AUTOMATYKA • 2009 • Tom 13 • Zeszyt 2

Krzysztof M. Ocetkiewicz*

Algorytm branch-and-bound dla pewnego problemu szeregowania zadań uwarunkowanych czasowo

1. Wprowadzenie

Szeregowanie zadań uwarunkowanych czasowo jest uogólnieniem klasycznego szeregowania zadań. Czas potrzebny do wykonania zadania nie jest tu stały, lecz zależy od chwili, w której procesor rozpocznie jego wykonywanie. Ponieważ sytuacja, w której funkcja opisująca czas wykonywania jest dowolna, jest trudna do analizy, najczęściej rozważa się przypadki, w których funkcja ta jest liniowa. Status wielu takich problemów został już określony [1] (niestety większość z nich jest NP-trudna), ciągle jednak pozostaje kilka problemów, których status nie jest jeszcze ustalony. Jednym z tych problemów jest jednoprocesorowy problem szeregowania zadań, których czasy wykonywania są liniowo zależne od momentu rozpoczęcia wykonywania, a podstawowy czas wykonywania jest wspólny dla wszystkich zadań. Kryterium optymalizacji jest tu całkowity czas wykonywania, tj. suma, po wszystkich zadaniach czasów zakończeń ich wykonywania. Istnieje algorytm, który w wielomianowym czasie rozwiązuje, w sposób optymalny, niektóre instancje tego problemu [2], jednak instancje te muszą spełnić bardzo restrykcyjne warunki. Skonstruowano również pewną liczbę wielomianowych algorytmów przybliżonych ([3–6]), lecz żaden z nich nie daje gwarancji, że znalezione rozwiązanie będzie leżeć choć w pobliżu rozwiązania optymalnego. W pracy tej pokazany zostanie algorytm typu branch-and-bound, pozwalający znacznie przyspieszyć poszukiwanie optymalnego rozwiązania przedstawionego problemu.

2. Sformułowanie problemu

Dany jest jeden procesor oraz zestaw n nieprzerywalnych zadań $J_1, ..., J_n$. Czas wykonywania zadania J_i wyraża się wzorem $p_i = a + b_i s_i$, gdzie s_i to moment rozpoczęcia wykonywania zadania J_i, $a > 0$ to podstawowy czas wykonywania zadania (wspólny dla wszyst-

* Katedra Algorytmów i Modelowania Systemów, Politechnika Gdańska

kich zadań), zaś $b_i \geq 0$ to współczynnik wydłużania *i*-tego zadania. Należy znaleźć taki harmonogram π (permutację zbioru zadań), że łączny czas wykonywania ($\Sigma_{j=1,..n}\ C_{\pi\ (j)}$, gdzie $C_i = s_i + p_i$) jest najmniejszy z możliwych. Czas rozpoczęcia wykonywania zadań jest równy zero ($s_{\pi(1)} = 0$). Funkcja celu ma postać [3]:

$$TC = \sum_{i=1}^{n} C_{\pi(i)} = n + a \sum_{i=2}^{n} \sum_{j=2}^{i} \prod_{k=j}^{i} \left(1 + b_{\pi(k)}\right) \tag{1}$$

Ponieważ podstawowy czas wykonywania nie ma wpływu na kolejność zadań w harmonogramie (jego zmiana nie spowoduje, że optymalny harmonogram przestanie być takim), a jedynie „skaluje" wartość funkcji celu, zazwyczaj przyjmuje się, że $a = 1$. Złożoność tego problemu nie została jeszcze ustalona. Znane są jednak pewne właściwości optymalnego harmonogramu. Jedną z nich jest V-kształtność [3]:

(i) zadanie z największym współczynnikiem wydłużania powinno być wykonane jako pierwsze,
(ii) zadanie z najmniejszym współczynnikiem wydłużania nie powinno być wykonywane ani drugie, ani ostatnie (gdy $n > 3$),
(iii) zadania umieszczone w harmonogramie przed zadaniem z pkt. (ii) powinny być wykonywane w kolejności nierosnących współczynników wydłużania,
(iv) zadania umieszczone w harmonogramie za zadaniem z pkt. (ii) powinny być wykonywane w kolejności niemalejących współczynników wydłużania.

Kolejną przydatną cechą dowolnego harmonogramu w tym problemie jest jego „odwracalność". Kolejność wszystkich, poza pierwszym, zadań w harmonogramie można odwrócić (tzn. z harmonogramu $J_1, J_2, ..., J_n$ otrzymujemy harmonogram $J_1, J_n, ..., J_2$) nie zmieniając przy tym wartości funkcji celu [3].

2.1. Reprezentacja harmonogramu

Dzięki powyższym cechom rozwiązania, zamiast przechowywać rozwiązanie problemu w postaci permutacji zbioru zadań, możemy przedstawić je w postaci ciągu binarnego. Dla permutacji π długości n tworzymy ciąg binarny, w którym bit j_i odpowiada zdaniu $J_{i+1:n}$, gdzie $J_{i+1:n}$ jest zadaniem, którego współczynnik wydłużania znalazłby się na *i*-tej pozycji po uporządkowaniu wg niemalejących współczynników wydłużania. j_i jest równe 1, gdy odpowiadające mu zadanie powinno być wykonane po zadaniu $J_{1:n}$, zaś 0 w przeciwnym wypadku. Reprezentacja taka nie pozwala przedstawić wszystkich możliwych harmonogramów, a jedynie wszystkie harmonogramy V-kształtne – właśnie te, które nas interesują (tylko harmonogram V-kształtny może być optymalny). Co więcej, bit j_{n-1} powinien mieć zawsze wartość 0 (pkt. (i)) zaś wartość bitu j_0 jest nieistotna (można więc przyjąć $j_0 = 0$). Z odwracalności harmonogramu wynika także, że istnieją co najmniej dwa optymalne harmonogramy, z których w jednym $j_{n-2} = 0$, w drugim zaś $j_{n-2} = 1$. Z rozważań tych wynika, że algorytm pełnego przeszukiwania ma do przejrzenia przestrzeń o 2^{n-3} elementach.

3. Algorytm branch-and-bound

W najogólniejszej wersji, algorytmy typu branch-and-bound ograniczają rozmiar przestrzeni poszukiwań poprzez odrzucenie tych jej fragmentów, co do których możemy z całą pewnością stwierdzić, że nie zawierają rozwiązania optymalnego. Potrzebujemy do tego górnego oszacowania wartości optymalnego rozwiązania (np. wartości najlepszego dotychczas znalezionego rozwiązania) oraz metody oszacowania z dołu wartości całego rozwiązania na podstawie jego części. Dla uproszczenia opisu, od tej chwili przez J_i będziemy oznaczać zadanie na i-tej pozycji w rozważanym harmonogramie (analogicznie b_i to współczynnik wydłużania tego zadania). Rozważmy zestaw zadań J_p, ..., J_q i niech $s_p = T$. Momenty zakończenia wykonywania tych zadań C_p, ..., C_q możemy obliczyć następująco:

$$\begin{aligned} C_i &= (T-1)\prod_{k=p}^{q}\left(1+b_{\pi(k)}\right)+\sum_{j=p}^{i}\prod_{k=j}^{i}\left(1+b_{\pi(k)}\right) \text{ dla } i=p,...,q \\ \sum_{i=q}^{p} C_i &= \left((q-p+1)+\right)(T-1)\sum_{i=p}^{q}\prod_{k=i}^{q}\left(1+b_{\pi(k)}\right)+\sum_{i=p}^{q}\sum_{j=p}^{i}\prod_{k=j}^{i}\left(1+b_{\pi(k)}\right) \end{aligned} \tag{2}$$

Wykorzystując to równanie i wyróżniając dwa zadania J_p i J_q ($p < q$), możemy podzielić harmonogram na trzy części: pierwsza część to zadania J_1, ..., J_p, druga obejmuje zadania J_{p+1}, ..., J_{q-1}, ostatnia zaś zadania J_q, ..., J_n. Wkład poszczególnych części do funkcji celu wyraża się równaniami:

$$TC_1 = \sum_{i=2}^{p}\sum_{j=2}^{i}\prod_{k=j}^{i}\left(1+b_{\pi(k)}\right)+p,$$

$$TC_2 = \left(\sum_{j=2}^{p}\prod_{k=j}^{p}\left(1+b_{\pi(k)}\right)\right)\left(\sum_{j=p+1}^{q-1}\prod_{k=p+1}^{j}\left(1+b_{\pi(k)}\right)\right)+$$
$$+\sum_{i=p+1}^{q-1}\sum_{j=p+1}^{i}\prod_{k=j}^{i}\left(1+b_{\pi(k)}\right)+(q-p-1),$$

$$TC_3 = \left(\sum_{j=2}^{p}\prod_{k=j}^{p}\left(1+b_{\pi(k)}\right)\right)\left(\prod_{k=p+1}^{q-1}\left(1+b_{\pi(k)}\right)\right)\left(\sum_{j=q}^{n}\prod_{k=q}^{j}\left(1+b_{\pi(k)}\right)\right)+$$
$$+\left(\sum_{j=p+1}^{q-1}\prod_{k=j}^{q-1}\left(1+b_{\pi(k)}\right)\right)\left(\sum_{j=q}^{n}\prod_{k=q}^{i}\left(1+b_{\pi(k)}\right)\right)+\sum_{i=q}^{n}\sum_{j=q}^{i}\prod_{k=j}^{i}\left(1+b_{\pi(k)}\right)+(n-p-1).$$

Zdefiniujmy następujące symbole:

$$S_1 = \sum_{i=2}^{p}\sum_{j=2}^{i}\prod_{k=j}^{i}\left(1+b_{\pi(k)}\right), \qquad S_2 = \sum_{i=p+1}^{q-1}\sum_{j=p+1}^{i}\prod_{k=j}^{i}\left(1+b_{\pi(k)}\right), \quad S_3 = \sum_{i=q}^{n}\sum_{j=q}^{i}\prod_{k=j}^{i}\left(1+b_{\pi(k)}\right),$$

$$X_{p,q} = \sum_{j=p+1}^{q-1}\prod_{k=p+1}^{j}\left(1+b_{\pi(k)}\right), \quad Y_{p,q} = \sum_{j=p+1}^{q-1}\prod_{k=j}^{q-1}\left(1+b_{\pi(k)}\right), \qquad Z_{p,q} = \prod_{k=p+1}^{q-1}\left(1+b_{\pi(k)}\right),$$

$$P_p = \sum_{j=2}^{p}\prod_{k=j}^{p}\left(1+b_{\pi(k)}\right), \qquad R_q = \sum_{j=q}^{n}\prod_{k=q}^{j}\left(1+b_{\pi(k)}\right).$$

Można zauważyć, że S_1, S_2 i S_3 to całkowite czasy wykonywania odpowiednich części harmonogramu przy założeniu, że są to jedyne zadania w harmonogramie. Wykorzystując te oznaczenia, wartość funkcji celu możemy przedstawić w postaci $TC = S_1 + S_2 + S_3 + P_pY_{p,q} + R_qX_{p,q} + P_pR_qZ_{p,q}$. W granicznym przypadku (p+1 = q) wartość funkcji celu wyraża się wzorem $\Sigma_{j=1,..n}\ C_j = S_1 + S_3 + P_pR_q$. Spośród tych symboli wartości S_1 i P_p zależą wyłącznie od zadań w pierwszej części harmonogramu, S_3 i R_q od zadań z trzeciej jego części, zaś pozostałe wartości ($S_2, X_{p,q}, Y_{p,q}, Z_{p,q}$) od zadań w środkowej (drugiej) części. Gdy znana jest początkowa (pierwsza) i końcowa (trzecia) część harmonogramu, to aby oszacować wartość funkcji celu całego harmonogramu, musimy oszacować wartości symboli S_2, $X_{p,q}$, $Y_{p,q}$ ($Z_{p,q}$ jest iloczynem wartości współczynników wydłużania powiększonych o jeden i jego wartość nie zależy od ich kolejności). Wartości $X_{p,q}$ oraz $Y_{p,q}$ możemy ograniczyć z dołu przez wartość tych sum obliczoną dla zadań uporządkowanych w kolejności nierosnących (dla $X_{p,q}$) lub niemalejących (dla $Y_{p,q}$) współczynników wydłużania [7]. Obserwację tę możemy wykorzystać także do oszacowania wartości S_2 – jest to suma ($q - p - 1$) składników, z których każdy ma postać taką, jak $X_{p,q}$. S_2 ograniczamy z dołu przez sumę dolnych ograniczeń kolejnych składników. Zdefiniujmy operację REV(π, p, q) jako:

$$REV(\pi, p, q) = \left\{\pi(1),...,\pi(p-1),\pi(q),\pi(q-1),...,\pi(p+1),\pi(p),\pi(q-1),...,\pi(n)\right\} \quad (3)$$

Operacja REV bierze harmonogram π i zwraca harmonogram, w którym zdania na pozycjach p, ..., q zostały odwrócone w stosunku do wejściowego harmonogramu. Wpływ takiej operacji na wartość funkcji celu możemy przedstawić wzorem:

$$TC_{REV(\pi,p,q)} - TC_{\pi} = (P_{p+1} - R_{q-1})(B_{p,q} - A_{p,q}) \quad (4)$$

gdzie:

$$A_{p,q} = \sum_{j=p}^{q}\prod_{k=j}^{q}\left(1+b_{\pi(k)}\right), \quad B_{p,q} = \sum_{j=p}^{q}\prod_{k=p}^{j}\left(1+b_{\pi(k)}\right)$$

Kolejna operacja (nazwijmy ją *SWAP*) polega na zamianie miejscami dwóch zadań w harmonogramie. Jeżeli zamienianymi zadaniami są $J_{\pi(p)}$ oraz $J_{\pi(q)}$, to

$$SWAP(\pi, p, q) = \{\pi(1),...,\pi(p-1),\pi(q),\pi(p+1),...,\pi(q-1),\pi(p),\pi(q+1),...,\pi(n)\} \quad (5)$$

Zmianę wartości funkcji celu, spowodowaną wykonaniem tej operacji, możemy opisać równaniem:

$$TC_{SWAP(\pi,p,q)} - TC_{\pi} = (b_{\pi(p)} - b_{\pi(q)})(R_{q+1}+1)(A_{p+1,q-1}+1) - (P_{p-1}+1)(B_{p+1,q-1}+1) \quad (6)$$

Wynika to z następującego faktu, iż $SWAP(\pi, p, q)$ = REV (REV(π, p + 1, q – 1), p, q). Weźmy teraz pewien ciąg zadań J_p, ..., J_q oraz pewną liczbę $k > 0$.

Twierdzenie 1. *Jeżeli* $b_i \geq k$ ($i = p, ..., q$), *to* $\max(A_{p,q}, B_{p,q}) - \min(A_{p,q}, B_{p,q}) \geq \geq \min(A_{p,q}, B_{p,q}) / k$.

Dowód. Niech $m = q - p + 1$ oraz $\alpha_i = b_{i+p-1}$ dla $i = 1, ..., m$. Wówczas
$A_{p,q} = \Sigma_{j=1,..n} \Pi_{k=1,..j} (\alpha_k + 1)$.
Niech $W_i = \Pi_{j=1,..i} (\alpha_j + 1)$ oraz $S_i = \Sigma_{j=1,..i} W_j$.

$$S_1 \leq \frac{k+1}{k} S_1 = \frac{k+1}{k} W_1$$

$$S_{i+1} = W_{i+1} + S_i \leq W_{i+1} + \frac{k+1}{k} W_i \leq W_{i+1} + \frac{\alpha_{i+1}+1}{k} W_i = W_{i+1} + \frac{1}{k} W_{i+1} = \frac{k+1}{k} W_{i+1}$$

Zatem niezależnie od kolejności zadań, $A_{p,q}$ i $B_{p,q}$ nie będą większe od $((k + 1) / k)\Pi_{j=p,...,q} (b_j + 1)$, czyli $\max(A_{p,q}, B_{p,q}) \leq ((k + 1) / k) \Pi_{j=p,...,q} (b_j + 1)$. Równocześnie zarówno $A_{p,q}$, jak i $B_{p,q}$ jest większe od $\Pi_{j=p,...,q} (b_j + 1)$. Wobec tego:

$$\max\left(A_{p,q}, B_{p,q}\right) - \min\left(A_{p,q}, B_{p,q}\right) \leq \frac{k+1}{k} \prod_{j=p}^{q} \left(b_j + 1\right) - \prod_{j=p}^{q} \left(b_j + 1\right) \leq \frac{1}{k} \min\left(A_{p,q}, B_{p,q}\right).$$

Załóżmy teraz, że w harmonogramie π dla pewnego p i q mamy $b_p > b_q$ oraz $P_{p-1} \geq \geq R_{q+1}(k+1) / k + 1 / k$.

$$\left(B_{p+1,q-1}+1\right)\left(P_{p-1}+1\right) - \left(A_{p+1,q-1}+1\right)\left(R_{q+1}+1\right) \geq$$

$$\geq \left(R_{q+1}+1\right)\left(B_{p+1,q-1} - A_{p+1,q-1} + \frac{1}{k} B_{p+1,q-1} + \frac{1}{k} \right) \geq$$

$$\geq \left(R_{q+1}+1\right)\left(-\frac{1}{k} B_{p+1,q-1} + \frac{1}{k} B_{p+1,q-1} + \frac{1}{k} \right) > 0.$$

Analogicznie: $(B_{p+1,q-1} + 1)(R_{q+1} + 1) - (A_{p+1,q-1} + 1)(P_{p-1} + 1) < 0$. Niezależnie od wartości $A_{p+1,q-1}$ oraz $B_{p+1,q-1}$ o znaku wyrażenia $(B_{p+1,q-1} + 1)(P_{p-1} + 1) - (A_{p+1,q-1} + 1)(R_{q+1} + 1)$ decyduje pozycja większego z wyrazów P_{p-1}, R_{q+1} (jeżeli większy stoi przy $B_{p+1,q-1}$ – wyrażenie jest dodatnie, jeśli przy $A_{p+1,q-1}$ – ujemne). Mając zatem pewien początkowy i końcowy fragment harmonogramu i wiedząc, że przedstawione założenia są spełnione, możemy odrzucić wszystkie harmonogramy o takim początku i końcu, gdyż na pewno nie będzie wśród nich optymalnego – zamiana zadań J_p i J_q miejscami powoduje poprawę harmonogramu. Podobne rozumowanie można przeprowadzić, gdy $P_{p-1} \geq R_{q+1}\,(b_{\min} + i) / (b_{\min} + 1) + i / (b_{\min} + 1)$, co może być łatwiejsze do spełnienia dla bardzo małych wartości współczynników wydłużania zadań. Dowód wykorzystuje obserwację, że $|A_{p,q} - B_{p,q}| \leq (b_{\min} + q - p + 1) / (b_{\min} + 1)\,\Pi_{j=\mathrm{p},\ldots,q}\,(b_j + 1)$.

3.1. Algorytm

Procedura B&B(n, b, h, *poziom*, s, p, q)

n – ilość zadań w instancji; b – wektor uporządkowanych niemalejąco współczynników wydłużania (b_0 to najmniejszy współczynnik wydłużania, b_{n-1} — największy); h – wektor binarny opisujący budowany harmonogram; *poziom* – numer ustalanego w danym kroku zadania; s – częściowa wartość rozwiązania; p – bieżąca wartość współczynnika P_p; q – bieżąca wartość współczynnika R_q

(1) jeżeli *poziom* < 1
  (1.1) jeżeli $s + pq < best$, niech $best = s + pq$ i zapamiętaj h jako dotychczasowy najlepszy harmonogram;
  (1.2) zakończ procedurę
(2) niech $E_s = s + E_1[poziom] + qE_2[poziom] + pE_2[poziom] + pqE_3[poziom]$
(3) jeżeli $E_s \geq best$, zakończ procedurę
(4) niech $B = 1 + b_{poziom}$, $k = b_1 + poziom + 1 / (b_1 + 1)$, $add = poziom + 1 / (b_1 + 1)$
(5) jeżeli $poziom > (k + 1) / k$, niech $k = (k + 1) / k$; $add = 1 / k$
(6) jeżeli $poziom < 1$ lub $p < kq + add$ lub $b_{poziom} = b_{poziom-1}$
  (6.1) niech $h[poziom] = 0$ oraz wywołaj B&B(n, b, h, $poziom - 1$, $s + B(p + 1)$, $B(p + 1)$, q)
(7) jeżeli $kp + add > q$ lub $b_{poziom} = b_{poziom-1}$
  (7.1) niech $h[poziom] = 1$ oraz wywołaj B&B(n, b, h, $poziom - 1$, $s + B(q + 1)$, p, $B(q + 1)$)

Algorytm branch-and-bound

wejście: n – ilość zadań w instancji; b – wektor współczynników wydłużania

(1) uporządkuj zadania tak, aby $b_i \leq b_{i+1}$, dla $i = 1, \ldots, n - 1$
(2) oblicz $E_1[\mathrm{i}] = \Sigma_{j=1,\ldots,i}\,\Sigma_{k=1,\ldots,j}\,\Pi_{l=1,\ldots,j}\,(b_l + 1)$ dla $i = 1, \ldots, n - 1$

(3) oblicz $E_2[i] = \Sigma_{j=1,...,i} \Pi_{k=1,...,j} (b_k + 1)$ dla $i = 1, ..., n - 1$

(4) oblicz $E_3[i] = \Pi_{j=1,...,i} (b_j + 1)$ dla $i = 0, ..., n - 1$

(5) ustaw $h[1] = h[n] = h[n - 1] = 0$

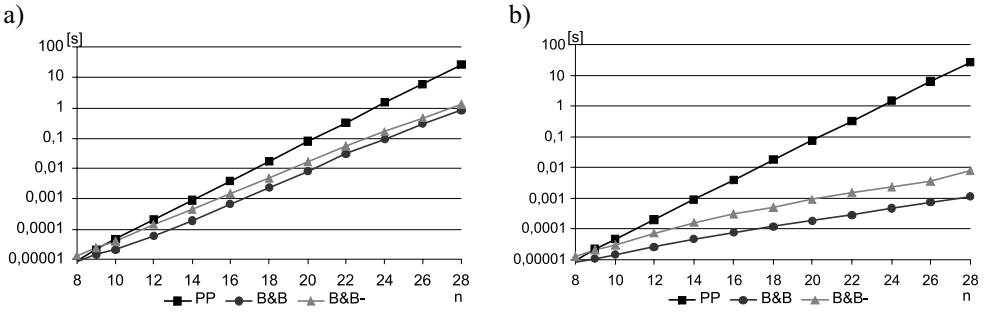
(6) znajdź wstępne rozwiązanie przy użyciu pewnej heurystyki; ustaw *best* na wartość funkcji celu znalezionego rozwiązania pomniejszoną o n

(7) wywołaj B&B(n, b, h, $n - 2$, $(1 + b_{n-2})$, 0, $(1 + b_{n-2})$)

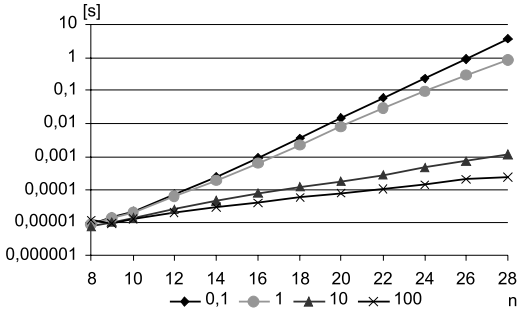
Powyższy algorytm podsumowuje przedstawione rozważania. W krokach (2)–(4) algorytmu branch-and-bound wyznaczane są odpowiednie oszacowania dolne parametrów S_2 (2), $X_{p,q}$, $Y_{p,q}$ (3) i $Z_{p,q}$ (4). W kroku (6) wyznaczana jest, przy użyciu dowolnej heurystyki, wartość początkowa najlepszego rozwiązania. W procedurze B&B, w kroku (1) sprawdzamy, czy cały harmonogram został już skonstruowany. Jeżeli tak, wyznaczana jest jego wartość funkcji celu, i odpowiednio aktualizowane jest najlepsze dotychczasowe rozwiązanie. W krokach (2)–(3) odrzucane są rozwiązania cząstkowe, które na pewno nie będą optymalne (gdyż dolne oszacowanie na uzyskaną wartość przekracza najlepsze dotychczasowe rozwiązanie). W krokach (4)–(5) wyznaczane jest efektywniejsze kryterium odrzucania harmonogramów, które nie na pewno nie będą optymalne. Wreszcie w krokach (6)–(7) do rozwiązania dołączane jest kolejne zadanie (jeżeli tylko nie narusza wybranego kryterium).

4. Wyniki

Początkowa wartość najlepszego znanego rozwiązania podczas testów była wyznaczana przy użyciu prostej, a jednocześnie efektywnej heurystyki *N*1 z pracy [6]. Wszystkie wyniki są średnią z 20–1000 (w zależności od rozmiaru instancji problemu) wykonań algorytmu dla losowych danych. Rysunek 1 oraz tabela 1 przedstawiają porównanie czasu działania algorytmu pełnego przeszukiwania (PP) i branch-and-bound (B&B) dla różnych rozmiarów instancji problemu (symbol „–” oznacza, że dla danego przypadku nie przeprowadzono pomiaru czasu). Dodatkowo na wykresie przedstawiono także czas działania algorytmu branch-and-bound wykorzystującego tylko pierwszą z przedstawionych metod ograniczania przestrzeni (B&B–). Dane losowane były z rozkładem równomiernym z zakresu <0, 1> (rys. 1a), <0, 10> (rys. 1b) oraz <0, 100> (tylko tabela). Rysunek 2 przedstawia różnice w czasie działania w zależności od zakresu wartości współczynników wydłużania dla różnych rozmiarów instancji problemu. Dane były losowane z rozkładem równomiernym z zakresu <0; α> dla $\alpha = 0{,}1, 1, 10, 100$. Można zauważyć, że wraz ze wzrostem zakresu wartości, czas działania algorytmu maleje (przestrzeń jest bardziej efektywnie ograniczana).



Rys. 1. Porównanie czasu działania algorytmu pełnego przeszukiwania i branch-and-bound



Rys. 2. Czas działania algorytmu branch-and-bound w zależności od zakresu danych

Tabela 1
Porównanie czasu działania pełnego przeszukiwania i branch-and-bound

n	PP [s]	B&B(α=1) [s]	B&B(α=10) [s]	B&B(α=100) [s]
8	0,00001	0,00001	0,00001	0,00002
10	0,00005	0,00003	0,00002	0,00002
14	0,00089	0,00022	0,00006	0,00004
18	0,01742	0,00272	0,00015	0,00008
22	0,33307	0,03083	0,00036	0,00015
26	6,17603	0,33480	0,00086	0,00027
30	140,9034	3,84899	0,00266	0,00037
34	–	21,41071	0,01991	0,00049
38	–	156,4273	0,02816	0,00057
40	–	392,5959	0,11415	0,00067

5. Podsumowanie

W pracy przedstawiony został algorytm branch-and-bound dla problemu szeregowania zadań uwarunkowanych czasowo $1 \mid p_i = 1 + a_i s_i \mid \Sigma C_i$. Zastosowany algorytm umożliwia znalezienie optymalnego rozwiązania dla instancji o 6–10 zadań większych (a przy dużych wartościach parametrów zadań jeszcze większych) niż te, które jesteśmy w stanie rozwiązać algorytmem pełnego przeszukiwania (28–30 zadań).

Literatura

[1] Cheng T.C.E., Ding Q., Lin B.M.T., *A concise survey of scheduling with time-dependent processing times*. EJOR, 152, 2004, 1–13.

[2] Gawiejnowicz S., Lai T.-C., Chiang M.-H., *Polynomially solvable cases of scheduling deteriorating jobs to minimize total completion time*. Extended Abstracts of the 7th Workshop on Project and Management Scheduling, University of Osnabrück, 2000, 131–134.

[3] Mosheiov G., *V-shaped policies for scheduling deteriorating jobs*. Operations Research, 39, 1991, 979–991.

[4] Gawiejnowicz S., Kurc W., Pankowska L., *A greedy approach for a time-dependent scheduling problem*. LNCS, 2328, 2002, 79–86.

[5] Gawiejnowicz S., Kurc W., Pankowska L., *Minimalizacja łącznego czasu zakończenia zbioru zadań czasowo-zależnych metodą kolejnych ulepszeń*. Sterowanie procesami dyskretnymi, zarządzanie i inżynieria produkcji, 2004, 47–54.

[6] Ocetkiewicz K.M., *Porównanie heurystyk dla problemu szeregowania zadań czasowo-zależnych o wspólnym podstawowym czasie wykonywania*. Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej, 12, 2007, 145–152.

[7] Browne S., Yechiali U., *Scheduling deteriorating jobs on a single processor*. Operations Research, 38, 1990, 495–498.