Wojciech BOŻEJKO
Politechnika Wrocławska
Paweł RAJBA
Uniwersytet Wrocławski
Mieczysław WODECKI
Politechnika Wrocławska

WŁASNOŚCI ELIMINACYJNE DLA PEWNEGO PROBABILISTYCZNEGO PROBLEMU SZEREGOWANIA ZADŃ

Streszczenie. W pracy rozpatrujemy pewien problem szeregowania zadań z żądanymi terminami zakończenia. Są one reprezentowane przez zmienne losowe o rozkładzie normalnym. Przedstawiamy algorytm oparty na metodzie tabu search. Definiujemy probabilistyczne bloki, których własności eliminacyjne są stosowane przy generowaniu otoczeń. Dzięki temu uzyskaliśmy znaczne skrócenie czasu obliczeń.

ELIMINATION PROPERTIES FOR A PROBABILISTIC PROBLEM OF SCHEDULING JOBS

Summary. In the paper work we consider a problem of jobs scheduling with requested due dates. They are represented by random variables with a normal distribution. We propose an algorithm based on the tabu search method.and define probabilistic blocks whose elimination properties are used during neighborhood generating. Thanks to this we have achieved a significant reduction of the calculations time.

1. Wstęp

W wielu dziedzinach gospodarki takich jak: transport, budownictwo, rolnictwo, handel czy turystyka pewne parametry zachodzących tam procesów mają ze swej natury charakter losowy (zależą np. od pogody, popytu, warunków geologicznych, itp.). Niepewność danych przekłada się bezpośrednio na wielkość ryzyka, stąd np. w budownictwie stosuje się bufory (czasowe, kosztowe, materiałowe) wyznaczane na podstawie norm określonych dla standardowych czynności. Często prowadzi to do znacznego obniżenia konkurencyjności (pozycji przetargowej) przedsiębiorstwa.

Obiecujące są prowadzone w ostatnich latach badania, w których niepewność jest uwzględniana już na etapie budowy modelu oraz konstrukcji algorytmu. W tym celu stosuje się metody probabilistyki (jeżeli niepewne informacje mają charakter losowy) lub teorii zbiorów rozmytych. W pierwszym przypadku ([6], [7], [3], [14]) istotna jest znajomość rozkładu danych. Ponieważ niektóre procesy mają znaczną ”historię”, więc

istnieje możliwość określenia rozkładów zmiennych losowych. W wielu jednak problemach niepewność danych nie ma charakteru losowego, lecz wynika np. z unikalności procesu. W tym przypadku naturalnym sposobem reprezentowania niepewności są liczby rozmyte ([2]).

W pracy, na przykładzie pewnego jednomaszynowego problemu szeregowania zadań, przedstawiamy metodę konstrukcji algorytmu przybliżonego (przeszukiwania z tabu) dla losowych żądanych terminów zakończenia zadań. Przy przeszukiwaniu otoczeń stosujemy własności eliminacyjne bloków (słabsze od tych stosowanych w problemach szeregowania z kryterium $C_{\max}$). Badamy także stabilność rozwiązań, tj. wpływ losowych zaburzeń danych na zmiany wartości funkcji kryterialnej.

2. Jednomaszynowy problem szeregowania zadań

W rozpatrywanym problemie, każde zadanie ze zbioru $J = \{1, \ldots, n\}$ należy wykonać bez przerywania na maszynie, która w dowolnej chwili może wykonywać tylko jedno zadanie. Dla zadania $i \in J$, niech p_i, d_i, w_i będą odpowiednio: *czasem wykonywania*, *żądanym terminem zakończenia* oraz *karą za spóźnienie*. Jeżeli jest ustalona pewna kolejność wykonywania zadań oraz C_i ($i \in J$) jest terminem zakończenia zadania $i \in J$, to gdy $C_i > d_i$ wówczas zadanie nazywamy *spóźnionym*. W przeciwnym przypadku zadanie nazywamy *terminowym*. Rozpatrywany problem szeregowania (ang. *single machine problem with tardiness costs*, w skrócie **SMTC**) polega na wyznaczeniu kolejności wykonywania zadań o *najmniejszym koszcie*, tj. dla której suma kar zadań spóźnionych jest minimalna.

Niech Π będzie zbiorem permutacji elementów z J. Dla permutacji $\pi \in \Pi$, $C_{\pi(i)} = \sum_{k=1}^{i} p_{\pi(k)}$ jest terminem zakończenia zadania $\pi(i)$ w permutacji π, tj. gdy zadania są wykonywane zgodnie z kolejnością ich występowania w π. Zmienną

$$U_{\pi(i)} = \begin{cases} 0, & C_{\pi(i)} \leqslant d_{\pi(i)}, \\ 1, & C_{\pi(i)} > d_{\pi(i)}, \end{cases}$$

nazywamy *spóźnieniem* zadania $\pi(i)$ w permutacji π, a sumę kar zadań spóźnionych

$$W(\pi) = \sum_{k=1}^{n} w_{\pi(i)} U_{\pi(i)} \tag{1}$$

kosztem wykonywania zadań (*wagą permutacji*).

Rozpatrywany w pracy problem **SMTC** polega więc na wyznaczeniu w zbiorze Π permutacji o najmniejszym wadze, tj. permutacji $\pi^* \in \Pi$, takiej że

$$W(\pi^*) = \min_{\pi \in \Pi} \left\{ \sum_{k=1}^{n} w_{\pi(i)} U_{\pi(i)} \right\}.$$

W literaturze problem ten jest oznaczany $1||\sum w_i U_i$ i należy do klasy problemów NP-trudnych. Algorytmy optymalne jego rozwiązywania oparte na metodzie programowania dynamicznego przedstawili Lawler, Moore [4] (algorytm pseudo wielomianowy o złożoności obliczeniowej $O(n \min\{\sum_j p_j, \max_j\{d_j\}\})$) oraz Sahni [9] (dla danych całkowitoliczbowych algorytm ma złożoność $O(n \min\{\sum_j p_j, \sum_j w_j, \max_j\{d_j\}\})$), a oparte metodzie podziału i ograniczeń: Potts, Van Wassenhove [8] Villareal, Bulfin [11] oraz Wodecki [13].

Algorytmy dokładne dla jednomaszynowych NP-trudnych problemów szeregowania pozwalają na efektywne wyznaczenie rozwiązań optymalnych jedynie wówczas, gdy liczba zadań nie przekracza 80 ([13], algorytm równoległy). Dlatego, w praktyce stosuje się najczęściej algorytmy przybliżone (głównie metaheurystyki). Są to zazwyczaj adaptacje algorytmów rozwiązywania bardziej znanego w literaturze problemu $1||\sum_i w_i T_i$, gdzie $T_i = \max\{0, C_i - d_i\}$ (np. Wodecki [13]).

3. Metoda przeszukiwania z tabu

Do rozwiązywania NP-trudnych problemów optymalizacji dyskretnej stosuje się przede wszystkim algorytmy przybliżone. Z praktycznego punktu widzenia są one w większości zupełnie wystarczające (wiele z nich znajduje rozwiązania gorsze od optimum o nie więcej niż kilka procent). Najlepsze należą do grupy metod poszukiwań lokalnych (ang. *local search*), których działanie sprowadza się do iteracyjnego przeglądania zbioru rozwiązań dopuszczalnych i wybieraniu najlepszego ze względu na ustalone kryterium. Jedną z nich jest przeszukiwanie z tabu (ang. *tabu search*, TS). Głównymi elementami metody są:

- otoczenie – podzbiór zbioru rozwiązań dopuszczalnych, którego elementy są przeglądane,
- ruch – funkcja przekształcająca jedno rozwiązanie w inne,
- lista tabu – lista zawierająca atrybuty pewnej liczby ostatnio rozpatrywanych rozwiązań.

Algorytm oparty na tej metodzie zazwyczaj kończy działanie po wykonaniu ustalonej liczby iteracji. Złożoność obliczeniowa algorytmu jest zdeterminowana przez procedurę wyznaczenie otoczenia, jego przeglądanie oraz warunek zakończenia. Poniżej przedstawiamy główne elementy algorytmu rozwiązywania problemu $1||\sum w_i U_i$.

3.1. Ruch i otoczenie

Niech $\pi = (\pi(1), \ldots, \pi(n))$ będzie pewną permutacją zadań zbioru J. Ruch typu *zamień* (ang. *swap move*) $s_l^k,\ k \neq l,\ k, l = 1, 2, \ldots, n$, zamienia pozycjami dwa elementy $\pi(k)$ z $\pi(l)$ (znajdujące się odpowiednio na pozycjach k oraz l), generując z π nową permutację $s_l^k(\pi) = \pi_l^k$, przy czym

$$\pi_l^k(i) = \begin{cases} \pi(i), & \text{jeśli } i \neq k \wedge i \neq l, \\ \pi(k), & \text{jeśli } i = l, \\ \pi(l), & \text{jeśli } i = k. \end{cases}$$

W skrócie będzie on nazywany *s-ruchem*. Wszystkich możliwych takich ruchów jest $n(n-1)/2$ (oczywiście zachodzi równość $s_l^k = s_k^l$). Złożoność obliczeniowa wykonania *s-ruchu* wynosi $O(1)$.

Otoczeniem permutacji π jest zbiór

$$\mathcal{N}(\pi) = \left\{ s_l^k(\pi) \vee \pi(k) \in \mathcal{M}(\pi) \ \vee \ \pi(l) \in \mathcal{M}(\pi) \right\}. \tag{2}$$

Przy implementacji algorytmu z otoczenia usuwa się elementy, których atrybuty znajdują się na liście ruchów zakazanych.

3.2. Lista ruchów zakazanych

Aby zapobiec powrotowi do tej samej permutacji, po niewielkiej liczbie iteracji algorytmu, pewne atrybuty każdego ruchu zapamiętuje się na liście ruchów zakazanych. Obsługiwana jest ona na zasadzie kolejki FIFO. Wykonując ruch s_j^r (tj. generując z $\pi \in \Pi$ permutację π_j^r) na listę tabu zapisujemy atrybuty tego ruchu, czyli trójkę $(\pi(r), j, W(\pi_j^r))$. Załóżmy, że wyznaczając pewne otoczenie permutacji β rozpatrujemy ruch s_l^k generujący element β_l^k. Jeżeli na liście jest trójka (r, j, Ψ) taka, że $\beta(k) = r$, $l = j$ oraz $W(\beta_l^k) \geqslant \Psi$, to ruch taki pomijamy. W literaturze lista ruchów zakazanych jest realizowana na wiele sposobów.

4. Randomizacja

W tym rozdziale rozpatrujemy problem **SMTC**, w którym żądane terminy zakończenia zadań są zmiennymi losowymi o rozkładach normalnym. W literaturze badano głównie problem szeregowania z losowymi czasami wykonywania zadań. Van den Akker i Hoogeveen [10] rozpatrywali rozkłady jednostajny i normalny, a Pinedo [6] rozkłady wykładnicze. Obszerny przegląd metod i algorytmów rozwiązywania problemów optymalizacji kombinatorycznej z losowymi parametrami przedstawił Vondrák [12].

W praktyce istnieją duże trudności z ustaleniem rozkładów prawdopodobieństwa losowych parametrów modelu. Szczególnie, gdy mamy do czynienia z unikalnymi procesami i w związku z tym brak jest jakichkolwiek danych statystycznych. Zazwyczaj w takim przypadku, korzystając z wiedzy eksperta, przyjmujemy pewne wartości (deterministyczne) dla nieznanych parametrów. W pierwszej kolejności przedstawimy metodę randomizacji takich parametrów.

Niech (p_i, w_i, d_i), $i = 1, 2, \ldots, n$ będzie pewnym przykładem (danymi) rozpatrywanego w pracy problemu **SMTC**. Proces randomizacji polega na wyznaczeniu zmiennych losowych $\tilde{d}_i$, $i \in J$ reprezentujących żądane terminy zakończenia zadań przy założeniu, że wartość oczekiwana $E(\tilde{d}_i) = d_i$. Będziemy rozpatrywać rozkład normalny, tj. $\tilde{d}_i \sim N(d_i, c \cdot d_i)$, gdzie parametr c wyznaczamy eksperymentalnie. Łatwo sprawdzić, że rozkłady te spełniają przyjęte założenie, czyli $E(\tilde{d}_i) = d_i$ $(i = 1, \ldots, n)$.

Losowa wersja instancji problemu **SMTC** jest *n*-elementowym ciągiem trójek $(p_i, w_i, \tilde{d}_i)$, gdzie $\tilde{d}_i$ jest zmienną losową reprezentującą żądany termin zakończenia zadania, a p_i i w_i są liczbami, tak jak w wersji deterministycznej.

Niech $\pi \in \Pi$ będzie dowolną permutacją zadań. Termin zakończenia zadania $\pi(i) \in J$, $C_{\pi(i)} = \sum_{j=1}^{i} p_{\pi(j)}$, a spóźnienie

$$\tilde{U}_{\pi(i)} = 0, \text{ gdy } C_{\pi(i)} \leqslant \tilde{d}_{\pi(i)} \text{ oraz } \tilde{U}_{\pi(i)} = 1, \text{ gdy } C_{\pi(i)} > \tilde{d}_{\pi(i)}.$$

Odpowiadająca funkcji celu (1) zmienna losowa jest więc postaci

$$\tilde{W}(\pi) \sum_{i=1}^{n} w_{\pi(i)} \widetilde{U}_{\pi(i)}. \tag{3}$$

W takim przypadku, do oceny rozwiązań (permutacji zbioru Π) stosuje się zazwyczaj ważone kombinacje liniowe momentów centralnych zmiennej losowej (3). Z przeprowadzonych eksperymentów obliczeniowych wynika, że wystarczy uwzględnić

jedynie wartość oczekiwaną. W związku z tym do oceny rozwiązania będziemy stosowali funkcje

$$W_1(\pi) \;=\; \sum_{i=1}^{n} w_{\pi(i)} E(\tilde{U}_{\pi(i)}). \tag{4}$$

Łatwo sprawdzić, że dla rozkładu normalnego

$$E(\tilde{U}_{\pi(i)}) = P(C_{\pi(i)} > \tilde{d}_{\pi(i)}) = F_{\tilde{d}_{\pi(i)}}(C_{\pi(i)}),$$

gdzie F_X jest dystrybuantą zmiennej losowej *X*. Ostatecznie więc

$$W_1(\pi) \;=\; \sum_{i=1}^{n} w_{\pi(i)} E(\tilde{U}_{\pi}(i)) = \sum_{i=1}^{n} w_{\pi}(i)(E(\tilde{U}_{\pi(i)}). \tag{5}$$

Powyższe kryterium będziemy stosowali w algorytmach, opartych na metodzie przeszukiwania z tabu, rozwiązywania problemu **SMTC** z losowymi terminami zakończenia zadań.

Niech $\mathcal{AD}$ będzie algorytmem rozwiązywania problemu **SMTC** z kryterium optymalności (1), którego główne elementy przedstawiono w rozdziale 3. W dalszej części pracy będziemy go nazywali **algorytmem deterministycznym**. Przez $\mathcal{APN}$ oznaczamy modyfikacje algorytmu detrministycznego dla losowych danych (tj. losowych terminów zakończenia zadań) o rozkładzie normalnym. W skrócie nazywamy go **algorytmem probabilistycznym**. Jako kryterium porównawcze rozwiązań dopuszczalnych problemu stosuje się w tych algorytmach funkcję (5)).

5. Metoda blokowa dla problemu z losowymi parametrami

Od początku lat 80-tych ubiegłego wieku własności blokowe (przegląd pośredni elementów przestrzeni rozwiązań) sa z powodzeniem stosowane w konstrukcjach algorytmów metaheurystycznych. W pierwszej kolejności, przy rozwiązywaniu wielomaszynowych problemów z minimalizacją czasów zakończenia wykonywania wszystkich zadań (kryterium $C_{\max}$). Dokładnie zostały one opisane w monografii Grabowski i in. [1]. Bloki oraz ich własności były, jak do tej pory, stosowane głównie w konstrukcjach algorytmów rozwiązywania problemów deterministycznych. W tym rozdziale rozpatrujemy prosty, jednomaszynowy (ale NP-trudny) problem szeregowania zadań. Przedstawiamy definicje oraz metody konstrukcji pewnych subpermutacji, które posiadają podobne własności eliminacyjne jak klasyczne bloki. W odróżnieniu od bloków, będziemy je nazywali α *blokami*. W bloku, każda zmiana kolejności elementów nie generuje rozwiązania o mniejszej wartości funkcji kryterialnej. W przypadku α bloków ta własność nie musi być spełniona. Mogą być więc eliminowane lepsze od bieżącego rozwiązania. Parametr α umożliwia kontrolowanie tego błędu. W rozpatrywanym jednomaszynowym problemie szeregowania zadań definiujemy dwa typy α bloków:

- zadań terminowych,
- zadań spóźnionych.

Poniżej przedstawiamy algorytmy wyznaczania w permutacji zadań (bieżącym rozwiązaniu algorytmu TS) α bloków.

1. α **blok zadań terminowych**:
   - kolejno, dla $i = 1, 2, \ldots, n$ wykonaj:
   - jeśli $p_i \leqslant d_i$, to i jest pierwszym elementem bloku oraz $d_{min} := d_i$;
   - *sprawdzenie, czy do bloku można dodać kolejne elementy*
     — aktualizacja $d_{i_\alpha} = \alpha * d_i$
     — jeśli $C_i > d_{min}$, do bloku dodaj element i oraz
     — jeżeli $d_{min} > d_{i_\alpha}$, to $d_{min} := d_i$
2. α **blok zadań spóźnionych**:
   - kolejno, dla $i = 1, 2, \ldots, n$ wykonaj:
   - jeśli $p_i > d_i$, to i jest pierwszym elementem bloku oraz $S := S_i$;
   - *sprawdzenie, czy do bloku można dodać kolejne elementy*
     — jeśli $S + p_i > d_{i_\alpha}$, do bloku dodaj element i oraz
     — $d_{i_\alpha} = \alpha * d_i$

W algorytmie TS, generując otoczenie (2) pomijamy ruchy zmieniające kolejność elementów w α blokach. Dzięki temu zmniejszy się wielkość otoczenia a co za tym idzie, czas wykonywania pojedynczej iteracji algorytmu.

6. Stabilność algorytmu

Eksperymenty obliczeniowe przeprowadzono w celu zbadania stabilności algorytmów, a ściślej - rozwiązań wyznaczanych przez te algorytmy. Stabilność jest pewną miarą umożliwiającą oszacowanie wpływu zaburzeń danych na zmiany wartości funkcji celu. W pierwszej kolejności przedstawiamy metodę generowania zbioru danych, na bazie których będzie badana stabilność algorytmów.

Niech $\delta = ((p_1, w_1, d_1), \ldots, (p_n, w_n, d_n))$ będzie przykładem danych (deterministycznych) dla problemu szeregowania **SMTC**, a $\mathfrak{D}(\delta)$ zbiorem danych generowanych z δ przez zaburzenie żądanych terminów zakończenia zadań. Zaburzenie polega na zmianie terminów zakończenia $(d_i, i = 1, \ldots, n)$ na losowo wyznaczone wartości (tj. liczby generowane zgodnie z pewnym przyjętym rozkładem, np. jednostajnym, normalnym, itd.). Dowolny element zbioru $\mathfrak{D}(\delta)$ jest ciągiem trójek postaci $(p_1, w_1, d'_1), \ldots, (p_n, w_n, d'_n)$, gdzie zaburzony żądany termin zakończenia d'_i $(i = 1, \ldots, n)$ jest realizacją pewnej zmiennej losowej. Wobec tego, zbiór $\mathfrak{D}(\delta)$ zawiera przykłady danych deterministycznych dla problemu **SMTC** różniących się pomiędzy sobą jedynie żądanymi terminami zakończenia zadań.

Niech $\mathcal{A} = \{A_{popt}, A\}$, gdzie A_{popt}, A są algorytmami rozwiązywania problemu **SMTC** przy czym, A_{popt} jest algorytmem "prawie optymalnym", a A algorytmem którego badamy stabilność (w szczególności może to być A_{popt}). Dalej, niech $F(\mathcal{A}, \pi_\delta, \varphi)$ będzie kosztem wykonania zadań (1) dla przykładu φ w kolejności określonej przez permutację π_δ wyznaczonej przez algorytm $\mathcal{A}$ dla danych δ. Wówczas

$$\Delta(\mathcal{A}, \delta, \mathfrak{D}(\delta)) = \frac{1}{|\mathfrak{D}(\delta)|} \sum_{\varphi \in \mathfrak{D}(\delta)} \frac{F(\mathcal{A}, \pi_\delta, \varphi) - F(A_{popt}, \pi_\varphi, \varphi)}{F(A_{popt}, \pi_\varphi, \varphi)},$$

nazywamy *stabilnością rozwiązania* π_δ (wyznaczonego przez algorytm $\mathcal{A}$ dla przykładu δ) na zbiorze danych zaburzonych $\mathfrak{D}(\delta)$. Parametr ten pozwala na ocenę "odporności" rozwiązania π_δ na ewentualne zmiany terminów zakończenia zadań.

Niech Ω będzie zbiorem przykładów dla problemu **SMTC**. Wyrażenie

$$\mathbb{S}(\mathcal{A}, \Omega) = \frac{1}{\Omega} \sum_{\delta \in \Omega} \Delta(\mathcal{A}, \delta, \mathfrak{D}(\delta)) \tag{6}$$

nazywamy *współczynnikiem stabilności* algorytmu $\mathcal{A}$ na zbiorze Ω. Im mniejsza jest jego wartość, tym wyznaczone przez algorytm rozwiązania są stabilniejszy, tj. losowe zaburzenia danych powodują niewielkie pogorszenie wartości funkcji kosztu.

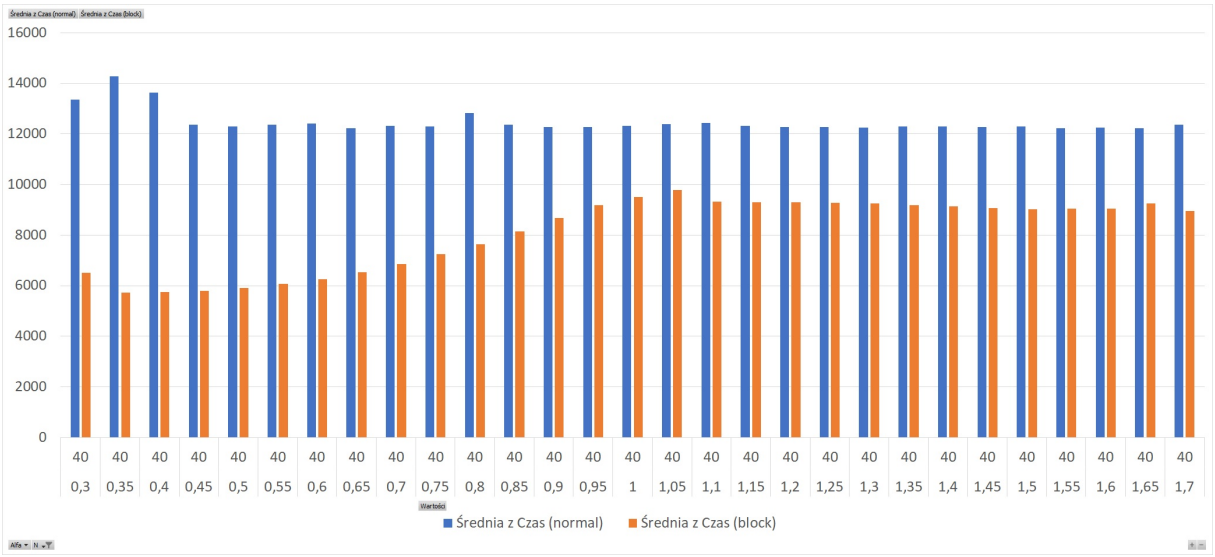
7. Eksperymenty obliczeniowe

Eksperymenty obliczeniowe przeprowadzono w celu zbadania efektywności stosowania α bloków w konstrukcji algorytmu probabilistycznego $\mathcal{APN}$. Dane deterministyczne Ω pobrano z biblioteki OR-Library [5] (są to przykłady dla problemu $1||\sum_i w_i T_i$). Dla każdego $n = 40, 50$ i 100 dane te zawierają 375 przykładów. Dla każdej instancji problemu deterministycznego wyznaczono instancję danych probabilistycznych ze zmienną losową o rozkładach: normalnym $\tilde{d} \sim N(d_i, c \cdot d_i)$, gdzie $c \in \{0.01, 0.02, \ldots, 0.2, 0.3, 0.4, 0.5\}$. Zbiór tych danych (zwanych probabilistycznymi) oznaczymy przez $\tilde{\Omega}$. Obliczenia algorytmu $\mathcal{AD}$ wykonano na przykładach ze zbiory Ω, a algorytmu $\mathcal{APN}$ na przykładach ze zbioru $\tilde{\Omega}$.

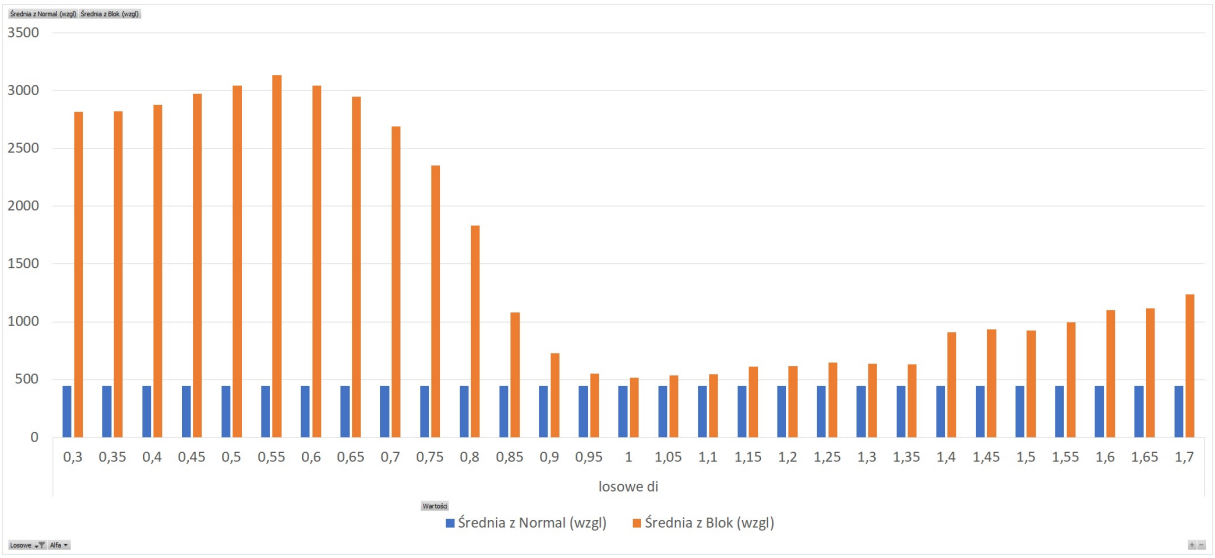
Dla każdego przykładu danych deterministycznych $\delta = ((p_1, w_1, d_1), \ldots, (p_n, w_n, d_n))$ ($\delta \in \Omega$) wygenerowano 100 przykładów danych zaburzonych (tj. $|\mathcal{D}(\delta)| = 100$). Zaburzenie polega na zmianie $(d_i,\ i = 1, \ldots, n)$ na losowo wyznaczone wartości, zgodnie z rozkładem $N(d_i, c \cdot d_i)$. W sumie wyznaczono 37500 przykładów. Zostały one następnie rozwiązane przez algorytm $\mathcal{AD}$. Otrzymane wyniki stanowiły podstawę do wyznaczenia współczynnika stabilności (6) badanego algorytmu. Przy uruchamianiu każdego algorytmu za permutację startową przyjęto $\pi = (1, 2, \ldots, n)$, a ponadto:

- długość listy ruchów zakazanych: n,
- liczba iteracji algorytmu: $n/2$ lub n.

W pierwszej kolejności zbadano wpływ stosowania bloków na czasy obliczeń algorytmu $\mathcal{APN}$ (na PC z zegarem 2.8GHz). Na rysunku 1 przedstawiono średnie czasy obliczeń w zależności od parametru α z przedziału od 0.3 do 1.7. Obliczenia wykonano dla liczby zadań $n = 40$. Największe przyspieszenie, ponad dwukrotne, uzyskano dla $\alpha = 0.35$, a najmniejsze (ponad 20%), dla $\alpha = 1.05$. Otrzymano więc znaczące skrócenie czasu obliczeń. W przypadku probabilistycznych problemów szeregowania zadań bardzo ważna jest stabilność wyznaczanych rozwiązań. Im rozwiązanie jest bardziej stabilnie, tym jest mniej wrażliwe na zmiany losowych parametrów. Na rysunku 2 przedstawiono współczynnik stabilności (6) algorytmu $\mathcal{APN}$ z wykorzystaniem bloków oraz bez bloków. Dzięki stosowania własności blokowych w procedurze generowania otoczenia wyznaczane przez algorytm TS rozwiązania okazały się znacznie stabilniejsze. Podobne wyniki otrzymano dla pozostałych przykładów, tj. dla $n = 50$ oraz 100.



Rys. 1. Czasy obliczeń algorytmu $\mathcal{APN}$ wariantu z oraz bez bloków ($n = 40$).



Rys. 2. Stabilność rozwiązań algorytmu z i bez bloków.

8. Uwagi i wnioski

W pracy rozpatrywano problem szeregowania zadań, w którym niepewne dane są reprezentowane przez zmienne losowe o rozkładzie normalnym. Przedstawiono konstrukcję algorytmu opartego na metodzie przeszukiwania z tabu. Wprowadzono tzw. α bloki o podobnych własnościach eliminacyjnych jak klasyczne bloki stosowane w algorytmach rozwiązywania problemów szeregowania z kryterium $C_{\max}$. Przeprowadzono eksperymenty obliczeniowe w celu zbadania wpływu bloków na czasy obliczeń oraz stabilności wyznaczanych rozwiązań. Otrzymane wyniki jednoznacznie wskazują, że zastosowanie bloków skraca znacznie czas obliczeń oraz poprawia stabilność rozwiązań. Zastosowanie elementów probabilistyki w adaptacji metody przeszukiwania z tabu pozwala skutecznie rozwiązywać także problemy z niepewnymi danymi. Są

to zazwyczaj bardzo trudne praktyczne problemy optymalizacyjne, znacznie lepiej opisujące rzeczywistość niż modele deterministyczne.

Praca powstała w wyniku realizacji projektu badawczego o nr DEC 2017/25/B/ST7 /02181 finansowanego ze srodków Narodowego Centrum Nauki.

LITERATURA

1. Grabowski J., Nowicki E., Smutnicki C.: Metoda blokowa w zagadnieniach szeregowania zadń, AOW EXIT, Warszawa, 2003.
2. Iscibuchi H., Yamamoto N., Misaki S., Tanaka H.: Local Searcg Algorithm for Flow Shop Scheduling with Fuzzy Due-Dates, International Journal of Production Economics, 33, 1994, p. 53–66.
3. Jang W., Klein C.M.: Minimizing the expected number of tardy jobs when processing times are normally distributed, Operations Research Letters, 30, 2002, p. 100–106.
4. Lawler E.L., Moore J.M.: A Functional Equation and its Applications to Resource Allocation and Sequencing Problems, Management Science, 16, 1969, p. 77–84.
5. OR Library http://www.ms.ic.ac.uk/info.html
6. Pinedo M.: Stochastic Scheduling with Release Dates and Due Dates, Operations Research, Vol. 31, No. 3, 1983, p. 559–572 .
7. Pinedo M., Scheduling: Theory, Algorithms, and Systems. Prentice Hall, 1995.
8. Potts C.N., Van Wassenhove L.N.: A Branch and Bound Algorithm for the Total Weighted Tardiness Problem, Operations Research, Vol. 33, 1985, p. 177–181.
9. Sahni S.K.: Algorithms for Scheduling Independent Jobs, J.Assoc. Comput. Match., 23, 1976, 116–127.
10. Van den Akker M.: Hoogeveen H., Minimizing the number of late jobs in a stochastic setting usinga chance constraint, J. Sched., 11, 2008, 59–69.
11. Villareal F.J., Bulfin R.L.: Scheduling a Single Machine to Minimize the Weighted Number of Tardy Jobs, IEE Trans., 15, 1983, p. 337–343.
12. Vondrák J.: Probabilistic methods in combinatorial and stochastic optimization, PhD, MIT, 2005.
13. Wodecki M.: A Branch-and-Bound Parallel Algorithm for Single-Machine Total Weighted Tardiness Problem, Advanced Manufacturing Technology, 37, 2007, p. 996–1004.
14. Zhu X., Cai X.: General Stochastic Single-Machine Scheduling with Regular Cost Functions, Math. Comput. Modelling, Vol. 26, No. 3, 1997, p. 95–108.