada szukaniu krzywej kompromisu pomiędzy kosztem wykonania zadań a jakością uszeregowania, P2 to minimalizacja ilości zasobu nieodnawialnego jednego rodzaju przy założonej jakości uszeregowania /np. założony czas zakończenia wykonania wszystkich zadań/, P3 to minimalizacja wskaźnika jakości uszeregowania /np. minimalizacja czasu zakończenia wykonywania wszystkich zadań/ przy ograniczonej ilości zasobu nieodnawialnego, i na koniec, P4 odpowiada minimalizacji łącznego kosztu wykonania wszystkich zadań.

W dalszym ciągu używać będziemy następującej terminologii. Niech $\mathcal{G}$ oznacza zagadnienie szeregowania zadań z pewną pojedynczą funkcją celu typu f_{max} albo $\sum f_j$. Zagadnienia $\mathcal{G}$ będziemy oznaczać za pomocą znanego zapisu symbolicznego [1], np. $\mathcal{G} = 1||C_{max}$, $P2||L_{max}$, itp. Będziemy mówić, że rozważamy problem P2 /P1, P3, P4/ dla zagadnienia $\mathcal{G}$, jeżeli model matematyczny problemu jest taki jak w $\mathcal{G}$, z tym tylko wyjątkiem, że teraz $p_{ij} = a_{ij} - x_{ij}$, funkcja celu F_1 jest zdefiniowana tak jak w $\mathcal{G}$, a F_2 zadane jest przez /1/.

Podamy teraz pewne związki między problemami P1 ÷ P4 dla ustalonego zagadnienia $\mathcal{G}$. W tym celu oznaczmy: $T_1 = \min_{\varphi \in \Phi} F_1(\varphi, u)$, $T_2 = \min_{\varphi \in \Phi} F_1(\varphi, 0)$. T_1 i T_2 zawsze istnieją ponieważ Φ jest zbiorem skończonym.

Twierdzenie 1. Jeżeli F_1 jest funkcją ciągłą oraz nierosnącą ze względu na każde x_{ij} dla ustalonych pozostałych składowych x oraz dla ustalonego $\varphi \in \Phi$, to $\{(T,G(T)) : T_1 \leq T \leq T_2\} = P$, gdzie dla dowolnego $T \in [T_1,T_2]$, $G(T) = \min\{F_2(x) : x \in X$, istnieje $\varphi \in \Phi$ takie, że $F_1(\varphi, x) \leq T\}$.

Zauważmy, że G(T) jest minimalną wartością funkcji celu w P2 dla ustalonego T a zatem Twierdzenie 1 mówi, że w celu znalezienia zbioru wszystkich punktów efektywnych P wystarczy rozwiązać P2 dla wszystkich T z przedziału $[T_1,T_2]$. Można podać przykłady na to, że odwrócona procedura znajdowania P, to znaczy poprzez rozwiązywanie P3 dla wszystkich K z odpowiedniego przedziału, prowadzi do punktów, które nie są efektywne. Można jednak wykazać, że procedura ta daje zbiór wszystkich punktów efektywnych w słabym sensie. Założenie o ciągłości i monotoniczności F_1 jest spełnione w większości rozpatrywanych powszechnie modeli.

Należy jeszcze przypomnieć znany fakt, że rozwiązanie P4 dla dowolnych $w_1, w_2 > 0$ daje punkt efektywny. Ponadto zauważmy, że kładąc $w_1=w_2=1$

nie zmniejszamy ogólności rozważań.

W dalszym ciągu będziemy mówić o wielomianowym algorytmie rozwiązującym P1 dla zagadnienia $\mathcal{G}$ /w którym należy znaleźć nieskończony zbiór P i W/ tylko w takiej sytuacji kiedy zbiory P i W są określone jednoznacznie przez skończoną liczbę parametrów, liczba ta jest ograniczona przez wielomian od rozmiaru zagadnienia $\mathcal{G}$, oraz czas wymagany przez algorytm dla znalezienia każdego z parametrów jest ograniczony przez wielomian od rozmiaru zagadnienia $\mathcal{G}$.

3. Problemy jednomaszynowe z funkcją celu f_{max}

Przedstawimy w tym punkcie wyniki dotyczące algorytmów i złożoności obliczeniowej problemów P1 ÷ P4 dla zagadnień jednomaszynowych, w których funkcją celu jest f_{max}.

$1||C_{max}$. Ponieważ w tym przypadku P_1 nie zależy od kolejności wykonywania zadań π, problemy P2 ÷ P4 stają się prostymi zadaniami programowania liniowego. W przypadku P4, π^o jest dowolną permutacją a $x_j^o = u_j$ dla $c_j < 1$ oraz $x_j^o = 0$ dla $c_j \geq 1$. Rozważmy problem P2. Należy znaleźć rozwiązanie zadania $\min \{ \sum_{j=1}^n c_j x_j : \sum_{j=1}^n x_j \geq \sum_{j=1}^n a_j - T,\ 0 \leq x_j \leq u_j, j \in J \}$; π^o jest dowolne.

Algorytm 1

1. Podstaw $x_j^o = 0$, $j \in J$; $\Delta = \sum_{j=1}^n a_j - T$; $k = 1$; niech $i_1,\ldots,i_n$ będą indeksami zadań uporządkowanymi wg niemalejących c_j.
2. Jeżeli $k = n + 1$ lub $\Delta \leq 0$ to stop /jeżeli $k = n + 1$ i $\Delta > 0$, zadanie nie ma rozwiązania/. W przeciwnym przypadku podstaw $x_{i_k}^o = \min\{\Delta, u_{i_k}\}$; $\Delta := \Delta - x_{i_k}^o$; $k := k + 1$ i przejdź do 2.

Złożoność obliczeniowa tego algorytmu wynosi $O(n \log n)$. Podobnie prosty algorytm, o tej samej złożoności, można podać dla P3. Algorytm 1 rozwiązuje również zadanie P1, jeżeli w miejsce T podstawimy $T = T_1 = \sum_{j=1}^n (a_j - u_j)$. Wtedy $x_j^o = u_j$, $j \in J$. Niech $y^j \in R^n$ będzie punktem, którego składowe $i_1,\ldots,i_j$ są równe $u_{i_1},\ldots,u_{i_j}$ a pozostałe składowe są zerowe, $j \in J$, $y^o = 0$. Niech $\alpha_k = \sum_{j=1}^n (a_j - y_j^k)$ a $\beta_k = \sum_{j=1}^n c_j y_j^k$, $k = 0,1,\ldots,n$. Wtedy zbiór P jest sumą n odcinków łączących punkty (α_k, β_k), $(\alpha_{k+1}, \beta_{k+1})$, $k = 0,\ldots,n-1$, zbiór $W = \{\pi^o\} \times X^o$, gdzie π^o jest dowolną permutacją na J a X^o jest sumą n odcinków łączących punkty y^k, y^{k+1}, $k = 0,1,\ldots,n-1$. Złożoność tego algorytmu, w sensie podanym w punkcie 2 tej pracy, wynosi $O(n \log n)$. Wszystkie podane tu rozważania przenoszą się trywialnie na zagadnienie $1|<|C_{max}$.

$1|r_j|C_{max}$. Dla dowolnego $x \in X$ oraz dla dowolnego $\pi \in \Pi$ zachodzi $F_1(\pi^o,x) \leq F_1(\pi,x)$, gdzie π^o jest permutacją otrzymaną przez uszeregowanie zadań zgodnie z niemalejącymi wartościami r_j, tzn. $r_{\pi^o(1)} \leq r_{\pi^o(2)} \leq \ldots \leq r_{\pi^o(n)}$, [1]. Wynika stąd, że π^o jest permutacją optymalną dla zadań P2 ÷ P4. Ponadto, zbiór W ma postać $\{\pi^o\} \times X^o$, gdzie X^o jest pewnym podzbiorem X. Nie zmniejszając ogólności rozważań przenumerujmy tak zadania, żeby $\pi^o = \langle 1,2,\ldots,n\rangle$. Wyznaczenie x^o dla każdego z zadań P2 ÷ P4 sprowadza się wtedy do rozwiązania odpowiedniego problemu programowania liniowego. Przykładowo zadanie P4 ma postać $\min_{x\in X}\left[\max_{1\leq j\leq n}\left(r_j + \sum_{i=j}^{n}(a_j - x_j)\right) + \sum_{j=1}^{n} c_j x_j\right]$. Zadanie to rozwiązuje poniższy algorytm o złożoności $O(n^2)$.

Algorytm 2

1. Podstaw $x_j^o = 0$, $\Delta_j = r_j + \sum_{i=j}^{n} a_i$, $1 \leq j \leq n$; $T^o = \max_{1\leq j\leq n} \Delta_j$; $K^o = 0$; $k = \max\{j: \Delta_j = T^o\}$.
2. Jeżeli $c_j \geq 1$ lub $x_j^o = u_j$, $k \leq j \leq n$ to stop.
3. Znajdź największe $k \leq l \leq n$ takie, że $c_l = \min\{c_j : k \leq j \leq n,\ x_j^o \neq u_j\}$
4. Jeżeli $u_l < T^o - \max_{1\leq j\leq n} \Delta_j$ to $x_l^o = u_l$; $T^o := T^o - x_l^o$; $\Delta_j := \Delta_j - x_l$, $k \leq j \leq l$; $K^o := K^o + c_l x_l^o$ i przejdź do 2.
   Jeżeli $u_l \geq T^o - \max_{1\leq j\leq n} \Delta_j$ to $x_l^o = T^o - \max_{1\leq j\leq n} \Delta_j$; $T^o := T^o - x_l^o$; $K^o := K^o + c_l x_l^o$; znajdź $k = \max\{j : 1 \leq j \leq n,\ \Delta_j = T^o\}$ i przejdź do 2.

Po skończeniu działania algorytmu para (π^o,x^o) jest rozwiązaniem optymalnym dla P4 oraz $F_2(x^o) = K^o$, $F_1(\pi^o,x^o) = T^o$.
W celu znalezienia x^o dla zadania P2 można wykorzystać algorytm 2 po następującej modyfikacji: wyrzucić krok 2, po wykonaniu kroku 4 przejść do 3 oraz między kroki 3 a 4 wprowadzić krok 3a. Jeżeli $\min\{T^o - \max_{1\leq j\leq n} \Delta_j, u_l\} \geq T^o - T$ to $x_l^o = T^o - T$; $T^o := T^o - x_l^o$; $K^o := K^o + c_l x_l^o$ i stop.
Złożoność tak zmodyfikowanego algorytmu pozostaje bez zmian i para (π^o,x^o) jest rozwiązaniem optymalnym zadania P2 oraz $F_2(x^o) = K^o$, $F_1(\pi^o,x^o) = T^o = T$. Warto tutaj jeszcze zauważyć, że algorytm rozwiązuje zadanie P2 tylko w przypadku $T_1 \leq T \leq T_2$, gdzie $T_1 = \max_{1\leq j\leq n}\left(r_j + \sum_{i=j}^{n}(a_i - u_i)\right)$, $T_2 = \max_{1\leq j\leq n}\left(r_j + \sum_{i=j}^{n} a_i\right)$. Jeżeli $T < T_1$ to P2 nie ma rozwiązania, a gdy $T > T_2$ to $x^o = 0$.

Podobnie modyfikacja algorytmu 2 dla zadania P3 jest następująca: z kroku 2 należy wyrzucić warunek $c_j \geq 1$ oraz między kroki 3 a 4 wprowadzić krok 3a.

3a. Jeżeli $\min\{T^o - \max_{1<j\leq n} \Delta_j, u_1\} \geq \frac{K-K^o}{c_1}$ to $x_1^o = \frac{K-K^o}{c_1}$; $T^o := T^o - x_1$; $K^o := K^o + c_1 x_1^o$ i stop.

Algorytm 2 rozwiązuje także zadanie P1 po wyrzuceniu warunku $c_j \geq 1$ z kroku 2. Oznaczmy przez $i_1, i_2, \ldots, i_s$ /$s \leq n$/ poszczególne wartości 1 wyznaczane w kroku 3 w trakcie działania algorytmu. Niech $\alpha_o = T_2$, $\beta_o = 0$ $\alpha_i = \alpha_o - \sum_{j=1}^{i} x_{i_j}^o$, $\beta_i = \sum_{j=1}^{i} c_{i_j} x_{i_j}^o$, $i = 1,\ldots,s$. Łatwo zauważyć, że zbiór odcinków łączących punkty $(\alpha_i, \beta_i), (\alpha_{i+1}, \beta_{i+1})$, $i = 0,1,\ldots, \ldots,s-1$ określa zbiór P. Zbiór W wyznaczamy następująco. Niech $y^j \in R^n$ będzie punktem, którego składowe $i_1,\ldots,i_j$ są równe $x_{i_1}^o,\ldots,x_{i_j}^o$ a pozostałe składowe są zerowe, $j = 1,\ldots,s$, $y^o = 0$. Niech zbiór $X^o \subset X$ będzie sumą s odcinków łączących punkty y^j, y^{j+1}, $j = 0,\ldots,s-1$. Mamy wtedy $W = \{\pi^o\} \times X^o$. Złożoność obliczeniowa tego algorytmu, w sensie podanym w punkcie 2 tej pracy, jest $O(n^2)$.

Przedstawione algorytmy można bezpośrednio zastosować do problemów P1 - P4 dla zagadnienia $1|r_j,\prec|C_{max}$ zmieniając tylko sposób wyboru π^o. Permutację π^o otrzymujemy tak jak dla klasycznego problemu $1|r_j,\prec|C_{max}$, dla $p_j = a_j$.

$1||L_{max}$. Dla dowolnego $x \in X$ oraz dla dowolnego $\pi \in \Pi$ zachodzi

$$F_1(\pi,x) = \max_{1\leq j\leq n}\left(\sum_{i=1}^{j}(a_{\pi(i)} - x_{\pi(i)}) - d_{\pi(j)}\right) = \max_{1\leq j\leq n}\left(\sum_{i=1}^{j}(a_{\pi(i)} - x_{\pi(i)}) + r_{\pi(j)}\right) - D = \max_{1\leq j\leq n}\left\{r_{\pi^I(j)} + \sum_{i=j}^{n}\left(a_{\pi^I(i)} - x_{\pi^I(i)}\right)\right\} - D,$$

gdzie: $D = \max_{1\leq j\leq n} d_j$; $r_j = D - d_j$, $\pi^I(j) = \pi(n-j+1)$, $1 \leq j \leq n$.

Z powyższego wynika, że zadania P1 ÷ P4 dla $1||L_{max}$ są po odpowiednich modyfikacjach, o złożoności $O(n)$, równoważne zadaniom P1-P4 dla $1|r_j|C_{max}$. Dlatego też nie będziemy omawiać szczegółowo tych zagadnień pozostawiając je czytelnikowi. To samo dotyczy przypadku $1|\prec|L_{max}$.

$1||T_{max}$. Dla dowolnego $x \in X$ oraz dla dowolnego $\pi \in \Pi$ zachodzi $F_1(\pi^o,x) \leq F_1(\pi,x)$, gdzie π^o jest permutacją otrzymaną przez uszeregowanie zadań zgodnie z niemalejącymi wartościami d_j, tzn. $d_{\pi^o(1)} \leq d_{\pi^o(2)} \leq \ldots \leq d_{\pi^o(n)}$, [1]. Podobnie jak dla problemów z $1|r_j|C_{max}$ permutacja jest permutacją optymalną dla zadań P2 - P4. Zbiór W ma postać $\{\pi^o\} \times X^o$, gdzie X^o jest pewnym podzbiorem X. Nie zmniejszając ogólności rozważań przenumerujmy zadania tak, żeby $\pi^o = \langle 1,2,\ldots,n\rangle$. Wyznaczenie x^o dla każdego z zadań P2 - P4 sprowadza się do rozwiązania odpowiedniego problemu programowania liniowego. Dla zadania P4 ma ono postać

$$\min_{x\in X}\left[\max_{1\leq j\leq n}\left(\max\left\{\sum_{i=1}^{j}(a_i - x_i) - d_j, 0\right\}\right) + \sum_{j=1}^{n} c_j x_j\right].$$

Zadanie to rozwiązuje poniższy algorytm o złożoności $O(n^2)$. Jest to uefektywniona wersja algorytmu z pracy [11].

Algorytm 3

1. Podstaw $x_j^0 = 0$; $\Delta_j = \max\left\{\sum_{i=1}^{j} a_i - d_j, 0\right\}$, $1 \le j \le n$; $T^0 = \max\limits_{1\le j\le n} \Delta_j$; $K^0 = 0$; $k = \min\left\{j : \Delta_j = T^0\right\}$.
2. Jeżeli $T^0 = 0$ to stop. Jeżeli $T^0 > 0$ oraz $c_j \ge 1$ lub $x_j^0 = u_j$, $1 \le j \le k$, to stop.
3. Znajdź najmniejsze $1 \le l \le k$ takie, że $c_l = \min\left\{c_j : 1 \le j \le k, x_j^0 \ne u_j\right\}$.
4. Jeżeli $u_l < T^0 - \max\limits_{1\le j<l} \Delta_j$, to $x_l^0 = u_l$; $T^0 := T^0 - x_l^0$; $\Delta_j := \max\left\{\Delta_j - x_l, 0\right\}$, $1 \le j \le k$; $K^0 := K^0 + c_l x_l^0$ i przejdź do 2.
   Jeżeli $u_l \ge T^0 - \max\limits_{1\le j<l} \Delta_j$, to $x_l^0 = T^0 - \max\limits_{1\le j<l} \Delta_j$; $T^0 := T^0 - x_l^0$; $K^0 := K^0 + c_l x_l^0$; znajdź $k = \min\left\{j = 1 \le j < l, \Delta_j = T^0\right\}$ i przejdź do 2.

Po zakończeniu działania algorytmu para (π^0, x^0) jest rozwiązaniem optymalnym zadania P4 oraz $P_2(x^0) = K^0$, $P_1(\pi^0, x^0) = T^0$. W celu znalezienia x^0 dla P2, P3, należy algorytm zmodyfikować w identyczny sposób jak zmodyfikowano algorytm 2 do wyznaczenia x^0 w P2, P3 dla $1|r_j|C_{max}$, z tym że w krokach 3a w miejsce $\max\limits_{1\le j\le n} \Delta_j$ należy dać $\max\limits_{1\le j<l} \Delta_j$. Wielkości T_1, T_2 są równe odpowiednio $\max\limits_{1\le j\le n} \max \left\{\sum_{i=1}^{j}(a_i - u_i) - d_j, 0\right\}$, $\max\limits_{1\le j\le n} \max \left\{\sum_{i=1}^{j} a_i - d_j, 0\right\}$. Algorytm 3 rozwiązuje, podobnie jak algorytm 2, także zadanie P1 po wyrzuceniu warunku $c_j \ge 1$ z kroku 2. Konstrukcja zbiorów P i W jest identyczna.

Przedstawione algorytmy można zastosować bezpośrednio do problemów P1 - P4 dla zagadnienia $1|\prec|T_{max}$ zmieniając sposób wyboru π^0. Permutację π^0 otrzymujemy tak jak dla klasycznego problemu $1|\prec|T_{max}$ z $p_j = a_j$. $1||f_{max}$. Dla problemów P1, P3 i P4 nie są znane aktualnie algorytmy wielomianowe. Przedstawimy algorytm wielomianowy dla P2, oparty na pewnym schemacie obliczeniowym przedstawionym w [9] dla znajdowania ε-aproksymacji zbioru P. Schemat rozwiązania problemu jest następujący.

Niech $D_j = \max\left\{t : f_j(t) \le T\right\}$, $j \in J$; wobec faktu, że f_j są niemalejące, dla istnienia D_j wystarczy założyć lewostronną ciągłość funkcji f_j. Problem P2 możemy teraz sformułować następująco:

$$\min\left\{\sum_{j=1}^{n} c_j x_j : x \in X, \text{ istnieje } \pi \in \Pi \text{ takie, że } C_{\pi(j)} = \sum_{i=1}^{j} a_{\pi(i)} - x_{\pi(i)} \le D_{\pi(j)},\ j \in J\right\}$$

Ponieważ dla każdego $x \in X$, jeżeli $\pi \in \Pi$ oraz $C_{\pi(j)} \le D_{\pi(j)}$, $j \in J$, to $C_{\pi'(j)} \le D_{\pi'(j)}$, $j \in J$, gdzie π' jest permutacją otrzymaną przez uszeregowanie zadań zgodnie z niemalejącymi D_j, zatem $\pi^0 = \pi'$. Optymalny wektor

skróceń x^0 otrzymamy stosując dla naszego zagadnienia algorytm problemu P2 dla $1||L_{max}$ po uprzednim przenumerowaniu zadań tak, aby $\pi^0 = \langle 1,2,\ldots,n\rangle$ oraz przyjęciu $d_j = D_j$, $j \in J$, i $T = 0$. Złożoność obliczeniowa tego algorytmu wynosi $O(n^2)$. Rozważania te można prosto przenieść na problem P2 dla $1|<|f_{max}$, zmieniając tylko sposób wyboru permutacji π^0. Permutację π^0 otrzymujemy tak jak dla klasycznego problemu $1|<|L_{max}$ z $p_j = a_j$ oraz $d_j = D_j$, $j \in J$.

Mimo, że potrafimy rozwiązać P2 dla dowolnego T w wielomianowym czasie, tym razem jednak nie potrafimy wykorzystać faktu, że $\{(T, G(T)) : T_1 \leq T \leq T_2\} = P$ /Twierdzenie 1/ dla rozwiązania P1 w wielomianowym czasie. Wynika to stąd, że nie potrafimy dla dowolnych f_i przedstawić w sposób jednoznaczny zbioru P za pomocą skończonej liczby jego elementów - tak jak to było w problemach rozpatrywanych do tej pory. Można jednak podać algorytm o złożoności $O(n^2 \frac{T_2-T_1}{\varepsilon})$, który znajduje skończony podzbiór zbioru Ω, mający tę własność, że z dowolną dokładnością $\varepsilon > 0$ aproksymuje zbiór P. Sens tej aproksymacji jest następujący. Dla dowolnego $\varepsilon > 0$ podzbiór $P_\varepsilon \subset \Omega$ nazywamy ε-aproksymacją zbioru P jeżeli dla dowolnej pary $(\alpha,\beta) \in P$ istnieje para $(\alpha',\beta') \in P_\varepsilon$ taka, że $\alpha' \leq \alpha \cdot \varepsilon$ i $\beta' \leq \beta + \varepsilon$. W [9] zaproponowano następujący schemat obliczeniowy dla znajdowania zbioru P_ε. Niech T_1 i T_2 będą zdefiniowane tak jak w Twierdzeniu 1. Dla $k = 0,1,\ldots,k_1$ należy znaleźć parę $(\pi^{(k)}, x^{(k)})$ będącą rozwiązaniem problemu P2 dla $T = T^{(k)}$, gdzie $T^{(k+1)} = T^{(k)} - \varepsilon$, $T^{(0)} = T_2$ oraz k_1 spełnia warunek $T^{(k_1)} \leq T_1 + \varepsilon$. Zbiór P_ε tworzą pary

$$P_\varepsilon = \{(T^{(0)}, F_2(x^{(0)})), \ldots, (T^{(k_1)}, F_2(x^{(k_1)}))\}.$$

Wykorzystując tę samą ideę oraz algorytm wielomianowy dla P2 można sformułować algorytm o złożoności $O(n^2 \log \frac{T_2-T_1}{\varepsilon})$ dla P3, który znajduje π^ε i x^ε spełniające $|F_1(\pi^0,x^0) - F_1(\pi^\varepsilon,x^\varepsilon)| < \varepsilon$ dla dowolnego $\varepsilon > 0$ oraz algorytm o złożoności $O(n^2 \frac{T_2-T_1}{\varepsilon})$ znajdujący pary $(\pi^\varepsilon, x^\varepsilon)$ spełniające $|F_1(\pi^0,x^0) + F_2(x^0) - F_1(\pi^\varepsilon,x^\varepsilon) - F_2(x^\varepsilon)| < \varepsilon$ dla dowolnego ε. Konstrukcja tych algorytmów jest oczywista.

Złożoność obliczeniowa decyzyjnej wersji P4 jest sprawą otwartą, natomiast decyzyjna wersja P3 jest problemem wielomianowym - jest ona taka sama jak dla P2.

Można podać algorytmy wielomianowe dla pewnych wąskich podproblemów problemu P3 i P4. Na przykład przy bardzo ostrych założeniach upraszczających: $a_j = a$, $u_j = u$, $c_j = c$, $j \in J$, problem P3 można rozwiązać następująco. Określmy na osi czasu punkty $0 = \tau_0 < \tau_1 < \ldots \tau_n = n\,a - \min\{K/c, n\,u\}$ takie, że $a \geq \Delta_j \geq a - u$, gdzie $\Delta_j = \tau_{j+1} - \tau_j$, $j = 0,\ldots,n-1$, przy czym żadnego z nich nie można przesunąć w lewo nie naruszając powyższych ograniczeń. W punkcie τ_n znajdujemy j_n, dla którego $f_{j_n}(\tau_n) = \min_{1 \leq i \leq n} f_i(\tau_n)$, w punkcie τ_{n-1} znajdujemy j_{n-1}, dla

którego $f_{j_{n-1}}(\tau_{n-1}) = \min_{1\le i\le n, i\neq j_n} f_i(\tau_{n-1})$, itd. Para (π^o, x^o), gdzie $\pi^o = \langle j_1,\ldots,j_n\rangle$, $x^o_{j_i} = a_{j_i} - \Delta_{j_i}$, $i = 1,\ldots,n$, jest rozwiązaniem P3. Algorytm ma złożoność $O(n^2)$ a optymalność pary (π^o, x^o) wynika natychmiast z faktu, że dla każdej permutacji π skrócenia x_j, które minimalizują $F_1(\pi, x)$ przy warunku $F_2(x) \le K$ są takie, że zadanie $\pi(j)$ kończone jest w chwili τ_j, $j = 1,\ldots,n$.

$1|r_j-x_j|C_{max}$. W zagadnieniu tym czasy wykonywania zadań są ustalone, natomiast zmianom podlegają czasy gotowości zadań do realizacji. Pokazano /Nowicki praca nieopublikowana /, że problemy P2 - P4 dla tego zagadnienia są silnie NP-trudne. W dowodzie wykazuje się, że problem $1||\sum w_j T_j$, o którym wiadomo, że jest silnie NP-zupełny, jest pseudowielomianowo transformowalny do decyzyjnych wersji zadań P2, P4.

4. Problemy jednomaszynowe z funkcją celu $\sum f_j$

W tym punkcie przedstawimy tylko kilka wyników.

$1||\sum C_j$. W [11] zaproponowano następujący schemat rozwiązania problemu P4. Dla ustalonej permutacji π $F_1(\pi, x) + F_2(x) = \sum_{j=1}^{n}[(n-j+1)a_{\pi(j)} + (c_{\pi(j)} - (n-j+1))x_{\pi(j)}]$. Funkcja ta przyjmuje wartość minimalną ze względu na $x \in X$ dla $x_{\pi(j)} = u_{\pi(j)}$, gdy $c_{\pi(j)} < n-j+1$ i zero, w przeciwnym wypadku, $j \in J$. Ponieważ wartość $x_{\pi(j)}$ zależy tylko od pozycji j w permutacji π, a nie zależy od zadań, które występują przed lub po $\pi(j)$, problem ten można sprowadzić do problemu przydziału, w którym koszt umieszczenia zadania i na pozycji j wynosi $K_{ij} = a_i(n-j+1) + \min\{0, u_i[c_i - (n-j+1)]\}$. Należy każdemu zadaniu i przydzielić jedną i tylko jedną pozycję j tak, aby łączny koszt był minimalny. Jest to znany problem, rozwiązywany w czasie $O(n^3)$. Nie jest znana złożoność obliczeniowa dla problemów P2, P3.

$1||\sum w_j C_j$. Dla rozwiązania problemu P4 dla tego zagadnienia w [12] podano algorytm typu podziału i ograniczeń. Nie jest znana złożoność obliczeniowa dla tego problemu jak również dla P2 i P3. Można jednak podać proste algorytmy wielomianowe dla przypadku szczególnego: $a_j = a$, $u_j = u$, $c_j = c$, $j \in J$. Dla P1 - P4, π^o jest permutacją otrzymaną przez uporządkowanie zadań zgodnie z nierosnącymi w_j. Załóżmy, nie zmniejszając ogólności rozważań, że $\pi^o = \langle 1,2,\ldots,n\rangle$. Przykładowo algorytm znajdujący x^o dla P2 jest następujący.

Algorytm 4

1. Podstaw $x_j^o = 0$, $j \in J$; $\Delta = s \sum_{j=1}^{n} j w_j - T$; $k = 1$.
2. Jeżeli $\Delta \leq 0$ lub $k = n+1$ to stop /jeżeli $\Delta > 0$ i $k = n+1$, zadanie nie ma rozwiązania/. W przeciwnym wypadku $x_k^o = \min\{\Delta / \sum_{j=k}^{n} w_j, u\}$; $\Delta := \Delta - x_k^o \sum_{j=k}^{n} w_j$; $k := k+1$ i przejdź do 2.

Podobnie proste algorytmy można podać do znajdowania x^o dla P3, P4. Sposób konstrukcji zbiorów P i W w P1 jest trywialny.

5. Problemy przepływowe i gniazdowe

Zajmiemy się najpierw zagadnieniem przepływowym z funkcją celu f_{max}, $P||C_{max}$. Zagadnienie to w klasycznym ujęciu jest silnie NP-zupełne począwszy od $m = 3$. Dla $m = 2$ istnieje znany algorytm wielomianowy Johnsona [1]. W pracach [4],[5] pokazano, że problemy P2 - P4 dla $F2||C_{max}$ są NP-trudne już w takiej sytuacji kiedy $c_{1j} = c$ oraz $u_{2j} = 0$, $j \in J$. W [4] podano szereg algorytmów o złożoności wielomianowej dla różnych szczególnych przypadków problemu P2. Natomiast w [5],[6] zaproponowano dla P4 przy $c_{ij} < 1$ następujące algorytmy heurystyczne o złożoności obliczeniowej $O(n^2)$:

Algorytm G

Wybierz dowolną permutację $\pi^G \in \Pi$ i dla znalezienia x^G rozwiąż zadanie $\min\{F_1(\pi^G, x) + F_2(x) : x \in X\}$.

Algorytm M

Znajdź permutację π^M minimalizującą $F_1(\pi, \bar{x})$, gdzie $\bar{x}_{ij} = (1 - c_{ij})\, u_{ij}$, $i = 1,2$, $j \in J$ i następnie dla znalezienia x^M rozwiąż zadanie $\min\{F_1(\pi^M, x) + F_2(x) : x \in X\}$.

Założenie, że $c_{ij} < 1$ nie ogranicza ogólności algorytmów ponieważ istnieje rozwiązanie optymalne, dla którego $x_{ij}^o = 0$ jeżeli $c_{ij} \geq 1$, a zatem jeżeli $c_{ij} \geq 1$, wówczas możemy przyjąć $u_{ij} = 0$ i jako c_{ij} położyć dowolną liczbę z przedziału $(0,1)$.

Niech dla pewnego konkretnego problemu P4 dla $F2||C_{max}$, E^H oznacza wartość funkcji celu otrzymaną przez algorytm heurystyczny H a E^*, optymalną wartość funkcji celu. Niech $\rho^H = \sup E^H/E^*$ po wszystkich problemach konkretnych. Zachodzi $\rho^G = 2$. Jeżeli $u_{2j} = 0$, $c_{1j} = c$, $j \in J$ to $\rho^M = \frac{3-c}{2}$ dla $c \leq \frac{1}{2}$ i $\rho^M = 1 + c\,(1-c)$ dla $\frac{1}{2} \leq c \leq 1$. Jeżeli $u_{2j} = 0$, $j \in J$, to $\rho^M \leq 1 + \bar{c}\,(1-\underline{c})/[\underline{c}\,(1-\underline{c}) + \bar{c}]$, gdzie $\underline{c} = \min_{j \in J} c_{1j}$, $\bar{c} = \max_{j \in J} c_{1j}$; oraz w ogólnym przypadku $\rho^M \leq \min\{2,\ 3 - 2\min\{\underline{c}, \underline{d}\}\}$, gdzie $\underline{d} = \min_{j \in J} c_{2j}$.

Jedynym rozważanym w literaturze problemem dla zagadnienia gniazdowego $J||C_{max}$ jest problem P3. Algorytm typu podziału i ograniczeń dla tego problemu podany został w [2].

Tab. 1 zawiera zestawienie wyników dotyczących złożoności obliczeniowej algorytmów dla problemów P1 - P4 dla typowych zagadnień szeregowania zadań.

Tablica 1

Złożoność obliczeniowa problemów szeregowania ze zmiennymi czasami wykonywania zadań i liniowymi funkcjami kosztów

Problem / dla zagadnienia	P1	P2	P3	P4
$1\|\|C_{max}$	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(n)$
$1\|\prec\|C_{max}$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O\ n^2$
$1\|r_j,\prec\|C_{max}$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$
$1\|\prec\|L_{max}$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$
$1\|\prec\|T_{max}$	$O(n^2)$, [10]	$O(n^2)$	$O(n^2)$	$O(n^2)$, [11]
$1\|\prec\|f_{max}$	$O(n^2 \frac{T_2-T_1}{\varepsilon})$, [9]	$O(n^2)$, [9]	$O(n^2 \log \frac{T_2-T_1}{\varepsilon})$, dla: $a_j=a, u_j=u, c_j=c$: $O(n^2)$	$O(n^2 \frac{T_2-T_1}{\varepsilon})$
$1\|r_j-x_j\|C_{max}$	Nie łatwiejszy niż P2	Silnie NP-trudny	Silnie NP-trudny	Silnie NP-trudny
$1\|\|\sum C_j$	↑	↑	↑	$O(n^3)$, [11]
$1\|\|\sum w_j C_j$	Dla: $a_j=a$, $u_j=u, c_j=c$: $O(n \log n)$	Dla: $a_j=a$, $u_j=u, c_j=c$: $O(n \log n)$	Dla: $a_j=a$, $u_j=u, c_j=c$: $O(n \log n)$	Algorytm typu podziału i ograniczeń [12] Dla: $a_j=a$ $u_j=u, c_j=c$: $O(n \log n)$
$F2\|\|C_{max}$	Nie łatwiejszy niż P2.	NP-trudny dla $c_{1j}=c$, $u_{2j}=0$, [4]	NP-trudny dla $c_{1j}=c$, $u_{2j}=0$, [4]	NP-trudny już dla $c_{1j}=c$, $u_{2j}=0$, [5] Algorytmy heurystyczne [5],[6]

Problem / dla zagadnienia	P1	P2	P3	P4
$J \mid\mid C_{max}$	Nie łatwiejszy niż P2.	NP-trudny	NP-trudny. Algorytm typu podziału i ograniczeń [2].	NP-trudny

LITERATURA

[1] Graham E.L., Lawler J.K., Lenstra A.H.G., Rinnoy Kan; Optimization and approximation in deterministic sequencing and scheduling: a survey, Ann.Discrete Math. 5 /1979/, 287-326.

[2] Grabowski J., Janiak A.; Job-shop Problem with Resource Constraints in: Large Scale Systems. Theory and Application, Pergamon Press, Oxford 1984, 475-480.

[3] Kelley Jr.J.E.; Critical-path planning and scheduling. Mathematical Basis, Oper.Res. 9 /1961/, 296-320.

[4] Nowicki E.; Minimalizacja kosztu w dwumaszynowym problemie przepływowym ze zmiennymi czasami wykonywania zadań /1986, w tym zeszycie/.

[5] Nowicki E., Zdrzałka S.; Two-Machine Flow Shop Scheduling Problem with Controllable Job Processing Times, Raport ICT Politechniki Wrocławskiej Nr 5/85, Wrocław 1985 /przesłane do redakcji EJOR/.

[6] Nowicki E., Zdrzałka S.; Dwumaszynowy problem przepływowy ze zmiennymi czasami wykonywania zadań, Zeszyty Naukowe AGH, Automatyka z. 39 /1985/, 161-169.

[7] Słowiński R.; Multiobjective Network Scheduling with Efficient Use of Renewable and Nonrenewable Resource, European J. Oper. Res. 7 /1981/, 265-273.

[8] Talbot B.F.; Resource-Constrained Project Scheduling with Time-Resource Tradeoff. The Nonpreemtive Case, Management Sci. 28 /1982/, 1197-1210.

[9] Tuzikov A.V.; A two-criterion scheduling problem allowing for variation in job execution time, Zh. Vychisl. Mat. i Mat. Fiz., 24 /1984/, 1585-1590.

[10] Van Wassenhove L.N., Baker K.R.; A bicriterion approach to time/cost tradeoffs in sequencing, European J. Oper. Res. 11 /1982/, 48-54.

[11] Vickson R.G.; Two single machine sequencing problems involving controllable job processing times, AIIE Trans. 12 /1980/, 258-262.

[12] Vickson R.G.; Choosing the job sequence and processing times to minimize total processing plus flow cost on a single machine, Oper. Res. 28 /1980/, 1155-1167.

Recenzent: Doc.dr h.inż.Jerzy Klamka

Wpłynęło do Redakcji do 1986.04.30

ЗАДАЧИ ТЕОРИИ РАСПИСАНИЙ С УЧЁТОМ ИЗМЕНЕНИЯ ДЛИТЕЛЬНОСТЕЙ ОБСЛУЖИВАНИЯ

Р е з ю м е

В статье рассматривается двухкритериальная задача теории расписаний. Первый критерий состоит в минимизации максимального или суммарного штрафа, который нужно заплатить в моменты окончания обслуживания задач. Второй критерий – минимизация затрат связанных с изменением длительности обслуживания. Представлен систематический обзор существующих результатов.

SCHEDULING PROBLEM WITH VARYING JOB PROCESSING TIMES

S u m m a r y

The paper presents a review of results on sequencing problems in which job processing times are themselves decision variables having their own associated linearly varying costs. Two performance indices are considered: the total job processing cost and the quality index of jobs scheduling. For each classical jobs scheduling model the following four problems are stated: the bicriterion problem, minimizing the cost under given performance index, minimizing the performance index under given cost, and minimizing the cost plus the performance index. The paper establishes general connections between these four problems and presents results concerning the polynomial algorithms, the computational complexity and the worse-case behaviour of heuristics.