Wojciech BOŻEJKO, Jarosław PEMPERA
Politechnika Wrocławska
Mieczysław WODECKI
Uniwersytet Wrocławski

SZYBKI ALGORYTM ROZWIĄZYWANIA CYKLICZNEGO PROBLEMU GNIAZDOWEGO

Streszczenie. W pracy jest rozpatrywany problem harmonogramowania operacji w cyklicznym elastycznym systemie gniazdowym. Przedstawiono nową, bardzo szybką, metodę wyznaczania czasu cyklu. W konstrukcji algorytmu herustycznego zastosowano otoczenie, którego generowanie jest inspirowane grą w golfa. Przy jego przeszukiwaniu korzysta się z dolnego ograniczenia funkcji kryterialnej.

A FAST ALGORITHM FOR THE FLEXIBLE JOB SHOP PROBLEM

Summary. Problem of operations scheduling in a cyclic flexible jobs shop is considered in the paper. The new, very fast, method of cycle time determination is presented. The neighborhood inspired by the golf game is applied in the heuristic algorithm designing. Lower bound of the criterion function is used in the neighborhood searching.

1. Wstęp

Harmonogramowanie operacji w elastycznym systemie gniazdowym wymaga jednoczesnego podjęcia decyzji na dwóch poziomach: (i) przydziału operacji do maszyn, (ii) wyznaczenia kolejności wykonywania operacji na każdej maszynie. W stosunku do klasycznych problemów szeregowania zadań jest to znaczne uogólnienie i istotnie utrudnia konstrukcję efektywnych algorytmów. Zdecydowana większość prac poświęconych elastycznemu problemowi gniazdowemu dotyczy minimalizacji czasu zakończenia wykonywanych wszystkich operacji. Ze względu na NP-trudność tego problemu, uwaga badaczy koncentruje się na konstrukcji algorytmów heurystycznych. W cyklicznym systemie produkcyjnym, podstawowy zbiór zadań wykonywany jest wielokrotnie w ustalonych odstępach czasu (Kampmeyer [6], Brucker i Kampmeyer [5], Smutnicki i Smutnicki [7], Bożejko i in. [4]). Umożliwia to znaczne uproszczenie czynności logistycznych związanych z dostarczaniem surowców oraz odbiorem wyrobów gotowych, ponieważ czynności te są wykonywane w regularnych odstępach czasu. Podstawową trudnością z jaką mamy do czynienia przy konstruowaniu algorytmów, dla tego typu problemów, jest brak efektywnych metod wyznaczania czasu cyklu oraz dobrych dolnych lub górnych oszacowań. Do rozwiązania rozpatrywanego w pracy problemu zastosowano algorytm oparty na metodzie przeszukiwania z tabu, w którym otoczenie jest

inspirowane grą w golfa (dokładny jego opis zamieszczono w pracy Bożejko i in. [2]).

2. Elastyczny problem gniazdowy

W tym rozdziale krótko przedstawiamy elastycznym problem gniazdowy, a w następnym - cykliczną wersję tego problemu, będącą zasadniczym tematem pracy.

Dany jest zbiór zadań $\mathcal{J} = \{1, 2, \ldots, n\}$, które należy wykonać na maszynach ze zbioru $\mathcal{M} = \{1, 2, \ldots, m\}$. Zadanie jest ciągiem pewnych operacji występujących zgodnie z porządkiem technologicznym. Dla każdej operacji określony jest podzbiór maszyn. Operacja musi być wykonana, bez przerywania, na jednej z maszyn z tego podzbioru. Ze względu na różną wydajność maszyn, czas wykonania operacji zależy od przydzielonej maszyny. Problem (w skrócie oznaczany przez FJS) polega na przydzieleniu o peracji do maszyn oraz wyznaczeniu kolejności wykonywania operacji na maszynach (zgodnie z relacją porządku technologicznego), aby zoptymalizować pewne kryterium (dokładnie problem ten jest opisany w pracy Bożejko i in. [2], [4]).

Niech $\mathcal{O} = \{1, 2, \ldots, o\}$ będzie zbiorem wszystkich operacji. Zbiór ten można rozbić na ciągi odpowiadające zadaniom, przy czym zadanie $j \in \mathcal{J}$ jest ciągiem o_j operacji, które będą kolejno wykonywane na odpowiednich maszynach. Operacje te są indeksowane liczbami $(l_{j-1} + 1, \ldots, l_{j-1} + o_j)$, gdzie $l_j = \sum_{i=1}^{j} o_i$ jest liczbą operacji pierwszych j zadań, $j = 1, 2, \ldots, n$, przy czym $l_0 = 0$, $o = \sum_{i=1}^{n} o_i$. Dalej, niech $\mathcal{M}^i \subset \mathcal{M}$ $(i \in \mathcal{O})$ będzie zbiorem maszyn, na których może być wykonywana operacja i, a $p_{i,k}$ $(k \in \mathcal{M}^i)$ będzie czasem wykonywania operacji i na maszynie k.

Przez $\mu = (\mu_1, \ldots, \mu_o)$ oznaczamy przydział operacji do maszyn, gdzie $\mu_a \in \mathcal{M}^a$ jest maszyną przydzieloną do wykonania operacji $a \in \mathcal{O}$. Zbiór

$$\mathcal{O}^l = \{a \in \mathcal{O} : \ \mu_a = l\} \tag{1}$$

zawiera operacje wykonywane na maszynie $l \in \mathcal{M}$, przy czym $\cup_{i=1}^{m} \mathcal{O}^i = \mathcal{O}$.

Niech permutacja π_l będzie pewną kolejnością wykonywania operacji ze zbioru $\mathcal{O}^l$ na maszynie l $(|O^l| = n_l)$, oraz Π^l zbiorem wszystkich permutacji elementów z $\mathcal{O}^l$. Kolejność wykonywania operacji na maszynach jest określona przez złożenie (konkatenację) m permutacji $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$, $\pi_i \in \Pi^i$, $i = 1, 2, \ldots, m$. Niech Φ będzie zbiorem wszystkich takich permutacji. Zauważmy, że permutacja $\pi \in \Phi$ jednoznacznie określa przydział operacji do maszyn oraz kolejność wykonywania operacji.

Dowolne rozwiązanie $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$ $(\pi \in \Phi,)$ może być reprezentowane przez graf skierowany $H(\pi) = (\mathcal{V}, \mathcal{E}(\pi))$. Wierzchołki odpowiadają operacjom, tj. $\mathcal{V} = \mathcal{O}$. Waga wierzchołka $v \in \mathcal{O}$ jest równa p_v - czasowi wykonywania operacji v na maszynie $\mu(v)$. Zbiór łuków $\mathcal{E} = \mathcal{R} \cup \mathcal{K}(\pi)$, gdzie:

1) $\mathcal{R} = \bigcup_{j=1}^{n} \bigcup_{i=1}^{o_j - 1} \{(l_{j-1} + i, l_{j-1} + i + 1)\}$, zawiera łuki reprezentujące porządek technologiczny, a

2) $\mathcal{K}(\pi) = \bigcup_{k=1}^{m} \bigcup_{i=1}^{|\mathcal{O}^k| - 1} \{(\pi_k(i), \pi_k(i+1))\}$, łuki łączące operacje wykonywane na tej samej maszynie, tj. reprezentują kolejność π_k, $(k = 1, 2, \ldots, m)$.

3. Cykliczny elastyczny problem gniazdowy

W cyklicznym systemie produkcyjnym, ustalony zbiór zadań zwany MPS-em (ang. *minimal part set*), jest wykonywany wielokrotnie. Zakładamy, że w każdym z MPS-ów, na każdej maszynie operacje są wykonywane w takiej samej kolejności. Wobec tego, w każdym cyklicznym harmonogramie kolejność wykonywania operacji może być reprezentowany przez złożenie permutacji operacji na poszczególnych maszynach w pierwszym MPS-ie. Cykliczność procesu wymaga spełnienia następującego ograniczenia:

(A) każda operacja jest kolejno wykonywana po upływie czasu cyklu.

Dla ustalonej kolejności $\pi \in \Phi$ (rozwiązania problemu FJS), niech $\mathcal{S}^k = (S_1^k, S_2^k, \ldots, S_o^k)$ będzie ciągiem terminów rozpoczęcia wykonywania operacji w k-tym MPS-ie, gdzie S_i^k oznacza termin rozpoczęcia wykonywania operacji i na maszynie μ_i w k-tym cyklu. Założyliśmy, że harmonogram czasowy pracy systemu jest cykliczny. Oznacza to, że istnieje stała $T(\pi)$ (okres) taka, że

$$S_{\pi(i)}^{k+1} = S_{\pi(i)}^k + T(\pi), \quad i = 1, ..., o, \ k = 1, 2, ... \tag{2}$$

Powyższa równość jest realizacją ograniczenia (A).

Wielkość $T(\pi)$ zależy oczywiście od permutacji π i jest nazywany *czasem cyklu*. Minimalną wartość $T(\pi)$, dla ustalonej kolejności wykonywania operacji na maszynach π, będziemy nazywać *minimalnym czasem cyklu* i oznaczać przez $T^\circ(\pi)$. Ponieważ kolejność wykonywania operacji w ramach każdego MPS-a jest taka sama, więc wystarczy wyznaczyć momenty rozpoczęcia wykonywania operacji $S_1, S_2, \ldots, S_o$ dla pierwszego MPS-a i dokonać ich przesunięcia o wielkość $T(\pi)$. Wobec tego

$$S_{\pi(i)}^k = S_{\pi(i)} + (k-1) \cdot T(\pi), \tag{3}$$

jest momentem rozpoczęcia operacji $i \in \mathcal{O}$ w k-tym cyklu, $k = 1, 2, ...$

4. Wyznaczanie minimalnego czasu cyklu

W tym rozdziale przedstawimy nowę metodę, dla ustalonej kolejności wykonywania operacji na maszynach (permutacji zbioru Φ), wyznaczenia minimalnego czasu cyklu. Metoda ta bazuje na grafie reprezentującym pierwszych $m+1$ MPS-ów (dla uproszczenia zapisu przyjmujemy, że $\eta = m+1$).

4.1. Graf cykliczny

Niech $\pi \in \Phi$ będzie pewnym rozwiązaniem, a $H^1(\pi) = (\mathcal{V}^1, \mathcal{E}^1(\pi))$ pierwszą składową, tj, grafem reprezentującym kolejność wykonywania operacji na maszynach dla pierwszego MPS-a (opis konstrukcji grafu zamieszczono w rozdziale 2.).

Przez $H^l(\pi) = (\mathcal{V}^l, \mathcal{E}^l(\pi))$ $(l = 2, 3, \ldots, \eta)$ oznaczamy graf reprezentującym kolejność wykonywania operacji dla l-tego MPS-a. Zbiór wierzchołków tego grafu

$$\mathcal{V}^l = \{v + (l-1) \cdot o : \ v \in \mathcal{V}^1\}. \tag{4}$$

Para wierzchołków ze zbioru $\mathcal{V}^l$ jest łukiem

$$(u, v) \in \mathcal{E}^1(\pi) \Longleftrightarrow (u + (l-1) \cdot o, v + (l-1) \cdot o) \in \mathcal{E}^l(\pi). \tag{5}$$

Graf $H^l(\pi)$ będziemy nazywali l-tą składową.

Zbiór wierzchołków grafu $H^l(\pi)$

$$\mathcal{A}^l = \{v \in \mathcal{V}^l : \ v = \pi_j(1) + (l-1)\cdot o, \ j = 1, 2, \ldots, m\}, \tag{6}$$

zawiera pierwsze operacje, a zbiór

$$\mathcal{B}^l = \{u \in \mathcal{V}^l : \ u = \pi_j(n_j) + (l-1)\cdot o, \ j = 1, 2, \ldots, m\}, \tag{7}$$

ostatnie operacje zadań wykonywane przez poszczególne maszyny w l-tym MPS-ie.

Dla ustalonej permutacji $\pi \in \Phi$ rozpatrujemy η pierwszych MPS-ów. Przypisujemy im graf $G^{\oplus}(\pi) = (\mathcal{V}^{\oplus}, \mathcal{E}^{\oplus}(\pi))$, zwany *grafem cyklicznym*, będący sumą η pierwszych kolejnych składowych, tj.

$$G^{\oplus}(\pi) = H^1(\pi) \oplus H^2(\pi) \oplus, \ldots, \oplus H^{\eta}(\pi), \tag{8}$$

przy czym, zbiór wierzchołków

$$\mathcal{V}^{\oplus} = \mathcal{V}^1 \cup \mathcal{V}^2 \cup, \ldots, \cup \mathcal{V}^{\eta},$$

a zbiór łuków

$$\mathcal{E}^{\oplus}(\pi) = \mathcal{E}^1(\pi) \cup \mathcal{E}^2(\pi) \cup, \ldots, \cup \mathcal{E}^{\eta}(\pi) \cup \mathcal{W},$$

gdzie $\mathcal{W}$ jest zbiorem łuków pomiędzy kolejnymi komponentami. Łączą one ostatnią operację wykonywaną na maszynie w pewnej komponencie z pierwszą wykonywaną na tej samej maszynie w następnej komponencie, a dokładnie

$$\mathcal{W} = \{(u, v) : \ u \in \mathcal{B}^i, v \in \mathcal{A}^{i+1}, \ \ \mu_u = \mu_v, \ i = 1, 2, \ldots, m\}.$$

Rozpatrujemy wierzchołek $a \in \mathcal{A}^1$. Odpowiada on pierwszej operacji pewnego zadania w pierwszym MPS-ie. Z definicji grafu $G^{\oplus}(\pi)$, wierzchołek $a^l = a + (l-1)\cdot o$ odpowiada, tej samej operacji, ale w l-tej składowej. Przez $L^l(a, a^l)$ oznaczamy długość najdłuższej drogi w grafie $G^{\oplus}(\pi)$ z wierzchołka a do wierzchołka $a^l, \ l = 2, 3, \ldots, \eta$.

Definiujemy macierz $\mathbf{L}$ o n wierszach i m kolumnach (indeksowanych, dla uproszczenia zapisu, liczbami $2, 3, \ldots, \eta$), której element

$$\lambda_{a,l} = L^l(a, a^l)/(l-1), \ a \in \mathcal{A}^1, \ l = 2, 3, \ldots, \eta. \tag{9}$$

Dalej, niech

$$\Lambda^*(\pi) = \max_{v \in \mathcal{A}^1} \max_{2 \leqslant l \leqslant \eta} \{\lambda_{v,l}\}, \tag{10}$$

będzie maksymalnym elementem macierzy $\mathbf{L}$. Jeżeli $\Lambda^* = L^k(a, a^k)/(k-1)$, to drogę $P^k(a, a^k)$ o długości $L^k(a, a^k)$ nazywamy *ścieżką krytyczną* w grafie $G^{\oplus}(\pi)$. Poniżej przedstawiamy dwa twierdzenia (dowody są przedstawione w pracy [3]) dotyczące zależności pomiędzy wartością Λ^*, a minimalnym czasem cyklu $T^{\circ}(\pi)$.

Twierdzenie 1. *Jeżeli $\pi \in \Phi$ jest dopuszczalną kolejnością wykonywania operacji, to minimalny czas cyklu $T^{\circ}(\pi) \geqslant \Lambda^*$.*

Twierdzenie 2. *Jeżeli $\pi \in \Phi$ jest rozwiązaniem dopuszczalnym problemu CFJS, to minimalny czas cyklu $T^{\circ}(\pi) \leqslant \Lambda^*$.*

Z twierdzenia 1 i 2 wynika, że dla ustalonej permutacji $\pi \in \Phi$ (tj. kolejności wykonywania zadań na maszynach w pierwszym MPS-ie) wartość $\Lambda^*(\pi)$ jest minimalnym czasem cyklu, tj. $T^\circ(\pi) = \Lambda^*(\pi)$. Wobec tego, aby zmniejszyć długość cyklu, czyli $T^\circ(\pi)$, należy z π wygenerować permutację β, dla której wartość $\Lambda^*(\beta)$ będzie mniejsza od $\Lambda^*(\pi)$. Jeżeli więc $\Lambda^*(\pi) = L^k(a, a^k)/(k-1)$, to warunkiem koniecznym (zmniejszenia wartości $\Lambda^*(\pi)$) jest skrócenie długości ścieżki krytycznej $L^k(a, a^k)$ w grafie $G^\oplus(\pi)$. Generowanie permutacji β będzie polegało na zmianie kolejności wykonywania pewnych operacji na maszynie lub przeniesienie operacji na inną maszynę z tego samego gniazda.

Niech $P^k(a, a^k)$ $(a \in \mathcal{A}^1,\ a^k = a + (k-1)\cdot o,\ 2 \leqslant k \leqslant \eta)$ będzie ścieżką krytyczną w grafie $G^\oplus(\pi)$. Maksymalny podciąg bezpośrednio występujących po sobie na tej ścieżce operacji wykonywanych na tej samej maszynie nazywamy **blokiem**.

Twierdzenie 3. *Jeżeli permutacja β została wygenerowana z $\pi \in \Phi$ przez zmianę kolejności operacji wewnętrznych pewnego bloku, to długość czasu cyklu*

$$T^\circ(\beta) \geqslant T^\circ(\pi).$$

Dowód. Twierdzenie dowodzi sie podobnie, jak twierdzeń blokowych dla szerokiej klasy problemów szeregowania zadań z kryterium $C_{\max}$, w szczególności dla elastycznego problemu gniazdowego [4].

5. Algorytm wyznaczania czasu cyklu

W oparciu o ideę gry w golfa wprowadzamy dwa typy ruchów przekształcających elementy przestrzeni rozwiązań problemu CFJS

$$\Gamma, \gamma :\ \Phi \to \Phi. \tag{11}$$

Pierwszy z ruchów Γ, odpowiadający mocnemu uderzeniu piłki w grze w golfa, generuje z $\pi \in \Phi$ rozwiązanie $\Gamma(\pi) \in \Phi$ znacznie się od niego różniące, powodując dywersyfikację poszukiwań. Natomiast, w przypadku ruchu typu γ, rozwiązanie $\gamma(\pi) \in \Phi$ różni się niewiele od π (odpowiada więc słabemu uderzeniu piłki).

Niech $S^{\triangleright\triangleright}$ i $S^{\triangleright}$ będą odpowiednio zbiorem ruchów mocnych oraz słabych. Jeżeli przekształcenie $\tau(\pi) = \gamma(\Gamma(\pi))$, gdzie $\Gamma \in S^{\triangleright\triangleright}$, $\gamma \in S^{\triangleright}$, $\pi \in \Phi$ to mówimy, że τ jest *ruchem złożonym* i zapisujemy $\tau = \gamma \circ \Gamma$. Wówczas zbiór

$$\mathcal{N}(\pi) = \{\tau(\pi) :\ \tau = \gamma \circ \Gamma,\ \Gamma \in S^{\triangleright\triangleright}, \gamma \in S^{\triangleright},\ \pi \in \Phi\} \tag{12}$$

jest *otoczeniem golfowym* elementu $\pi \in \Phi$ (zobacz [2]). Generując je będziemy stosowali także twierdzenie 3, tj. pominiemy ruchy zmieniające kolejność operacji wewnętrznych bloku. Otoczenie golfowe będziemy stosowali w algorytmie opartym na metodzie przeszukiwań lokalnych. W opisie algorytmu, pamięć krótkoterminowa *MEM* jest realizowana na zasadzie kolejki *FIFO*. Funkcja $ATR(\pi)$ zwraca atrybuty rozwiązania π. Algorytm kończy działanie po wykonaniu *Maxiter* iteracji.

Algorytm (AGF)

Niech $\pi \in \Phi$ będzie dowolnym rozwiązaniem startowym;

$\pi_{best} \leftarrow \pi$; $MEM \leftarrow 0$; $iter \leftarrow 0$;

repeat

Krok 1: Wygenerować otoczenie golfowe $\mathcal{N}(\pi)$ rozwiązania π,
pomijając elementy, których atrybuty znajdują się na liście *MEM*;
Krok 2: Wyznaczyć element $\beta^* \in \mathcal{N}(\pi)$ taki, że
$F(\beta^*) = \min\{F(\delta) : \delta \in \mathcal{N}(\pi)\}$;
if $F(\beta^*) < F(\pi_{best})$, then $\pi_{best} \leftarrow \beta^*$;
Krok 3: Podstaw $MEM \leftarrow MEM \cup ATR(\beta)$;
$iter \leftarrow iter + 1$;
until $iter < Maxiter$;

Implementacja algorytmu wymaga zdefiniowania:

1. zbiorów ruchów słabych oraz mocnych,
2. ustalenia atrybutów ruchów oraz zasad tworzenia i korzystania z pamięci MEM.

Jako słabe uderzenie zastosowaliśmy ruch typu wstaw (ang. *insert*). Jego wykonanie zmienia kolejność wykonywania operacji na maszynie. Z kolei silnym uderzeniem jest ruch typu transfer, przemieszczający operację na inną, z tego samego gniazda, maszynę. Oba ruchy są opisane w pracy [2].

Atrybuty generowanych rozwiązań pamiętane są w pamięci MEM w postaci trójek (s, v, k). Elementy s i v identyfikują operacje, a k maszynę. W kroku 1 algorytmu **AGF**, dla każdej trójki (s, v, k) znajdującej się w pamięci MEM, ze zbioru $\mathcal{N}(\pi)$ usuwane są rozwiązania, w których operacje s i v wykonywane są na maszynie k w kolejności s przed v. Natomiast w Kroku 3 następuje dodanie atrybutów wybranego elementu z otoczenia do pamięci MEM.

6. Strategia przeglądania otoczenia

W algorytmach opartych na przeszukiwaniu przestrzeni rozwiązań procedura wyznaczenie wartości funkcji celu ma duży wpływ na czas działania algorytmu. W przypadku rozpatrywanego w pracy problemu wyznaczenie minimalnego czasu cyklu (dla pojedynczego rozwiązania) ma złożoność obliczeniową $O(om^2)$. Zamiast więc obliczania dokładnej wartości czasu cyklu, będziemy korzystali ze znacznie szybciej wyznaczanego dolnego oszacowania.

Dowolny ruch złożony τ może być jednoznacznie reprezentowany przez trójkę $v = (i, k, s)$ $(i \in \mathcal{O}, k \in \mathcal{M}, s = 1, 2, \ldots, n)$. Jego wykonanie możemy podzielić na dwa etapy: (*1*) usunięcie operacji i, (*2*) wstawienie operacji i na pozycję s na maszynie k.

Niech $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ będzie permutacją wygenerowaną z π pierwszej etapie, a $\beta = (\beta_1, \beta_2, \ldots, \beta_m)$ w etapie drugim. W pierwszej składowej $\mathcal{H}^1(\pi)$ grafu cyklicznego $G^{\oplus}(\alpha)$ definujemy pewne drogi (dokładniej - długości tych dróg):

1. $R^l(j)$ $(l \in \mathcal{M}, j \in \mathcal{O})$ - długość najdłuższej drogi z wierzchołka $\alpha_l(1)$ do wierzchołka j,
2. $Q^l(j)$ oznaczamy długość najdłuższej drogi od wierzchołka j do wierzchołka reprezentującego operacją $\alpha_l(n_l)$.

Wartość wyrażenia

$$LB^l(\pi, v) = \max\{R^l(\alpha_k(s-1)), R^l(i) - p_i\} + \max\{Q^l(\alpha_k(s)), Q^l(i) - p_i\} + p_i\},$$

jest długością najdłuższej drogi w grafie $G^{\oplus}(\beta)$, z wierzchołka $\alpha_l(1)$ do $\alpha_l(n_l)$ przedzącej przez wierzchołek i.

Lemat 1. *Jeżeli permutacja β została wygenerowana z $\pi \in \Phi$ przez wykonanie ruchu złożonego $v = (i, k, s)$, to minimalny czas cyklu*

$$T^o((\beta) \geqslant LB(\alpha) = \max_{l \in \mathcal{M}} LB^l(\alpha).$$

7. Eksperymenty obliczeniowe

Głównym celem przeprowadzonych badań było wyznaczenie przyspieszenia procesu obliczeń algorytmu *AGF* wykorzystującego dolne oszacowanie długości czasu cyklu (AGF_{LB}) względem algorytmu wyznaczającego wartość dokładną. Oba algorytmy zostały zaprogramowane w języku C++ w środowisku Visual Studio 2010. Obliczenia wykonano na komputerze z procesorem Intel I7-core 2.4GHz. Punktem startowym była permutacja naturalna, liczba iteracji wynosi 10 000, długość pamięci krótkoterminowej 15. Przy każdym uruchomieniu algorytmu wyznaczono najlepsze rozwiązanie (tj. przybliżoną wartość czasu cyklu), oraz czas obliczeń.

W tabeli 1 przedstawiono wyniki obliczeniowe algorytmów dla przykładów z pracy Barnesa i Chambersa [1]. W pierwszej kolumnie jest nazwa przykładu a dalej, liczba zadań (*n*) oraz liczba maszyn (*m*). W następnej - wartości referencyjne, tj. rozwiązania elestycznego problemu gniazdowego z kryterium $C_{\max}$ (wartość ta jest górnym ograniczeniem optymalnego czasu cyklu). Kolejna kolumna zawiera wyznaczone przez algorytmy wartości czasu cyklu cyklu T^* (oba algorytmy wyznaczały takie same rozwiązania). Ostatnie dwie kolumny zawierają czasy obliczeń algorytmów.

Wyznaczone przez oba algorytmy czasy cyklu (kolumna T^*) były identyczne. Są przy tym istotnie mniejsze od wartości górnego ograniczenia $C_{\max}$. Zależność ta występuje w 11 z 14 przykładów. W dwóch przypadkach wyznaczona długość czasu cyklu nie jest liczbą całkowitą. Z tego wynika, że ścieżka krytyczna przechodzi przez dwie lub więcej składowych grafu cyklicznego.

Czas działania algorytmu AGF_{LB} jest znacznie krótszy od czasu działania algorytmu AGF (algorytm AGF działa od 3 do aż do około 25 razy dłużej od algorytmu AGF_{LB}). Biorąc pod uwagę także fakt, że najlepsze rozwiązania były wyznaczane po wykonaniu niewielkiej liczby iteracji stwierdzamy, że algorytm ten może być z powodzeniem stosowany do rozwiązywania praktycznych przykładów o dużych rozmiarach.

8. Podsumowanie

W pracy rozpatrywano cykliczny elastyczny problem gniazdowy. Przedstawiono model grafowy, dla ustalonej kolejności wykonywania operacji na poszczególnych maszynach. W oparciu o analizę ścieżek w tym grafie, udowodniono twierdzenia umożliwiające efektywne wyznaczenie minimalnej wartości czasu cyklu oraz jego dolnego oszacowania. W metaheurystyce zastosowano tzw. otoczenie golfowe, umożliwiające skuteczną zarówno intensyfikację jak i dywersyfikację procesu przeszukiwania przestrzeni rozwiązań a także, unikalną strategię wyznaczania rozwiązania o minimalnej wartości czasu cyklu znacząco przyspieszającą obliczenia. Na podstawie otrzymanych

Tabela 1

Wyniki obliczeniowe algorytmów AGF i AGF_{LB}.

przykład	$n \times m$	o	$C_{\max}$	T^*	$t_{AGF_{LB}}$	t_{AG}
setb4c9	15×11	150	914	910.50	37.6	187.6
setb4cc	15×12	150	907	886	45.8	299.0
setb4x	15×11	150	925	876	35.8	191.0
setb4xx	15×12	150	925	883	42.8	231.1
setb4xxx	15×13	150	925	873	51.1	298.8
setb4xy	15×12	150	910	845*	21.4	65.3
setb4xyz	15×13	150	903	838*	4.7	47.5
seti5c12	15×16	225	1174	1126	83.6	577.9
seti5cc	15×17	225	1136	1468	97.4	829.0
seti5x	15×16	225	1198	1105	80.7	510.2
seti5xx	15×17	225	1197	1115	91.4	673.5
seti5xxx	15×18	225	1197	1363.50	298.7	1367.4
seti5xy	15×17	225	1136	1468	101.6	794.6
seti5xyz	15×18	225	1125	1052	103.2	2514.7

* - rozwiązanie optymalne

wyników można stwierdzić, że przedstawiony algorytm w krótkim czasie wyznacza akceptowane w praktyce rozwiązania.

LITERATURA

1. Barnes, J. W., Chambers, J. B.: Flexible job shop scheduling by tabu search, Graduate program in operations research and industrial engineering, The University of Texas at Austin, Technical Report Series: ORP96-09 (1996).
2. Bożejko W., Uchronski M., Wodecki M.: The new golf neighborhood for the flexible job shop problem, Procedia Computer Science, 1, 2010, p. 289–296.
3. Bożejko, W., Pempera, J., Wodecki, M.: The golf algorithm for the cyclic flexible job shop problem, w redakcji.
4. Bożejko, W., Uchronski, M., Wodecki, M.: Parallel hybrid metaheuristics for the flexible job shop problem, Computers and Industrial Engineering, 59, 2010, p.323–333.
5. Brucker P., Kampmeyer T.: Cyclic job shop scheduling problems with blocking, Annals of Operations Research, 159, 2008, p. 161–181.
6. Kampmeyer T.: Cyclic Scheduling Problems, Ph. D. Thesis, University Osnabriick (2006).
7. Smutnicki C., Smutnicki A.: Harmonogramowanie cykliczne w systemie gniazdowym, Zastosowanie teorii systemów, AGH, 2007, p. 105 – 115.