ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2000
Seria: AUTOMATYKA z. 129 Nr kol. 1474

Józef GRABOWSKI, Jarosław PEMPERA
Politechnika Wrocławska

NOWE ALGORYTMY TABU SEARCH DLA SZEREGOWANIA ZADAŃ Z KRYTERIUM MINIMALNO KOSZTOWYM

Streszczenie. W niniejszej pracy przedstawia się szereg algorytmów opartych na technice przeszukiwania z zabronieniami (tabu search) dla zagadnienia szeregowania zadań na jednej maszynie z kryterium minimalizacji sumy kosztów zadań wykonywanych nieterminowo. Zaprezentowano szereg nowych technik realizacji zabronienia (tabu) ruchów podczas przeszukiwania rozwiązań sąsiednich. Przedstawiono wyniki obliczeniowe oraz rezultaty porównań algorytmów.

NEW TABU SEARCH ALGORITHMS FOR SINGLE MACHINE TOTAL WEIGHTED TARDINESS PROBLEM

Summary. This paper presents a collection of heuristics for the single machine total weighted tardiness problem. The algorithms considered are based on the insert approach and different sophisticated tabu techniques. The computation results and discussions of the performance of algorithms are presented .

1. Wstęp

Zagadnienie rozważane w niniejszej pracy można przedstawić jako następujące zagadnienie kombinatoryczne. Dany jest zbiór n zadań $J=\{J_1,J_2,...,J_n\}$, które mają być wykonane przy użyciu jednej maszyny. Zakładamy dalej, że maszyna ta może wykonywać w danej chwili tylko jedno zadanie oraz że wykonywanie zadań nie może być przerywane. Dla każdego zadania J_i, (i=1,2,...,n) określony jest czas wykonywania p_i oraz żądany termin zakończenia realizacji d_i. Przekroczenie terminu d_i związane jest z poniesieniem dodatkowych kosztów $f_i(C_i)=w_iT_i$, które zależą od „wagi" zadania w_i oraz od długości spóźnienia $T_i=\max(0,C_i-d_i)$, gdzie C_i jest terminem zakończenia realizacji zadania J_i. Zagadnienie optymalizacji polega na określeniu takiej kolejności wykonywania zadań na maszynie, aby suma ważonych kosztów spóźnień $F=\sum_{i=1}^{n} f_i(C_i)=\sum_{i=1}^{n} w_i T_i$ była minimalna. Niech $\pi=(\pi(1),\pi(2),\pi(3),...,\pi(n))$ będzie permutacją wykonywania zadań, wówczas całkowity koszt

ich realizacji wynosi: $F(\pi)=\sum_{i=1}^{n} f_{\pi(i)}(C_{\pi(i)})$, gdzie $C_{\pi(i)}=\sum_{k=1}^{i} p_{\pi(k)}$. Należy znaleźć permutację π^* taką, że $F(\pi^*)=\min_{\pi} F(\pi)$.

Zagadnienie to (NP-trudne) jest jednym z najtrudniejszych zagadnień kolejnościowych. Zastosowanie metod dokładnych ze względu na czas obliczeń staje się niemożliwe już dla zagadnień o niewielkich rozmiarach [1],[7],[9],[10]. Praktyczne znaczenie powyższego problemu w komputerowo zintegrowanym wytwarzaniu (CIM) oraz systemach wytwarzania na żądanie (JIT) [12] powoduje, iż do ich rozwiązania stosuje się zwykle szybkie algorytmy typu konstrukcyjnego oparte na statycznych regułach priorytetowych (SWPT) Shortest Weighted Proccesing Time [11], (EDD) Earliest Due Date [2] oraz na dynamicznych regułach priorytetowych algorytmów (COVERT) Cost Over Time oraz (AU) Apparent Urgency [8]. Jednakże rozwiązania otrzymywane tymi algorytmami często znacznie odbiegają od rozwiązań optymalnych, co potwierdzają wyniki obliczeniowe zawarte w [4][8]. Dlatego też aktualnym kierunkiem badań jest konstruowanie algorytmów typu popraw, które pozwalają uzyskać znacznie lepsze rozwiązania w czasie akceptowalnym przez praktyków. W pracy [5],[8] zostały zaproponowane algorytmy oparte na technice symulowanego wyżarzania, natomiast w pracach [3],[6] przedstawiono algorytmy oparte na technice tabu search, które mogą być użyte do rozwiązania naszego problemu. W niniejszej pracy przedstawiono szereg nowych szybkich algorytmów opartych na tego rodzaju technice, w których zaproponowano różne sposoby realizacji techniki tabu (zabronień). W efekcie uzyskano nowe oryginalne algorytmy, dające bardzo dobre rezultaty potwierdzone eksperymentami obliczeniowymi.

2. Algorytm tabu search (TS)

Ogólnie technika tabu search, podobnie jak wszystkie techniki popraw, rozpoczyna obliczenia od pewnej początkowej permutacji bazowej i przeszukuje jej rozwiązania „sąsiednie" w celu znalezienia permutacji o najmniejszym koszcie. Następnie znaleziona permutacja staje się nową permutacją bazową i proces przeszukiwań zostaje wznowiony. Permutacje sąsiednie otrzymywane są jako efekt wykonania „ruchu" wewnątrz permutacji bazowej, polegający na zmianie pozycji pewnych zadań. W celu zapobieżenia powrotu do rozwiązań już sprawdzonych wprowadza się mechanizm tabu. Zwykle realizowany jest on za pomocą listy, na której zapamiętywane są pewne atrybuty rozwiązań bazowych i/lub wykonywanych ruchów. Elementy listy tabu określają, które z ruchów nie mogą być aktualnie wykonywane. Ruchy te wyznaczają zabronione permutacje sąsiednie dla aktualnie

rozważanego rozwiązania bazowego. Ze względu na ograniczoną wielkość listy tabu „najstarsze" elementy listy są usuwane, natomiast nowe są dodawane. Proces poszukiwań kończy się wtedy, gdy zostanie spełniony warunek zatrzymania. Najczęściej bywa nim przekroczenie limitu czasowego, limitu iteracji, limitu iteracji bez poprawy, osiągnięcie zadowalającej wartości funkcji celu, znalezienie rozwiązania dokładnego. W każdym algorytmie opartym na proponowanej technice należy zdefiniować podstawowe komponenty takie, jak: ruch, formę realizującą mechanizm tabu, status zabronienia ruchów i/lub rozwiązań sąsiednich, strategie poszukiwań oraz kryterium zatrzymania.

2.1. Ruchy i rozwiązania sąsiednie

Niech $v=(a,b)$ będzie parą liczb określających pozycje w permutacji π, $a,b\in\{1,...,n\}$, $a\neq b$. Para ta definiuje ruch w permutacji π, który oznacza usunięcie zadania $\pi(a)$ z pozycji a i wstawienie go na pozycję b w π. Tego typu ruch generuje następujące rozwiązania sąsiednie π_v dla permutacji π

$$\pi_v=\begin{cases}(\pi(1),...,\pi(a-1),\pi(a+1),...,\pi(b),\pi(a),\pi(b+1),...,\pi(n)) & \text{dla } a<b,\\ (\pi(1),...,\pi(b-1),\pi(a),\pi(b),...,\pi(a-1),\pi(a+1),...,\pi(n)) & \text{dla } a>b.\end{cases}$$

Niech U oznacza zbiór wszystkich możliwych ruchów. Sąsiedztwo (zbiór sąsiadów) permutacji π generowane przez ruchy ze zbioru U będziemy oznaczać przez $N(U,\pi)=\{\pi_v: v\in U\}$. Największe sąsiedztwo generowane jest przez zbiór ruchów $X=\{(a,b): b\notin\{a,a-1\}, a,b\in\{1,2,...,n\}\}$. Warunek $b\notin\{a,a-1\}$ został wprowadzony w celu uniknięcia powtarzania generowanych rozwiązań. Zbiór $N(X,\pi)$ posiada $(n-1)^2$ elementów i jest wystarczający do spełnienia zasady styczności, tj. dla pewnej permutacji $\pi^{(1)}$ istnieje sekwencja permutacji $\pi^{(1)},\pi^{(2)},...,\pi^{(r)}$ taka, że $\pi^{(r)}$ jest permutacją optymalną oraz $\pi^{(i+1)}\in N(X,\pi^{(i)})$ dla $i=1,2,...,r-1$.

Duża liczność sąsiedztwa $N(X,\pi)$ wymaga znacznego nakładu obliczeniowego, w naszym przypadku $O(n^3)$, jeśli przyjmiemy, że dla całego sąsiedztwa wartość funkcji kryterialnej liczona jest bez korzystania z wyników pośrednich. Dla rozważanego problemu złożoność obliczeniową można zredukować do $O(n^2)$ generując rozwiązania sąsiednie w takiej kolejności, aby można było skorzystać z wyników dla wcześniej wygenerowanych rozwiązań [3].

2.2. Mechanizm tabu i status zabronienia dla ruchów

Tradycyjnie (TRA) w zagadnieniach o charakterze permutacyjnym mechanizm tabu jest realizowany przy użyciu cyklicznej listy T o długości L. W przypadku wykonania ruchu $v=(a,b)$ w π, podobnie jak w [6], do listy dodawany jest element $(\pi(a),\pi(a+1))$ dla $a<b$ i $(\pi(a-1),\pi(a))$ - w przeciwnym przypadku. Przed dodaniem nowego ruchu najstarszy element z listy jest usuwany. Ruch $v=(a,b)$ w π jest *zabroniony*, jeżeli istnieje w liście T element $(\pi(j),\pi(a)), j=a+1,...,b$ dla $a<b$, lub $(\pi(a),\pi(j)), j=b,...,a-1$, dla $a>b$.

W niniejszej pracy proponujemy szereg nowych metod realizacji mechanizmu tabu. Wspólną zasadniczą cechą tych metod jest przydział każdemu zadaniu J_i *Stopnia Tabu* ST_i. Wartościami ST_i są liczby całkowite z określonego przedziału $\langle 1,m\rangle$. Różne zadania otrzymują różne wartości ST_i, zależnie od pewnych cech charakteryzujących zadanie. W przypadku wykonania ruchu $v=(a,b)$, (polegającego na usunięciu zadania $J_{\pi(a)}$ z pozycji a permutacji π i wstawieniu na pozycję b w tej permutacji), stopień tabu $ST_{\pi(a)}$ zadania $J_{\pi(a)}$ ulega zmniejszeniu o jedną jednostkę. Ruch $v=(a,b)$ jest *zabroniony* jeżeli $ST_{\pi(a)}=0$. Sprawdzenie tego zabronienia posiada złożoność obliczeniową $O(1)$. Stopień tabu $ST_i=0$ zadania J_i jest *regenerowany* do wartości początkowej, jeżeli algorytm wykonał *li* iteracji po ostatnim wyzerowaniu tego stopnia. W niniejszych algorytmach zaprogramowaliśmy następujące metody wyznaczania stopni ST_i.

(A) Dla wszystkich zadań $J_i \in J$, ST_i są jednakowe i równe : 1-oznaczenie A(1), 2-oznaczenie A(2),..., 10-oznaczenie A(10), Symbol TSA(7) oznacza, że w algorytmie wszystkie ST_i są jednakowe i równe 7.

(B) Stopnie tabu ST_i są przydzielane zadaniom zgodnie z niżej opisaną regułą R1 (oznaczenie algorytmu TSR1).

(C) Jak wyżej, lecz zgodnie z regułą R2 (oznaczenie TSR2).

Niech $H(x, y, z)$ będzie pewną określoną funkcją, nazywaną *funkcją charakterystyczną* zmiennych niezależnych x, y oraz z. Dla wartości zmiennych $x=w_i$, $y=p_i$ oraz $z=d_i$ wartość funkcji charakterystycznej $H_i=H(w_i,p_i,d_i)$ będziemy nazywać *wartością charakterystyczną* zadania J_i, $i=1,2,...,n$. Niech J_e oraz J_f będą zadaniami posiadającymi odpowiednio najmniejszą oraz największą wartość charakterystyczną, czyli : $H^{min}=H_e=\min_i H_i$ oraz $H^{max}=H_f=\max_i H_i$.

REGUŁA R1

Zadanie o najmniejszej wartości charakterystycznej, czyli J_e otrzymuje najniższy stopień tabu $ST_e=1$, zadanie o największej wartości charakterystycznej J_f otrzymuje najwyższy stopień

tabu $ST_f=m$. Natomiast pozostałe zadania J_j, $j=1,2,...,n$, $j\neq e$, $j\neq f$, dla których $H^{min} \le H_j < H^{max}$ otrzymują rosnące (całkowitoliczbowe) wartości ST_t z przedziału $\langle 1,m-1\rangle$ wraz ze wzrostem ich wartości charakterystycznych H_j. Czyli, jeżeli dla pewnego k, $1\le k<m$, mamy: $H^{min} + (k-1)H^{śr} \le H_j < H^{max}+kH^{śr}$, gdzie : $H^{śr} = \frac{H^{max} - H^{min}}{m-1}$,

to zadanie J_j otrzymuje $ST_j=k$.

REGUŁA R2

Reguła R2 jest regułą przeciwną do REGUŁY R1. Teraz zadanie o najmniejszej wartości charakterystycznej J_e otrzymuje największy stopień tabu $ST_e=m$, natomiast zadanie o największej wartości charakterystycznej J_f otrzymuje najmniejszy stopień tabu $ST_f=1$. Pozostałe zadania J_j, $j=1,2,...,n$, $j\neq e$, $j\neq f$, dla których $H^{min} \le H_j < H^{max}$, otrzymują malejące (całkowitoliczbowe) wartości ST_t z przedziału $\langle 2,m\rangle$ wraz ze wzrostem ich wartości charakterystycznych H_j. Zatem, jeżeli teraz dla pewnego k, $1<k\le m$, mamy

$$H^{max} - (k-1)H^{śr} \ge H_j > H^{max} - kH^{śr},$$

to zadanie J_j otrzymuje $ST_j=k$. W regułach R1 oraz R2 przyjęliśmy $m=10$.

2.3. Strategia poszukiwań, warunki zatrzymania

Proces poszukiwań został podzielony na kilka etapów. Na wstępie zbiór ruchów X dzielony jest na n podzbiorów $X=\{X_1, X_2,...,X_n\}$, gdzie X_a oznacza zbiór ruchów polegający na przesunięciu zadania z pozycji a w π. Dla każdego podzbioru wyznacza się ruch r_a zwany *reprezentantem* taki, że $F(\pi_{r_a})=\min_{v\in X_a} F(\pi_v)$. Spośród wszystkich reprezentantów nie zabronionych oraz zabronionych ale profitujących, tzn. takich, że $F(\pi_v) < F(\pi^*)$ (gdzie π^* jest aktualnie najlepszym rozwiązaniem), wybiera się najlepszy ruch v, tzn. taki, że $F(\pi_v) = \min_{1\le a\le n} F(\pi_{r_a})$. Algorytm kończy działanie po przekroczeniu limitu iteracji *limititer*.

3. Algorytm TS

W przedstawionym dalej algorytmie wartości oznaczone górnym indeksem w postaci (*) oznaczają dotychczas znalezione najlepsze wartości, zerem (°) - wartości początkowe, natomiast bez indeksu oznaczają wartości bieżące. Algorytm startuje od rozwiązania początkowego π° otrzymanego algorytmem AU, kończy działanie po wykonaniu *limititer* iteracji.

INICJACJA

$it \leftarrow 0$, $T \leftarrow \varnothing$, $\pi^* \leftarrow \pi$, $\pi \leftarrow \pi$, $F^* = F(\pi^*)$.

PRZESZUKIWANIE

$it \leftarrow it+1$, wyznacz wszystkich reprezentantów r_a, $a=1,...,n$.

Regeneruj stopnie tabu (dotyczy algorytmów TSA(•), TSR1, TSR2).

WYBÓR

Spośród wszystkich reprezentantów dozwolonych i zabronionych ale profitujących, tzn. takich, że $F(\pi_{r_a}) < F^*$, wybierz najlepszy ruch v taki, że $F(\pi_v)=\min_{1 \le a \le n} F(\pi_{r_a})$.

Jeżeli v nie istnieje, wtedy usuń najstarszy element listy T i idź do WYBÓR (dot. TSTRA).

WYKONAJ_RUCH

Odpowiednio zmodyfikuj listę tabu (lub stopnie tabu).

$\pi \leftarrow \pi_v$, jeżeli $F(\pi_v) < F^*$ to $\pi^* \leftarrow \pi_v$, $F^* = F(\pi^*)$.

KRYTERIUM_ZATRZYMANIA

Jeżeli $it \ge limititer$, wtedy STOP.

4. Wyniki badań testujących

Algorytmy tabu search zaproponowane w niniejszej pracy zostały zakodowane w języku C++ i były testowane na komputerze IBM RISC System/6000, 200 MHz. Badania testowe (podobnie jak w [3] oraz [8]) przeprowadzono na przykładach, dla których wartości p_i, w_i oraz d_i zostały wygenerowane według rozkładu równomiernego, przy czym wartości p_i wygenerowano ze zbioru {1,2,...,100}, w_i ze zbioru {1,...,10}, natomiast d_i ze zbioru $[P(1-TF-RDD/2), P(1-TF+RDD/2)]$, gdzie $P=\sum_{1 \le i \le n} p_i$, RDD i TF są parametrami określającymi zakres wartości terminu zakończenia realizacji oraz średni rozrzut wartości tego terminu. Przeprowadzono badania dla wartości RDD i TF ze zbioru {0.2,0.4,0.8,1.0} oraz n={20,40,50,100}. Dla każdej kombinacji wartości n, RDD i TF wygenerowano 100 przykładów testujących, otrzymując w sumie 10,000 problemów.

We wszystkich wersjach algorytmu TS przyjęto $limititer$=1000, natomiast w przypadku TRA L=7, a w pozostałych li=24. Rozwiązanie początkowe dla wszystkich algorytmów otrzymano przy użyciu algorytmu AU [8] z k=0.5,0.9,2.0,2.0,2.0 dla TF= 0.2,0.4,0.6,0.8,1.0. Dla każdego przykładu wyznaczono następujące wielkości :

C^H -wartość funkcji celu otrzymana algorytmem H, H={AU,TSTRA,TSA(•),TSR1,TSR2}

$C^m = \min_H(C^H)$ - najmniejsza wartość funkcji celu otrzymana jednym z algorytmów.

Na podstawie tych wielkości obliczono :

r^H = $100\%(C^H - C^m)/C^m$ - względny błąd rozwiązania uzyskanego algorytmem H w stosunku do najlepszego rozwiązania otrzymanego jednym z algorytmów.

p^H - procent przykładów, dla których $C^H = C^m$.

r_{av}^H - średni względny błąd dla algorytmu H.

CJI – czas jednej iteracji.

Na podstawie otrzymanych wyników przedstawionych w tabeli 1 można stwierdzić, że rozwiązania otrzymane jednym z najlepszych algorytmów konstrukcyjnych AU są znacznie poprawiane przez algorytmy TSTRA, TSA(•), TSR1, TSR2. Ponadto, algorytmy TSA(•), TSR1 oraz TSR2 wykorzystując stopnie tabu ST_i są wyraźnie lepsze od algorytmów TSTRA z tradycyjną listą tabu. Najlepsze rezultaty uzyskano przy użyciu algorytmu TSR1 ze stopniami tabu ustalonymi na podstawie wartości charakterystycznych $H_i=d_i/p_i$. Dla n=40 średni błąd względny wynosi 0.01, natomiast dla problemów n=100 wartość błędu wzrosła do 0.09. Ponadto, przy użyciu tego algorytmu dla n=40 uzyskano 99,2 % rozwiązań najlepszych, natomiast dla n=100, uzyskano 84,8% tego rodzaju rozwiązań. Najgorsze rezultaty otrzymano dla algorytmu TSA(10) z jednakowymi stopniami tabu ST_i=10 dla wszystkich zadań. Dla n=40 oraz n=100 otrzymano największe wartości błędu oraz najmniejszy procent najlepszych rozwiązań. Natomiast w miarę obniżania się wartości stopnia tabu ST_i algorytmy TSA(•) uzyskiwały coraz lepsze rezultaty. Dla ST_i=1 wyniki algorytmów TSA(1) były porównywalne z wynikami najlepszego algorytmu TSR1 z funkcją $H_i=d_i/p_i$.

Analizując czasy przebiegu algorytmów należy stwierdzić, że algorytmy realizujące zabronienia ruchów przy użyciu stopnia tabu ST_i są średnio około 20% dla n=40 oraz około 40% dla n=100 szybsze w porównaniu z algorytmami tradycyjnymi. Wynika to oczywiście z faktu, że złożoność obliczeniowa sprawdzania zabronienia ruchów w algorytmach tradycyjnych jest większa niż w algorytmach ze stopniami tabu ST_i.

Reasumując, wydaje się, że algorytm TSR1 z $H_i=d_i/p_i$ jest algorytmem najlepszym z punktu widzenia uzyskiwanych wartości funkcji celu i czasów obliczeń.

Wszystkie algorytmy były testowane dla 1000 iteracji. Obserwując przebieg obliczeń algorytmów można było zauważyć, że zdecydowana poprawa wartości funkcji celu (ok. 99%) była uzyskiwana w pierwszych 100 iteracjach.

Tablica 1

		n=40			n=100		
Algorytm	$H(\bullet)$	r_{π}^{H}	P^{H}	CJI [ms]	r_{π}^{H}	p^{H}	CJI [ms]
AU		13.67	21.6	0.0	21.87	15.2	-
TS TRA		0.58	88.8	1.8	6.92	84.0	14.8
TSA(1)		0.01	96.8	1.8	0.09	83.2	9.7
TSA(2)		0.12	94.4	1.5	0.28	85.6	9.3
TSA(3)		0.17	92.8	1.5	0.30	84.8	9.4
TSA(4)		0.17	92.0	1.5	0.31	84.0	9.4
TSA(5)		0.35	90.4	1.5	0.31	84.0	9.3
TSA(6)		0.27	91.2	1.5	0.31	84.0	9.4
TSA(7)		0.35	90.4	1.5	0.31	84.0	9.4
TSA(8)		0.35	90.4	1.5	0.31	84.0	9.4
TSA(9)		0.27	91.2	1.5	0.31	84.0	9.4
TSA(10)		0.51	88.8	1.5	0.32	83.2	9.4
TSR1	w_i	0.27	91.2	1.5	0.19	85.6	9.4
TSR1	p_i	0.12	92.8	1.5	0.31	84.8	9.4
TSR1	d_i	0.12	92.8	1.5	0.30	84.8	9.4
TSR1	p_i/w_i	0.06	97.6	1.5	0.29	84.8	9.4
TSR1	p_i/d_i	0.07	96.8	1.5	0.32	83.2	9.4
TSR1	w_i/p_i	0.06	96.8	1.5	0.09	83.2	9.4
TSR1	w_i/d_i	0.30	93.6	1.5	0.05	87.2	9.4
TSR1	d_i/p_i	0.01	99.2	1.5	0.09	84.8	9.4
TSR1	d_i/w_i	0.06	97.6	1.6	0.27	85.6	9.4
TSR1	$w_i p_i$	0.07	93.6	1.6	0.16	84.0	9.4
TSR1	$w_i d_i$	0.30	91.2	1.5	0.19	87.2	9.4
TSR1	$p_i d_i$	0.15	94.4	1.5	0.27	86.4	9.4
TSR2	w_i	0.17	92.0	1.5	0.31	83.2	9.3
TSR2	p_i	0.12	92.0	1.5	0.18	86.4	9.4
TSR2	d_i	0.22	92.8	1.5	0.21	85.6	9.3
TSR2	p_i/w_i	0.30	91.2	1.5	0.32	83.2	9.4
TSR2	p_i/d_i	0.12	92.8	1.5	0.28	85.6	9.4
TSR2	w_i/p_i	0.30	91.2	1.5	0.32	83.2	9.3
TSR2	w_i/d_i	0.17	92.0	1.5	0.31	84.0	9.4
TSR2	d_i/p_i	0.35	90.4	1.5	0.31	84.0	9.4
TSR2	d_i/w_i	0.22	92.0	1.5	0.32	83.2	9.4
TSR2	$w_i p_i$	0.27	92.0	1.5	0.31	83.2	9.4
TSR2	$w_i d_i$	0.12	93.6	1.5	0.31	83.2	9.4
TSR2	$p_i d_i$	0.17	92.8	1.5	0.21	84.8	9.4

LITERATURA

1. Adrabiński A., Grabowski J., Wodecki M.: Algorytm rozwiązania zagadnienia kolejnościowego postaci $n|1|\sum w_i T_i$. Archiwum Automatyki i Telemechaniki, 33, 4, 1988, 623-636.
2. Baker K.R.: Introduction to sequencing and scheduling. Wiley, New York, 1974.

3. Grabowski J. Pempera J.: Some local search algorithms for single machine weighted tardiness problem. Proc. of the Fifteenth European Meeting on Cybernetics and Systems Research (EMCSR 2000), Vienna 2000, 145-154.
4. Grabowski J., Pempera J.: Algorytmy szeregowania zadań z kryterium minimalno-kosztowym. Zeszyty Naukowe Politechniki Śląskiej, s.Automatyka, z.125, Gliwice 1998, 38-45.
5. Matsuo H., Suh C.J., Sullivan R.S.: A Controled Search Simulated Annealing Method for the single Machine Weighted Problem. Annals of Operations Research, 21, 1989, 85-108.
6. Nowicki E., Zdrzałka S.: Single Machine Scheduling with Major and Minor Setup Times. Tabu search approach. Journal the Operational Research Society, 47, 1996, 1054-1064.
7. Potts C.N., Van Wassenhove L.N.: A Branch and Bound Algorithm for Total Weighted Tardiness Problem, Operations Research, 33, 1985, 363-377.
8. Potts C.N., Van Wassenhove L.N.: Single Machine Tardiness Sequencing Heuristics. IIE Transactions, 23, 1991, 346-354.
9. Rinnooy Kan A.H.G., Lageweg B.J., Lenstra J.K.: Minimizing Toatal Cost One-Machine Scheduling. Operations Research, 26, 1975, 908-927.
10. Shrage L., Baker K.R. : Dynamic Programming solution of Sequencing Problems with Precedence Constrains. Operations Research, 26, 1978, 444-449.
11. Smith W.E.: Various Optimizers for Single-Stage Production. Naval Research Logistics Quartely, 3, 1956, 59-66.
12. Smutnicki C.: Optimization and Control in Just-Time Manufacturing System. Seria Monografie, Oficyna Wydawnicza PW, Wrocław 1997.

Recenzent: Prof.dr hab.inż. T.Sawik

Abstract

This paper deals with heuristics for single machine total weighted tardiness problem. Consider n jobs $J_1,J_2,...,J_n$ to be processed without interruption on a single machine that can handle only one job at a time. Job J_i requires a processing time p_i, has a weight w_i, and has a due date d_i. For a given sequence of the jobs the completion time C_i and the tardiness $T_i=\max(0,C_i-d_i)$ of job J_i can be computed. The objective is to find a processing order of the jobs that minimizes $\sum_{i=1}^{n} w_i T_i$. The problem belongs to the class of NP-hard problems what justifies searching for heuristics algorithms. In the paper, we propose several approximation algorithms based on the insert approach and different sophisticated tabu techniques. Finally, the computation results and discussion of the performance of algorithms are presented.