STABILNOŚĆ ROZWIĄZAŃ PEWNEGO PROBLEMU SZEREGOWANIA Z PROBABILISTYCZNYMI CZASAMI WYKONYWANIA ZADAŃ

Wojciech BOŻEJKO, Paweł RAJBA, Mieczysław WODECKI

Streszczenie: W pracy badano stabilność rozwiązań wyznaczonych przez algorytmy oparte na metodzie poszukiwania z tabu (tabu serach), dla pewnego NP-trudnego jednomaszynowego problemu szeregowania. Czasy wykonywania zadań są deterministyczne lub zmiennymi losowymi. Dla rozkładów Erlanga losowe zaburzenia danych powodują pogorszenie wyników zaledwie o kilka procent.

Słowa kluczowe: szeregowanie zadań, algorytm przeszukiwania z tabu, spóźnienie zadania, rozkład Erlanga.

1. Wstęp

W wielu zagadnieniach związanych z procesem podejmowania decyzji z dziedziny planowania, zarządzania oraz sterowania występują duże trudności w jednoznacznym określeniu parametrów procesu lub też dane pochodzą z nieprecyzyjnych urządzeń pomiarowych. Podejmowanie decyzji w warunkach niepewności (tj. braku dokładnych wartości parametrów) jest więc codziennością. Prowadzone od kilkudziesięciu lat badania dotyczą głównie modeli deterministycznych. Stosowana w nich logika dwuwartościowa nie jest odpowiednia do modelowania współczesnej rzeczywistości.

W ostatnich latach można zaobserwować duże zainteresowanie metodami sztucznej inteligencji: sieciami neuronowymi, algorytmami ewolucyjnymi, itp. Symulują one występujące w przyrodzie „naturalne" procesy prowadzące (pomimo licznych zakłóceń) do bardzo korzystnych strategii. Jednak elementy niejednoznaczności (niedeterminizmu) występują w tych algorytmach jedynie w procesie przeglądania zbioru rozwiązań dopuszczalnych. Jak wskazują eksperymenty obliczeniowe (zamieszczone np. w pracy [1]) wyznaczane tymi metodami rozwiązania są mało stabilne. Niewielka zmiana parametrów powoduje znaczną zmianę wartości funkcji celu.

W przypadku niepewnych danych, przyjęcie pewnej wielkości (spośród wielu możliwych) i rozwiązanie w ten sposób otrzymanego zadania deterministycznego prowadzi do mało stabilnych rozwiązań ([2]). Problemy optymalizacyjne z niepewnymi danymi można rozwiązywać stosując transformację problemu niedeterministycznego do pewnego zbioru problemów deterministycznych. Do rozwiązywania tak wygenerowanych zagadnień można wówczas stosować znane algorytmy deterministyczne. W wyniku takiego podejścia otrzymujemy zbiór rozwiązań dopuszczalnych (deterministycznych). Na ich na podstawie należy dopiero skonstruować rozwiązanie problemu niedeterministycznego, zachowujące się w miarę stabilnie przy zaburzaniu danych. Ze względu na złożoność obliczeniową problemów optymalizacji dyskretnej (zwykle są one *silnie NP-trudne*) metoda ta jest mało efektywna.

Obliczenia są czasochłonne, a wyniki zamieszczone w literaturze wskazują na brak stabilności wyznaczanych rozwiązań.

Bardziej obiecujące są metody bazujące na elementach probabilistyki lub teorii zbiorów rozmytych. Pozwalają na uwzględnienie niepewności już na etapie budowy modelu matematycznego oraz bezpośrednio w konstruowanych algorytmach. Niepewne parametry mogą być przedstawiane za pomocą rozkładów zmiennych losowych lub liczb rozmytych. To pierwsze podejście [6], [11] i [13] wymaga znajomości pewnych danych statystycznych, a z ich uzyskaniem i weryfikacją mogą być duże trudności. W pewnych jednak dziedzinach gospodarki takich jak: transport, budownictwo, rolnictwo, handel czy turystyka niektóre parametry zachodzących tam procesów mają ze swej natury charakter losowy (zależą np. od pogody, popytu, itp.). Posiadają „długą historię" i istnieje możliwość określenia rozkładów prawdopodobieństw niepewnych danych. W wielu jednak problemach optymalizacji dyskretnej niepewność danych nie ma charakteru losowego, lecz wynika np. z unikalności procesu. Parametry (wielkości liczbowe) są określane wówczas przez eksperta. W tym przypadku naturalnym sposobem reprezentowania niepewności są liczby rozmyte ([3],[8]). Niepewna informacja może być przedstawiana, jako liczba rozmyta, na wiele sposobów. Istnieje także wiele metod porównywania liczb rozmytych. Z powodu braku jednoznaczności, prowadzi to do znacznie różniących się rozwiązań.

W pracy rozpatrujemy jednomaszynowy problem szeregowania zadań z najpóźniejszymi terminami zakończenia oraz minimalizacją sumy kosztów zadań spóźnionych. Czasy wykonywania zadań są deterministyczne lub zmiennymi losowymi o rozkładzie Erlanga. Na bazie tego problemu badana jest odporność na losową zmianę parametrów rozwiązań konstruowanych według metaheurystyki przeszukiwania z tabu.

2. Sformułowanie problemu oraz metoda rozwiązania

Algorytmy oparte na metodzie przeszukiwania z tabu są od wielu lat z powodzeniem stosowane do rozwiązywania NP-trudnych problemów optymalizacji kombinatorycznej. Są proste w implementacji, a wyniki porównawcze zamieszczone w literaturze wskazują, że wyznaczone przez te algorytmy rozwiązania tylko nieznacznie różnią się od najlepszych.

2.1. Jednomaszynowy problem szeregowania zadań

W rozpatrywanym problemie, każde z n zadań (ponumerowanych liczbami 1,...,n) należy wykonać bez przerywania na maszynie, która w dowolnej chwili może wykonywać co najwyżej jedno zadanie. Dla zadania i, niech p_i, d_i, w_i będą odpowiednio: *czasem wykonywania*, *oczekiwanym czasem zakończenia* oraz *karą* za spóźnienie zadania (tj. gdy czas jego zakończenia przekroczy d_i). Należy wyznaczyć taką kolejność wykonywania zadań, aby suma kar była jak najmniejsza.

Niech $J=\{1,2, \ldots ,n\}$ będzie zbiorem wszystkich zadań, a Π zbiorem permutacji elementów z J. Dla dowolnej permutacji $\pi \in \Pi$ przez $C_{\pi(i)}$, $(C_{\pi(i)} = \sum_{j=1}^{i} p_{\pi(j)})$ oznaczmy czas zakończenia wykonywania zadania i w permutacji π, (tj. gdy zadania są wykonywane w kolejności występowania w π). Wówczas

$$U_{\pi(i)} = \begin{cases} 0, & \text{gdy } C_{\pi(i)} \le d_{\pi(i)}, \\ 1, & \text{w przeciwnym przypadku,} \end{cases} \qquad (1)$$

nazywamy *spóźnieniem* zadania, a $w_{\pi(i)}U_{\pi(i)}$ karą za spóźnienie. Przez

$$F(\pi) = \sum_{i=1}^{n} w_{\pi(i)}U_{\pi(i)},$$

oznaczamy *koszt permutacji*. W rozważanym problemie, należy więc wyznaczyć permutację optymalną (o minimalnej karze) w zbiorze wszystkich permutacji Π. W literaturze jest on oznaczany przez $1||\sum w_iU_i$ i należy do klasy problemów *silnie NP-trudnych* (Karp [6]). Algorytmy optymalne jego rozwiązywania oparte na metodzie programowania dynamicznego przedstawiono w pracach: Lwlera i Moore [5] - algorytm pseudowielomianowy o złożoności obliczeniowej $O(n\min\{\sum_j p_j, \max_j\{d_j\}\})$ oraz Sarniego [9] (dla danych całkowitoliczbowych algorytm ten ma złożoności $O(n\min\{\sum_j p_j, \sum_j w_j, \max_j\{d_j\}\})$, a oparte na metodzie podziału i ograniczeń – w pracach Pottsa i van Wassenhove [7], Villareala i Bulina [10] oraz Wodeckiego [12]. Algorytmy dokładne pozwalają na efektywne wyznaczenie rozwiązań optymalnych jedynie wówczas, gdy liczba zadań nie przekracza 50 (80 w środowisku wieloprocesorowym [12]). W praktyce stosuje się algorytmy przybliżone (konstrukcyjne lub typu popraw). W większości są one adaptacjami algorytmów rozwiązywania bardziej znanego i częściej badanego problemu jednomaszynowego oznaczanego przez $1||\sum w_iT_i$ ([12]). Jest także wiele prac poświęconych szczególnym przypadkom rozpatrywanego w pracy problemu, dla których istnieją algorytmy optymalne o wielomia-nowej złożoności obliczeniowej.

2.2. Algorytm przeszukiwania z tabu

Do rozwiązywania NP-trudnych problemów optymalizacji kombinatorycznej stosuje się obecnie niemal wyłącznie algorytmy przybliżone. Wyznaczane przez te algorytmy rozwiązania są, z punktu widzenia wielu zastosowań, w pełni zadawalające (często różnią się od najlepszych rozwiązań o zaledwie kilka procent). Najlepsze z nich należą do grupy metod poszukiwań lokalnych (local search), których działanie sprowadza się do bezpośredniego przeglądania pewnego obszaru zbioru rozwiązań dopuszczalnych. Jedną z realizacji tej metody jest przeszukiwanie z tabu (tabu search), którego podstawowymi elementami są:

- *otoczenie* – podzbiór zbioru rozwiązań dopuszczalnych,
- *ruch* – funkcja, która przekształca jedno rozwiązanie w inne,
- *lista tabu* – lista zawierająca atrybuty pewnej liczby rozpatrywanych rozwiązań.

Niech $\pi \in \Pi$ będzie permutacją startową, L_{TS} listą tabu, a π^* najlepszym do tej pory znalezionym rozwiązaniem.

Algorithm Tabu Search (*TS*)

```
1  repeat
2     Wyznaczyć otoczenie N(π) permutacji π;
3     Usunąć z N(π) permutacje zakazane przez listę L_TS;
4     Wyznaczyć permutację δ ∈ N(π), taką że
5     F(δ) = min{F(β) β ∈ N(π)};
```

6 ***if*** ($F(\delta) < F(\pi^*)$) ***then*** $\pi^* := \delta$;
7 Umieścić atrybuty δ na liście L_{TS} ;
8 $\pi := \delta$
9 ***until*** (*warunek_zakończenia*).

Złożoność obliczeniowa algorytmu zależy przede wszystkim od sposobu generowania i przeglądania otoczenia. Poniżej przedstawiamy główne elementy algorytmu.

Ruch i otoczenie

Niech $\pi = (\pi(1),\ldots,\pi(n))$ będzie permutacją ze zbioru Π . Przez π_l^k ($k,l = 1,2,\ldots,n$) oznaczamy permutację otrzymaną z π przez zamianę pozycjami w π elementu $\pi(k)$ z $\pi(l)$. Mówimy wówczas, że permutacja π_l^k została wygenerowana z π przez ruch *typu zamień* s_l^k (tj. permutacja $\pi_l^k = s_l^k(\pi)$). Dalej, niech $M(\pi(k))$ będzie zbiorem ruchów typu zamień elementu $\pi(k)$ w permutacji π , a

$$M(\pi) = \bigcup_{\pi(k)\in L(\pi)} M(\pi(k)),$$

zbiorem wszystkich takich ruchów. Liczba elementów tego zbioru jest z góry ograniczone przez $(n-1)/2$.

Otoczeniem elementu $\pi \in \Pi$ jest zbiór permutacji

$$N(\pi) = \left\{ s_l^k(\pi) :\ s_l^k \in M(\pi) \cap L(\pi) \right\},$$

gdzie $L(\pi) = \{\pi(i) :\ C_{\pi(i)} > d_{\pi(i)}\}$ jest *zbiorem zadań spóźnionych* w π .

Przy implementacji algorytmu z otoczenia usuwa się permutacje, których atrybuty znajdują się na liście ruchów zakazanych L_{TS} .

Lista ruchów zakazanych

Aby zapobiec powstaniu cyklu, pewne atrybuty każdego ruchu zapamiętuje się na liście ruchów zakazanych. Obsługiwana jest ona na zasadzie kolejki FIFO. Wykonując ruch $s_j^r \in M(\pi)$ (tj. generując z $\pi \in \Pi$ permutację π_j^r) na listę tabu L_{TS} zapisujemy atrybuty tego ruchu, czyli trójkę $(\pi(r), j, F(\pi_j^r))$.

Załóżmy, że rozpatrujemy ruch $s_l^k \in M(\beta)$ generujący z $\beta \in \Pi$ permutację β_l^k . Jeżeli na liście L_{TS} jest trójka (r, j, Ψ) taka, że $\beta(k) = r$, $l = j$ oraz $F(\beta_l^k) \le \Psi$, to ruch taki jest zakazany i usuwany ze zbioru $M(\beta)$.

3. Probabilistyczne parametry zadań

W literaturze rozpatrywano problemy szeregowania z losowymi czasami zadań, głównie o rozkładzie normalnym lub jednostajnym (Van den Akker and Hoogeveen [11]) oraz wykładniczym Pinedo [6].

Niech $J = \{1,2,\ldots,n\}$ będzie zbiorem zadań do wykonania na jednej maszynie. Zakładamy, że czasy wykonywania zadań p_i $(i = 1,2,\ldots,n)$ są niezależnymi zmiennymi losowymi. Wówczas, dla ustalonej kolejności występowania zadań w permutacji π, czas zakończenia wykonywania zadania $C_{\pi(i)} = \sum_{j=1}^{i} p_{\pi(j}$ jest zmienną losową. Zmiennymi losowymi są także spóźnienie

$$U_{\pi(i)} = 0 \text{, gdy } C_{\pi(i)} \le d_{\pi(i)} \text{ oraz } U_{\pi(i)} = 1 \text{, gdy } C_{\pi(i)} > d_{\pi(i)}$$

oraz funkcja celu

$$F(\pi) = \sum_{i=1}^{n} w_{\pi(i)} U_{\pi(i)}. \quad (3)$$

W algorytmie przeszukiwania z tabu ***TS***, wybierając najlepszy element z otoczenia (instrukcja 5), porównuje się wartości funkcji celu. Ponieważ (3) jest zmienną losową, dlatego zastąpimy ją kombinacją wypukłą wartości oczekiwanej oraz odchylenia standardowego

$$W(\pi) = c \cdot E(F(\pi)) + (1-c) \cdot D(F(\pi)) \quad (c \in [0,1]). \quad (4)$$

W probabilistycznej wersji algorytmu ***TS*** należy w miejsce funkcji celi F (instrukcja 5 i 6) wstawić funkcję W zdefiniowaną w (4).

3.1 Czasy wykonywania zadań o rozkładzie Erlanga

Zakładamy, że czas wykonywania zadania ma rozkład Erlanga $p_i \sim \mathrm{E}(\alpha_i, \lambda), i \in J$. Rozkład ten lepiej niż inne rozkłady modeluje rzeczywisty charakter zjawisk występujących w problemach szeregowania związanych z budownictwem, transportem, produkcją małoseryjną, itp. Tym bardziej, że w praktyce częściej następuje wydłużenie czasu wykonywania zadania, niż jego skrócenie.

Punktem wyjścia są dane deterministyczne (p_i, w_i, d_i) $(i = 1,2,\ldots,n)$. Proces randomizacji polega na wyznaczeniu takich zmiennych o rozkładzie Erlanga $p_i \sim \mathrm{E}(\alpha_i, \lambda)$, aby wartość oczekiwana $\mathrm{E}(p_i) = p_i$. Przyjmujemy więc parametr

$$\lambda = \max\{\frac{2}{\min\{p_i : 1 \le i \le n}, 1\} \text{ oraz } \alpha_i = p_i \lambda.$$

Wówczas, termin zakończenia wykonywania zadania $i \in J$ (w permutacji $\pi = (1,2,\ldots,n)$)

$$C_i = \sum_{j=1}^{i} p_j \sim \mathrm{E}(\alpha_1 + \ldots + \alpha_i, \lambda).$$

Niech $F_i(x) = F_{\tilde{p}_1+\ldots+\tilde{p}_i}(x)$ będzie dystrybuantą czasu zakończenia wykonywania *i*-tego zadania $\tilde{C}_i$. Wówczas wartość oczekiwana

$$E(\tilde{U}_i) = 0 \cdot P(\tilde{C}_i \le d_i) + 1 \cdot P(\tilde{C}_i > d_i) = 1 - F_i(d_i)$$

oraz

$$E(\tilde{F}(\pi)) = E\left(\sum_{i=1}^{n} w_i \tilde{U}_i\right) = \sum_{i=1}^{n} w_i E\left(\tilde{U}_i\right) = \sum_{i=1}^{n} w_i (1 - F_i(d_i)). \quad (8)$$

Łatwo zauważyć, że $E(\tilde{U}_i^2) = 1 - F_i(d_i)$, więc wariancja

$$D^2(\tilde{U}_i) = D^2(\sum_{i=1}^{n} w_i \tilde{U}_i) = E(\tilde{U}_i^2) - (E(\tilde{U}_i))^2 = F_i(d_i)(1 - F_i(d_i)).$$

Wobec tego

$$D^2(\tilde{F}(\pi)) = \sum_{i=1}^{n} w_i \left(F_i(d_i)(1 - F_i(d_i))\right) + 2\sum_{i<j} w_i w_j \operatorname{cov}(\tilde{U}_i, \tilde{U}_j).$$

Kowariancję $\operatorname{cov}(\tilde{U}_i, \tilde{U}_j)$ pomiędzy zmiennymi $\tilde{U}_i$ oraz $\tilde{U}_j$ obliczamy ze wzoru

$$\operatorname{cov}(\tilde{U}_i, \tilde{U}_j) = E(\tilde{U}_i \tilde{U}_j) - E(\tilde{U}_i) E(\tilde{U}_j).$$

Ostatecznie

$$D^2(\tilde{F}(\pi)) = \sum_{i=1}^{n} w_i (F_i(d_i)(1 - F_i(d_i))) +$$
$$2\sum_{i<j} w_i w_j (FI + SI - (1 - F_i(d_i))(1 - F_j(d_j))), \quad (9)$$

gdzie

$$FI = \int_{d_i}^{d_j} \int_{d_j - x}^{\infty} f_i(x) f_j(y) dy dx, \quad \text{oraz} \quad SI = \int_{d_j}^{\infty} \int_{0}^{\infty} f_i(x) f_j(y) dy dx.$$

Wartości *FI* oraz *SI* można łatwo policzyć sprowadzając je do postaci wyrażonej przez dystrybuanty zmiennych losowych. Wobec tego, aby obliczyć wartość funkcji $W(\pi)$ określonej przez (4), należy skorzystać ze wzoru (8) oraz (9) (ustalając wcześniej eksperymentalnie parametr $c \in [0,1]$).

4. Stabilność algorytmów

W tym rozdziale wprowadzamy pewną miarę pozwalającą na badanie wpływu zmiany parametrów zadań na wartości funkcji celu (2), tj. stabilność rozwiązań.

Niech $\delta = ((p_1, w_1, d_1),\ldots,(p_n, w_n, d_n))$ będzie przykładem danych (deterministycznych) dla rozpatrywanego problemu szeregowania. Przez $\mathrm{D}(\delta)$ oznaczamy zbiór danych generowanych z δ przez zaburzenie czasów wykonywania zadań. Zaburzenie polega na zmianie tych czasów na losowo wyznaczone wartości. Zaburzone dane $\gamma \in \mathrm{D}(\delta)$ są więc postaci $\gamma = ((p'_1, w_1, d_1),\ldots,(p'_n, w_n, d_n))$, gdzie czas wykonywania p'_i $(i = 1,\ldots,n)$ jest realizacją zmiennej losowej p_i o rozkładzie Erlanga $\mathrm{E}(\lambda, \alpha_i)$ (Rozdział 3), przy czym wartość oczekiwana $E(\tilde{p}_i) = p_i$.

Niech $A = \{AD, AP\}$, gdzie AD i AP jest odpowiednio algorytmem deterministycznym oraz probabilistycznym (tj. rozwiązującymi przykłady z deterministycznymi lub losowymi czasami wykonywania zadań) dla rozpatrywanego problemu. Przez π_δ oznaczamy rozwiązanie (permutację) wyznaczoną przez algorytm A dla danych δ. Dalej, niech $F(A, \pi_\delta, \varphi)$ będzie kosztem wykonania zadań (2) dla przykładu φ w kolejności określonej przez rozwiązanie (permutację) π_δ wyznaczoną przez algorytmem A dla danych δ. Wówczas

$$\Delta(A, \delta, \mathrm{D}(\delta)) = \frac{1}{|\mathrm{D}(\delta)|} \sum_{\varphi \in \mathrm{D}(\delta)} \frac{F(A, \pi_\delta, \varphi) - F(AD, \pi_\varphi, \varphi)}{F(AD, \pi_\varphi, \varphi)},$$

nazywamy *stabilnością rozwiązania* π_δ (przykładu δ) wyznaczonego przez algorytm A na zbiorze danych zaburzonych $\mathrm{D}(\delta)$. Wyznaczając permutację π_φ, za rozwiązanie startowe w algorytmie A przyjęto π_δ (wówczas, $F(\pi_\delta, \varphi) - F(\pi_\varphi, \varphi) \geq 0$). Wyrażenie (5) jest średnim błędem względnym najlepszego rozwiązania π_δ w stosunku do najlepszych rozwiązań wyznaczonych, dla każdego przykładu danych zaburzonych $\varphi \in \mathrm{D}(\delta)$. Jeżeli $\Delta(A, \delta, \mathrm{D}(\delta)) = 0$ to oznacza, że dla każdego przykładu danych $\varphi \in \mathrm{D}(\delta)$, permutacja π_δ jest najlepszym rozwiązaniem.

Niech Ω będzie zbiorem przykładów deterministycznych dla rozpatrywanego problemu szeregowania zadań. Współczynnik stabilności algorytmu A na zbiorze Ω definiujemy następująco:

$$S(A, \Omega) = \frac{1}{|\Omega|} \sum_{\delta \in \Omega} \Delta(A, \delta, \mathrm{D}(\delta)). \quad (10)$$

Jest to średni błąd względny rozwiązań deterministycznych w stosunku do najlepszych rozwiązań danych zaburzonych wyznaczonych przez algorytm deterministyczny. Jeżeli $S(A, \Omega) = 0$ to oznacza, że rozwiązania wyznaczane przez algorytm A, dla danych deterministycznych nie są wrażliwe na zaburzenia. Innymi słowy, najlepsze rozwiązanie

wyznaczone dla dowolnych danych deterministycznych $\delta \in \Omega$ jest także najlepszym rozwiązaniem dla każdych danych zaburzonych $\varphi \in \mathrm{D}(\delta)$.

Do oceny wrażliwości rozwiązań na zaburzenia parametrów można stosować także inne miary, np. oparte o błąd średniokwadratowy.

5. Wyniki porównawcze algorytmów

Przedstawione w pracy algorytmy były testowane na wielu przykładach. Dane deterministyczne generowano (podobnie jak w pracy Potts i in. [7]) w następujący sposób. Dla ustalonej liczby zadań n (n=40, 50, 100) wyznaczano n trójek (p_i,w_i,d_i), i=1,2,...,n, gdzie czas wykonania zadania p_i oraz koszt w_i są realizacją zmiennej losowej o rozkładzie jednostajnym odpowiednio z przedziału [1,100] oraz [1,10]. Podobnie, linie krytyczne są losowane z przedziału [P(1-TF-RDD/2,P(1-TF+RDD/2] zależnego od parametrów RDD, TF=0.2,0.4,0.6,0.8,1.0 przy czym, $P = \sum_{i=1}^{n} p_i$. Dla każdej pary parametrów RDD, TF (takich par jest 25) generowano 5 przykładów. W sumie, zbiór danych deterministycznych Ω zawiera 375 przykładów (po 125, dla każdego n).

Przy uruchamianiu każdego algorytmu rozwiązaniem startowym była permutacja naturalna $\pi = (1,2,\ldots,n)$. Ponadto, przyjęto następujące parametry:

- długość listy ruchów zakazanych: n,
- maksymalna liczba iteracji algorytmu (*Warunek_Zakończenia*): n,
- we wzorze (4) parametr, $c = 0.8$.

Obliczenia algorytmu deterministycznego AD wykonano na przykładach ze zbioru Ω, a algorytmu probabilistycznego $\overline{AP}$ (rozkłady Erlanga) na przykładach ze zbioru $\overline{\Omega}$. W celu zbadania stabilności algorytmów (tj. wyznaczenia wartości parametru (10)), dla każdego przykładu danych deterministycznych $\delta \in \Omega$ wygenerowano 100 przykładów danych zaburzonych ($|\mathrm{D}(\delta)| = 10$) (sposób ich generowania jest przedstawiony w poprzednim rozdziale). Każdy z tych przykładów został rozwiązany przez algorytm AD. Na bazie tych obliczeń wyznaczono współczynniki stabilności obu algorytmów. Wyniki porównawcze przedstawiono w Tabeli 1.

Tabela 1. Stabilność algorytmów (średni błąd względny $S(A,\Omega)$ (10)).

liczba zadań n	Algorytm	
	deterministyczny AD	probabilistyczny $\overline{AP}$
40	0,72	0,08
50	0,64	0,07
100	0,53	0,06
średnio	**0,63**	**0,07**

Po wykonaniu n iteracji średni współczynniki stabilności algorytmu deterministycznego $S(AD,\Omega) = 0,63$, a probabilistycznego $S(\overline{AP},\Omega) = 0,07$. Oznacza to, że zaburzenie rozwiązania wyznaczonego przez algorytm AD powoduje pogorszenie się wartości funkcji celu średnio o 63%. W przypadku algorytmu $\overline{AP}$ pogorszenie to wynosi

średnio jedynie 7%. Tak więc średni błąd algorytmu deterministycznego jest 9 razy większy od średniego błędu algorytmu probabilistycznego. Maksymalny błąd algorytmu AP nie przekracza 31%, a algorytmu detrministycznego AP wynosi ponad 600%. Wykonano także obliczenia dla większej liczby iteracji. Współczynniki stabilności algorytmów uległy jedynie niewielkiej poprawie. Liczba n iteracji algorytmu opartego na metodzie przeszukiwania z tabu jest bardzo mała. Dzięki temu średni czas obliczeń jednego przykładu, na komputerze osobistym z procesorem Pentium 2,6 GHz, nie przekracza jednej sekundy.

Przeprowadzone eksperymenty obliczeniowe wykazały jednoznacznie, że rozwiązania wyznaczone przez algorytm probabilistyczny z czasami wykonywania zadań o rozkładzie Erlanga są bardzo stabilne.

5. Podsumowanie

W pracy zaproponowano metodę modelowania niepewnych danych przy pomocy zmiennych losowych o rozkładzie Erlanga. Przedstawiono konstrukcję algorytmu opartego na metodzie poszukiwania z tabu, dla rozwiązywania pewnego jednomaszynowego problemu szeregowana zadań z probabilistycznymi parametrami. Opisano metodę generowania danych testowych (z losowo zaburzonymi czasami wykonywania zadań) oraz zbadano stabilność algorytmów, tj. wpływ zmiany czasów wykonywania zadań na wartości funkcji celu. Otrzymane wyniki jednoznacznie wskazują, że znacznie stabilniejszy jest tzw. algorytm probabilistyczny

Praca częściowo finansowana z projektu badawczego MNiSW N514 232237.

Literatura

1. Bożejko W., Wodecki M.: Liczby rozmyte w problemach szeregowania zadań, Komputerowo Zintegrowane Zarządzanie, WNT, Warszawa 2002, 135-142.
2. Bożejko W., Wodecki M.: Stabilność metaheurystyk dla wybranych problemów szeregowania zadań, Komputerowo Zintegrowane Zarządzanie, WNT, Warszawa 2001, 85-94.
3. Itoh T., Ishii H.: Fuzzy due-date scheduling problem with fuzzy processing times, Int. Trans. Oper. Res., 6, 1999, 639-647.
4. Karp R.M.: Reducibility among Combinatorial Problems, Complexity of Computations, R.E. Millerand J.W. Thatcher (Eds.), Plenum Press, New York, 1972, 85-103.
5. Lawler E.L., Moore J.M.: A Functional Equation and its Applications to Resource Allocation and Sequencing Problems, Management Sci., 16, 1969, 77-84.
6. Pinedo M.: Scheduling: Theory, Algorithms, and Systems, Prentice Hall, 1995.
7. Potts C.N., Van Wassenhove L.N.: A Branch and Bound Algorithm for the Total Weighted Tardiness Problem, Operations Research, 33, 1985, 177-181.
8. Prade H.: Using fuzzy set theory in a scheduling problem, Fuzzy Sets and Systems, 2**,** 1979, 153–165.
9. Sahni S.K.: Algorithms for Scheduling Independent Jobs, J.Assoc. Comput. Mach., 23, 1976, 116-127.
10. Villareal F.J., Bulfin R.L.: Scheduling a Single Machine to Minimize the Weighted Number of Tardy Jobs, IE Trans., 15, 1983, 337-343.

11. Van den Akker M., Hoogeveen H.: Minimizing the number of late jobs in a stochastic setting usinga chance constraint, Journal of Scheduling, 11, 2008, 59–69.
12. Wodecki M.: A Branch-and-Bound Parallel Algorithm for Single-Machine Total Weighted Tardiness Problem, J. Adv. Manuf. Technology, 2008, 996-1004.
13. Zhu X., Cai X.: General Stochastic Single-Machine Scheduling with Regular Cost Functions, Math. Comput. Modelling, Vol. 26, No. 3, 1997, 95–108.

Dr Wojciech BOŻEJKO
Instytut Informatyki Automatyki i Robotyki Politechniki Wrocławskiej
ul. Janiszewskiego 11/17, 50-372 Wrocław
e-mail: wbo@ict.pwr.wroc.pl

Mgr Paweł RAJBA
Instytut Informatyki Uniwersytetu Wrocławskiego
ul.Joliot-Curie 50-383 Wrocław
e-mail: pawel.rajba@ii.uni.wroc.pl

Dr Mieczysław WODECKI
Instytut Informatyki Uniwersytetu Wrocławskiego
ul.Joliot-Curie 50-383 Wrocław
e-mail: mwd@ii.uni.wroc.pl