AUTOMATYKA • 2007 • Tom 11 • Zeszyt 1–2

Czesław Smutnicki*, Adam Smutnicki*

Nowe własności harmonogramów cyklicznych w systemie przepływowym

1. Wprowadzenie

Systemy produkcyjne realizujące wytwarzanie wieloasortymentowe, wielkoseryjne przy wolnozmiennym w czasie asortymencie wyrobów mogą zaspokajać zapotrzebowanie rynku bądź poprzez: (1) nieokresową zmianę jednolitego asortymentu produkcji, bądź też poprzez (2) cykliczne dostarczanie na wyjście *mieszanki* asortymentów. Skład ilościowy i jakościowy tej mieszanki zależy od średnio- i długoterminowych zamówień klientów. Zakładając techniczną realizowalność obu powyższych rozwiązań, można stwierdzić, że to drugie podejście jest bardziej atrakcyjne, bowiem eliminuje lub ogranicza magazyn wyrobów gotowych. Dodatkowo minimalizując czas cyklu, dla ustalonego zestawu zadań w cyklu zwiększamy wydajność systemu oraz stopień wykorzystania maszyn. Dalszą poprawę efektywności funkcjonowania systemu uzyskuje się poprzez eliminację lub ograniczenie magazynowania półproduktów w magazynach pośrednich lub pomiędzy stanowiskami. Prowadzi to do warunków znanych w literaturze jako *no store* (zakaz składowania międzystanowiskowego) lub *limited store* (ograniczony obszar składowania międzystanowiskowego, tzw. bufor), dość często wymienianych w kontekście systemów JIT oraz częstokroć będących koniecznością wynikającą z pracy ciągłej sytemu.

Systemy z ograniczonymi pojemnościami buforów lub ich brakiem są obiektem zainteresowań wielu badaczy ze względu na ich silne znaczenie praktyczne oraz problemy z uzyskaniem skutecznych algorytmów rozwiązywania, [1–10]. Zdecydowana większość otrzymanych dotychczas wyników dotyczy niecyklicznych systemów tego typu, [1–2, 4, 6–10]. Deterministyczne problemy cykliczne z ograniczeniami składowania należą do jednych z trudniejszych zagadnień optymalizacji dyskretnej. Świadczy o tym stosunkowo niewielka liczba pozycji literaturowych oraz kłopotliwe numerycznie procedury wyznaczania długości czasu cyklu i/lub harmonogramu [3, 5].

Silna NP-trudność już wielu najprostszych badanych wersji problemu, ogranicza zakres stosowania algorytmów dokładnych do instancji o małej liczbie zadań. Z tego powodu, do wyznaczania satysfakcjonujących rozwiązań, stosuje się powszechnie szybkie algoryt-

* Instytut Informatyki Automatyki i Robotyki, Politechnika Wrocławska

my przybliżone oparte na technikach przeszukiwań lokalnych. Prawie wszystkie metody tego typu opierają się na dwupoziomowej dekompozycji problemu:

1) wyznaczenie optymalnej kolejności zadań (poziom górny),
2) wielokrotne wyznaczenie minimalnej wartości kryterium dla danej kolejności zadań (poziom dolny).

O ile dla „klasycznych" problemów szeregowania rozwiązanie problemu dolnego poziomu sprowadza się do analizy specyficznego grafu możliwej do wykonania w sposób efektywny czasowo, to w przypadku postawionego zagadnienia rozwiązanie (wielokrotne) zagadnienia dolnego poziomu jest stosunkowo czasochłonne bowiem, w ogólności, wymaga rozwiązania pewnego zagadnienia PL. Stąd wszelkie własności szczególne, w tym pozwalające na bardziej efektywne wyliczanie czasu cyklu i poszukiwanie harmonogramu oraz ograniczenie liczności lokalnie przeglądanego sąsiedztwa i/lub przyspieszenie szybkości jego przeglądania są bardzo pożądane.

2. Opis problemu

Rozważany jest system wytwórczy o strukturze szeregowej złożony z m stanowisk (maszyn o jednostkowej przepustowości) indeksowanych kolejno 1, 2, ..., m. W systemie tym należy wykonywać cyklicznie (w sposób powtarzalny) n, niekoniecznie różnych, zadań danych zbiorem $J = \{1, 2, ..., n\}$. Pojedyncze zadanie odpowiada wytworzeniu jednej sztuki produktu finalnego (lub półproduktu). Wykonanie zadania odbywa się w m etapach, w sposób jednakowy dla wszystkich zadań, lecz w różnym czasie zależnym od zadania. Etap i wykonywany jest przez maszynę o numerze i, $i = 1, ..., m$. Ze względu na strategię wytwarzania, tj. brak możliwości składowania wyrobów podczas produkcji w magazynach i buforach międzystadialnych, zadanie zakończone na maszynie i, którego nie można przekazać na maszynę i+1 ze względu na jej zajętość, musi pozostawać na i (blokować ją) aż do chwili zwolnienia i+1. Zadanie $j \in J$ jest interpretowane jako sekwencja m operacji $O_{1j}, O_{2j}, ..., O_{mj}$ wykonywanych na maszynach 1, 2, …, m w takiej kolejności. Operacja O_{ij} odpowiada wykonywaniu zadania j na i-tej maszynie w czasie $p_{ij} > 0$. W danej chwili maszyna może wykonywać tylko jedno zadanie, oraz zadanie może być wykonywane tylko na jednej maszynie.

Zestaw zadań wykonywanych w cyklu może być dany arbitralnie lub wynikać z polityki jednostajnego i równoczesnego dostarczania wszystkich zamówionych partii produkcyjnych (tzw. mieszanki produktów wyjściowych). W tym drugim przypadku przyjmuje się, że złożone zostało zamówienie na produkty pochodzące ze zbioru $J^* = \{1, 2, \ldots, t\}$ w ilościach $r_1, \ldots, r_t$, odpowiednio. Zatem zachodzi potrzeba zrealizowania w systemie $N = \sum_{i=1}^{t} r_i$ zadań niekoniecznie różnych. Niech $c > 1$ będzie wspólnym dzielnikiem liczb $r_1, \ldots, r_t$. Zbiór J^* dzielimy na c identycznych podzbiorów nazywanych MPS (*Minimal Part Set*), z których każdy zawiera $n = r_1/c + \ldots + r_t/c$ zadań. MPS-y są przetwarzane jeden po drugim w sposób cykliczny, dostarczając mieszankę produktów w ilościach odpowiednio $r_1/c, \ldots, r_t/c$ na cykl. Dalej będziemy się zajmować wyłącznie pojedynczym MPS-em.

Przyjmujemy, że zadania ze zbioru J wykonywane są w *określonej kolejności*, takiej samej dla wszystkich MPS-ów, co zasadniczo implikuje cykliczność sekwencji, ale nieko-

niecznie harmonogramów czasowych. W pracy [1] pokazano, że w systemie przepływowym z ograniczeniami „bez magazynowania" dopuszczalna kolejność wykonywania zadań na maszynach musi być identyczna na każdej maszynie. Zatem, kolejność wykonywania zadań w każdym MPS-ie możemy opisać za pomocą jednej permutacji $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ elementów zbioru $\{1, \ldots, n\}$.

Harmonogram czasowy wykonywania zadań k-tego MPS-a (dla danej kolejności ich realizacji π) opisujemy przy pomocy macierzy $[S^k]_{m\times n}$, gdzie $S^k_{i,j}$ oznacza termin rozpoczęcia wykonywania zadania j na maszynie i, patrz także [5]. Harmonogram ten musi spełniać następujące ograniczenia:

$$S^k_{i,\pi(j)} + p_{i,\pi(j)} \le S^k_{i+1,\pi(j)}, \quad i = 1, \ldots, m-1, \quad j = 1, \ldots, n \tag{1}$$

$$S^k_{i,\pi(j)} + p_{i,\pi(j)} \le S^k_{i,\pi(j+1)}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n-1 \tag{2}$$

$$S^k_{i+1,\pi(j)} \le S^k_{i,\pi(j+1)}, \quad i = 1, \ldots, m-1, \quad j = 1, \ldots, n-1 \tag{3}$$

$$S^k_{i,\pi(n)} + p_{i,\pi(n)} \le S^{k+1}_{i,\pi(1)}, \quad i = 1, \ldots, m \tag{4}$$

$$S^k_{i+1,\pi(n)} \le S^{k+1}_{i,\pi(1)}, \quad i = 1, \ldots, m-1 \tag{5}$$

dla $k = 1, 2, \ldots$ Takie postawienie problemu, choć dopuszcza pewną swobodę w konstrukcji harmonogramu, równocześnie komplikuje jego wyznaczenie, bowiem warunki (4) oraz (5) wymuszają zależną analizę MPS-ów. Załóżmy zatem dalej, że nie tylko kolejność zadań jest cyklicznie powtarzana, ale także że harmonogram czasowy pracy systemu jest w pełni cykliczny. Oznacza to istnieje stała T (okres) taka, że

$$S^{k+1}_{i,\pi(j)} = S^k_{i,\pi(j)} + T, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n, \quad k = 1, 2, \ldots \tag{6}$$

Okres T zależy od π i jest nazywany **czasem cyklu** systemu. Minimalną wartość T, dla ustalonej π, będziemy nazywać **minimalnym czasem cyklu** i oznaczać przez $T(\pi)$. Zauważmy, że podstawiając (6) do (1)–(5) otrzymamy warunki odnoszące się tylko do harmonogramu k-tego MPS-a oraz, że wszystkie MPS-y dla $k = 1, 2, \ldots$ mają taką samą postać. Zatem wystarczy skonstruować harmonogram dla **jednego** MPS-a i dokonać jego translacji o wielkość kT, $k = 1, 2, \ldots$ na osi czasu. Ponieważ ograniczenia (1)–(5) są identyczne dla wszystkich k, to możemy pominąć górny indeks k w oznaczeniach. Wartość $T(\pi)$ oraz harmonogram S_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, n$ każdego cyklu można wyznaczyć poprzez rozwiązanie zadania postaci:

$$T(\pi) = \min_{T, S_{ij}} T \tag{7}$$

$$S_{i,\pi(j)} + p_{i,\pi(j)} \le S_{i+1,\pi(j)}, \quad i = 1, \ldots, m-1, \quad j = 1, \ldots, n \tag{8}$$

$$S_{i,\pi(j)} + p_{i,\pi(j)} \le S_{i,\pi(j+1)}, \quad i = 1, ..., m, \quad j = 1, ..., n-1 \tag{9}$$

$$S_{i+1,\pi(j)} \le S_{i,\pi(j+1)}, \quad i = 1, ..., m-1, \quad j = 1, ..., n-1 \tag{10}$$

$$S_{i,\pi(n)} + p_{i,\pi(n)} \le S_{i,\pi(1)} + T, \quad i = 1, ..., m \tag{11}$$

$$S_{i+1,\pi(n)} \le S_{i,\pi(1)} + T, \quad i = 1, ..., m-1 \tag{12}$$

Bez straty ogólności możemy przyjąć, że punktem zakotwiczenia harmonogramu jest $S_{1,\pi(1)} = 0$.

Wprowadzone oznaczenia pozwalają na eleganckie sformułowanie obu poziomów dekomponowanego problemu optymalizacji. Na poziomie górnym należy wyznaczyć permutację $\pi^* \in \Pi$, gdzie Π jest zbiorem wszystkich permutacji, taką że

$$T(\pi^*) = \min_{\pi \in \Pi} T(\pi) \tag{13}$$

Na poziomie dolnym należy, dla danej π, wyznaczyć $T(\pi)$. Poszukiwanie π^* można zrealizować w technice SA, TS czy GA, poprzez analogię do innych problemów posiadających zbiór rozwiązań opartych na permutacjach zbioru n-elementowego.

3. Problem dolnego poziomu

Realizacja (7)–(12) poprzez rozwiązanie problemu PL wydaje się wyjątkowo kosztowna obliczeniowo, bowiem w technice SA czy też TS wymagane jest wielokrotne jego rozwiązywanie. W pracy [3] pokazano jak wyznaczyć $T(\pi)$ oraz harmonogram S_{ij}, $i = 1, ..., m$, $j = 1, ..., n$ poprzez rozwiązanie problemu minimalnego przepływu w specyficznej sieci. Biorąc pod uwagę rozmiar sieci $O(nm)$ oraz złożoność algorytmu wyznaczania przepływu, otrzymamy i w tym przypadku względnie kosztowną obliczeniowo metodę o złożoności $O(n^3m^3)$. W pracy [3] podano także interpretację T za pomocą ścieżek w pewnym grafie cylindrycznym, lecz równocześnie nie zaproponowano ani algorytmu wyznaczania $T(\pi)$, ani dowodu poprawności modelu cylindrycznego. Poniżej proponujemy dwie oryginalne metody, pierwszą pozwalającą na wyznaczenie $T(\pi)$ w czasie $O(nm^2)$ na bazie typowego grafu siatkowego dla problemu przepływowego z buforowaniem [9, 10, 4], oraz drugą pozwalającą na wyznaczenie harmonogramu S_{ij}, $i = 1, ..., m$, $j = 1, ..., n$, przy znajomości $T(\pi)$, również w czasie $O(nm^2)$ na bazie grafu cylindrycznego. Obie sprowadzają się do analizy różnych grafów i bazują na kilku własnościach wprowadzonych poniżej.

Własność 1

$$T \ge T(\pi) \ge T^* \overset{def}{=} \max_{1 \le i \le m} T_i \tag{14}$$

gdzie

$$T_i \overset{def}{=} \begin{cases} \max\{S_{i,\pi(n)} + p_{i,\pi(n)}, S_{i+1,\pi(n)}\} - S_{i,\pi(1)} & 1 \le i < m \\ S_{i,\pi(n)} + p_{i,\pi(n)} - S_{i,\pi(1)} & i = m \end{cases} \tag{15}$$

Dowód

Bezpośrednio z warunków (11) i (12) otrzymujemy

$$S_{i,\pi(n)} + p_{i,\pi(n)} - S_{i,\pi(1)} \le T, \quad i = 1, ..., m \tag{16}$$

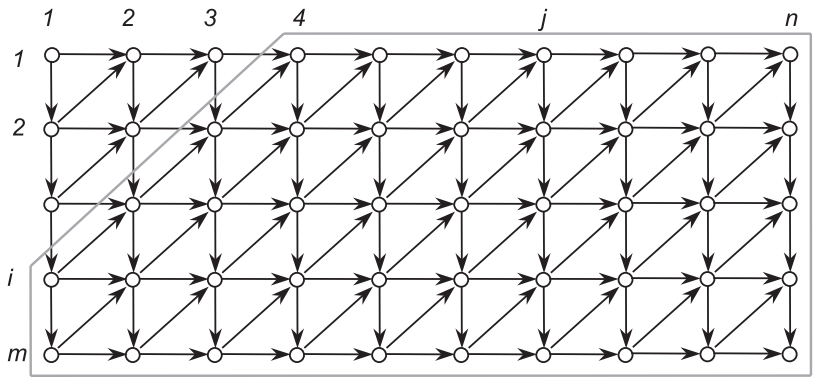
oraz

$$S_{i+1,\pi(n)} - S_{i,\pi(1)} \le T, \quad i = 1, ..., m-1 \tag{17}$$

dla każdego T, w tym także dla $T(\pi)$. Uwzględniając definicję (15) oraz łącząc warunki z (16), (17), otrzymujemy własność (14).

Wielkość T_i szacuje minimalny odstęp czasowy dla zdarzeń wzajemnie zależnych, powiązanych ze sobą za pomocą zbioru warunków (8)–(10). Wyznaczenie T_i jest możliwe przy zastosowaniu grafu oraz własności opisanych poniżej.

Dla danej kolejności π, budujemy graf $G(\pi) = (V, R \cup E \cup F)$, w formie prostokątnego grafu siatkowego zawierającego m wierszy i n kolumn, patrz interpretacja w [9–10] oraz rysunek 1.



Rys. 1. Graf siatkowy

Węzły zbioru V reprezentują wszystkie operacje zadań za zbioru J, zaś zbiory łuków R, E, F odpowiadają kolejności realizacji operacji i są definiowane odpowiednio do warunków (8)–(10):

$$V = \{1, 2, ..., m\} \times \{1, 2, ..., n\} \tag{18}$$

$$R = \bigcup_{i=1}^{m-1} \bigcup_{j=1}^{n} \{((i, j), (i+1, j))\} \tag{19}$$

$$E = \bigcup_{i=1}^{m} \bigcup_{j=1}^{n-1} \{((i, j), (i, j+1))\} \tag{20}$$

$$F = \bigcup_{i=1}^{m-1} \bigcup_{j=1}^{n-1} \{((i+1, j), (i, j+1))\} \tag{21}$$

Węzeł $(i, j) \in V$ reprezentuje i-tą operację zadania $\pi(j)$, jego waga wynosi p_i, $\pi_{(j)}$. Łuki ze zbioru R oraz E są obciążone wagą równą zero. Każdy łuk $((i+1, j), (i, j+1) \in F$ posiada wagę minus p_{i+1}, $\pi_{(j)}$ i odpowiada przekształconemu warunkowi (10) do postaci

$$(S_{i+1,\pi(j)} + p_{i+1,\pi(j)}) - p_{i+1,\pi(j)} \le S_{i,\pi(j+1)}, \quad i = 1, ..., m-1, \quad j = 1, ..., n-1 \tag{22}$$

Graf $G(\pi)$ jest grafem o regularnej strukturze niezależnej od π, posiadającym obciążenia węzłów i łuków z F zależne od π. Zauważmy dalej, że zdarzenia związane z czynnością reprezentowaną węzłem $(i, 1)$ mają wpływ tylko na pewien podzbiór czynności, będących następnikami niekoniecznie bezpośrednimi w grafie $G(\pi)$, patrz obszar ograniczony kolorem szarym w rysunku 1.

Niech d^i_{kj} oznacza długość najdłuższej drogi w $G(\pi)$ dochodzącej do węzła (k, j) (wraz z obciążeniem tego węzła) dla i ustalonego w (15). Przyjmując jako źródło węzeł $(i, 1)$, otrzymujemy z algorytmu Bellmana następujący wzór rekurencyjny:

$$d^i_{kj} \overset{def}{=} \begin{cases} \max\{d^i_{k-1,j}, d^i_{k,j-1}, d^i_{k+1,j-1} - p_{k+1,\pi(j-1)}\} + p_{k,\pi(j)} & 1 \le k < m \\ \max\{d^i_{k-1,j}, d^i_{k,j-1}\} + p_{k,\pi(j)} & k = m \end{cases} \tag{23}$$

gdzie $p_{k\pi(0)} = 0, k = 1, ..., m.$ Aby respektować własności źródła, przyjmujemy $d^i_{k0} = -L, k = 1, ..., i-1,$ $d^i_{0j} = -L,$ $j = 1, ..., i-1,$ $d^i_{i0} = 0,$ $d^i_{0i} = 0,$ gdzie L jest pewną dużą liczbą.

Własność 2

$$T_i = \begin{cases} \max\{d^i_{i,n}, d^i_{i+1,n} - p_{i+1,\pi(n)}\} & 1 \le i < m \\ d^i_{i,n} & i = m \end{cases} \tag{24}$$

Dowód

Niech i będzie ustalone, $1 \le i \le m.$ Z konstrukcji grafu wynika, że

$$S_{k,\pi(j)} = d^i_{k,j} - p_{k,\pi(j)} \tag{25}$$

są dopuszczalnymi w sensie warunków (8)–(10) oraz najwcześniejszymi możliwymi terminami rozpoczęcia tych czynności, które są następnikami czynności $(i, \pi(1))$ reprezentowanej węzłem $(i, 1)$. Ponieważ zgodnie z (23) mamy $d^i_{i,1} = \max\{-L, 0, 0-0\} + p_{i,\pi(1)}$ zatem $S_{i,\pi(1)} = 0.$ Podstawiając (25) do (15) otrzymujemy (24).

Własność 3

T^* można wyznaczyć w czasie $\theta(nm^2)$.

Dowód

Najbardziej czasochłonny obliczeniowo wzór (23) wymaga $\theta(nm^2)$ czasu.

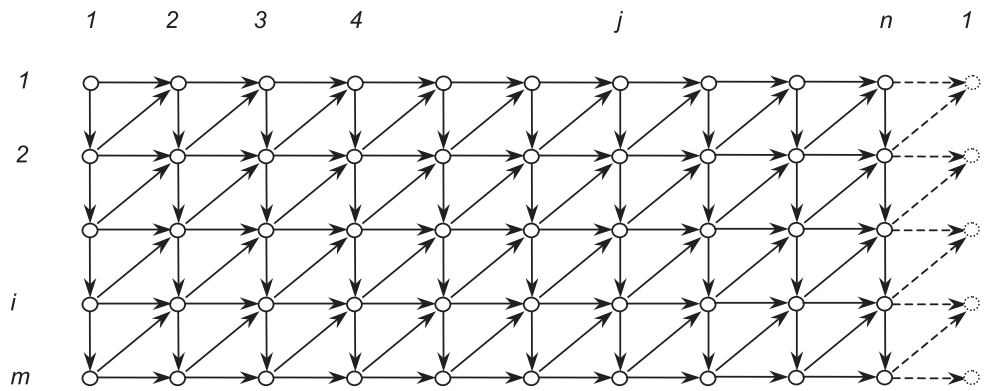
Własność 4

Istnieje rozwiązanie optymalne problemu (7)–(12) takie, że $T(\pi) = T^*$.

Dowód

Pokażemy wpierw jak skonstruować takie rozwiązanie, tzn. S_{ij}, $i = 1, ..., m$, $j = 1, ..., n$. Następnie wykażemy jego dopuszczalność, bowiem jego optymalność wynika natychmiast z własności 1.

Dla danej kolejności π, budujemy graf nieplanarny $H(\pi) = (V, R \cup E \cup E^* \cup E \cup F^*)$, w formie prostokątnego grafu siatkowego zawierającego m wierszy i n kolumn (identycznie jak dla $G(\pi)$), lecz zawiniętego na cylindrze (rys. 2). Definicje i interpretacje zbiorów V, R, E, F są takie same jak dla grafu $G(\pi)$, patrz wzory (18)–(21), takie same są również obciążenia węzłów i krawędzi.



Rys. 2. Graf cylindryczny

Zbiory E^*, F^* „zamykające" cylinder (zaznaczono linią przerywaną na rys. 2) są zdefiniowane następująco:

$$E^* = \bigcup_{i=1}^{m} \{((i, n), (i, 1))\} \tag{26}$$

$$F^* = \bigcup_{i=1}^{m-1} \{((i+1, n), (i, 1))\} \tag{27}$$

Każdy łuk $((i, n), (i, 1) \in E^*$ posiada wagę ujemną – T^* i reprezentuje ograniczenie (11) przekształcone do postaci

$$(S_{i,\pi(n)} + p_{i,\pi(n)}) - T^* \leq S_{i,\pi(1)} \tag{28}$$

Każdy łuk $((i+1, n), (i, 1) \in F^*$ posiada wagę ujemną ($-p_{i+1}$, $\pi_{(j)} - T^*$) i reprezentuje ograniczenie (12) przekształcone do postaci

$$(S_{i+1,\pi(n)} + p_{i,\pi(n)}) - p_{i,\pi(n)} - T^* \le S_{i,\pi(1)} \tag{29}$$

Z węzłem (i, j) kojarzymy zdarzenie $S_{i,\pi(j)}$ mające sens długości najdłuższej drogi dochodzącej do tego węzła (bez obciążenia tego węzła), i = 1, ..., m, j = 1, ..., n. Ponieważ graf $H(\pi)$ zawiera cykle (obiegają one wokół cylindra) oraz obciążenia dodatnie i ujemne na węzłach i krawędziach, zatem do wyznaczenia długości dróg należy użyć algorytmu Bellmana–Forda. Wychodząc od oryginalnej wersji tego algorytmu, formułowanej dla szukania najkrótszych dróg w grafie z cyklami i dowolnymi obciążeniami, dokonamy modyfikacji pozwalającej na szukanie dróg maksymalnych. Formalnie algorytm ten, dla grafu o s wierzchołkach i t krawędziach, wykonuje $O(s)$ cykli, z których każdy polega na analizie wszystkich t krawędzi grafu w celu aktualizacji długości dróg dochodzących do wierzchołków. Biorąc pod uwagę rozmiar i budowę grafu $H(\pi)$, otrzymalibyśmy algorytm o złożoności $O(n^2m^2)$, przy założeniu, że krawędzie grafu analizowane są w *dowolnej* kolejności. Poniżej podamy oryginalny bardziej efektywny algorytm o złożoności $O(nm^2)$.

Jako źródło harmonogramu wybieramy $S_{1,\pi(1)} = 0$, inicjujemy $S_{i,\pi(1)} = S_{i-1,\pi(1)} + p_{i-1,\pi(1)}$, dla i = 2, 3, ..., m, a następnie powtarzamy m-krotnie cykl obliczeniowy składający się z wzorów (30) oraz (31):

$$S_{i\pi(j)} = \begin{cases} \max\{S_{i,\pi(j-1)} + p_{i,\pi(j-1)}, S_{i+1,\pi(j-1)}\} & i = 1 \\ \max\{S_{i,\pi(j-1)} + p_{i,\pi(j-1)}, S_{i-1,\pi(j)} + p_{i-1,\pi(j)}, S_{i+1,\pi(j-1)}\} & 1 < i < m,\ j = 2, 3, ..., n \\ \max\{S_{i-1,\pi(j)} + p_{i-1,\pi(j)}, S_{i,\pi(j-1)} + p_{i,\pi(j-1)}\} & i = m \end{cases} \tag{30}$$

$$S_{i\pi(1)} = \begin{cases} \max\{S_{i,\pi(n)} + p_{i,\pi(n)} - T^*, S_{i+1,\pi(n)} - T^*\} & i = 1 \\ \max\{S_{i,\pi(n)} + p_{i,\pi(n)} - T^*, S_{i-1,\pi(1)} + p_{i-1,\pi(1)}, S_{i+1,\pi(n)} - T^*\} & 1 < i < m,\ \mathrm{j} = 1 \\ \max\{S_{i-1,\pi(1)} + p_{i-1,\pi(1)}, S_{i,\pi(n)} + p_{i,\pi(n)} - T^*\} & i = m \end{cases} \tag{31}$$

W ostatnim cyklu obliczeniowym nie realizujemy (31). Realizacja (30) i (31) odpowiada zastosowaniu algorytmu Bellmana–Forda, jeden cykl odpowiada propagacji aktualizacji $S_{i,\pi(j)}$ wzdłuż π. Jak łatwo sprawdzić, w każdym cyklu warunki dopuszczalności (8)–(10) są spełnione ze względu na postać (30). Jedyne naruszenia ograniczeń mogą się odnosić do (11)–(12). Ich naruszanie zależy od cyklu (w cyklu k-tym warunki (11)–(12) dla i = 1, 2, .., k–1 są spełnione) oraz od poprawności użycia T^* (osiągalności oszacowania). Ponieważ T^* jest dolnym ograniczeniem dla $T(\pi)$, zatem wystarczy wykazać, że $H(\pi)$ nie posiada cyklu o dodatniej długości, uniemożliwiającej wyznaczenie rozwiązania dopuszczalnego.

Tę część dowodu poprowadzimy przez zaprzeczenie. Przypuśćmy, że $H(\pi)$ posiada cykl o dodatniej długości. Cykl ten musi obiegać cylinder bowiem $G(\pi)$ był acykliczny. Bez

straty ogólności rozważań możemy przyjąć, że cykl przechodzi przez węzeł (i, 1) dla pewnego i, $1 \leq i \leq m$. Zatem z własności 2 (T_i jako droga w grafie) mamy $T_i - T^* > 0$, co daje $T_i > T^*$ i przeczy własności 1, że T^* było maksymalne.

W praktyce, liczba cykli opartych na wzorach (30)–(31) potrzebna do ustalenia się wielkości $S_{i,\pi(j)}$ jest dużo mniejsza niż m. Stąd otrzymujemy już bez dowodu.

Własność 5

Rozwiązanie optymalne problemu (7)–(12) można znaleźć w czasie $O(nm^2)$.

4. Eksperyment komputerowy

Celem przeprowadzonego eksperymentu komputerowego było przebadanie jakości różnych metod lokalnej optymalizacji stosowanych w dwu-poziomowych schemacie przybliżonej optymalizacji. Rozważono algorytm genetyczny z pracy [5] z nowym mechanizmem ekspresji genów, algorytm symulowanego wyżarzania z auto-strojeniem i logarytmicznym schematem studzenia [11], podstawowy wariant metody tabu (bez wykorzystania własności blokowych i akceleratora).

Algorytmy zostały zaprogramowany w języku C++ i testowane był na pierwszych dziewięciu grupach przykładów należących do zestawu zaproponowanego przez Taillarda [12]. Każda z tych grup składa się z 10 trudnych instancji o tej samej liczbie maszyn m i tej samej liczbie zadań n. Algorytm genetyczny zrealizowano dokładnie tak jak w pracy [5] w wersji GA-E (z pełnym operatorem ekspresji genów). Każdy z algorytmów zależnych od wartości losowych (tj. GA i SA) został uruchomiony 10 razy dla każdej instancji testowej, z wyjątkiem TS, który działa deterministycznie.

Dla każdej instancji problemu, dla każdego z testowanych algorytmów oraz dla każdego z 10 uruchomień wyznaczono:

T^* – najmniejszy czas cyklu uzyskany podczas badań algorytmów dla tej instancji,

$B_i^A = 100\% \cdot (T(\pi_i^A) - T^*)/T^*$ – błąd rozwiązania π_i^A otrzymanego podczas i-tego uruchomienia algorytmu A.

Bazując na wyliczonych wartościach błędów dla każdej instancji i dla każdego algorytmu wyznaczono:

AB – średni błąd,

$MINB$ – minimalny błąd,

$MAXB$ – maksymalny błąd.

Ostatecznie dla każdego algorytmu i każdej grupy przykładów wyznaczono wartości średnie wyżej wymienionych wielkości. Wyniki przedstawiono w tabeli 1.

Tabela 1
Porównanie efektywności metod lokalnych poszukiwań

Grupa	GA-E			SA			TS		
	MINB	AB	TIME	MINB	AB	TIME	MINB	ITER	TIME
20×5	0,75	1,92	3,5	0,5	2,43	0,5	0,68	40	3,8
20×10	0,43	1,73	13,2	0,41	1,84	2,3	0,53	300	14,8
20×20	0,30	1,31	48,6	0,22	1,57	9,1	0,54	250	51,7
50×5	2,94	4,21	13,7	0	2,16	4,4	3,5	50	18,8
50×10	2,58	3,93	30,9	0	1,52	14,4	3,07	50	50,1
50×20	2,02	2,94	115,2	0	1,32	61,1	2,61	50	171,2
100×5	5,86	6,94	29,1	0	1,93	20,6	18,53	10	48,5
100×10	4,88	5,88	61,0	0	1,62	58,4	12,1	10	97,8
100×20	3,01	3,77	228,8	0	1,1	223,9	7,09	10	288,7

5. Uwagi i wnioski

Konsekwencje wprowadzonych i wykazanych własności są wielorakie. Własność 1 pozwala znaleźć wartość minimalnego czasu cyklu $T(\pi)$ bez konieczności wyznaczania dokładnego harmonogramu S_{ij}, $i = 1, ..., m$, $j = 1, ..., n$, co może być skutecznie wykorzystywane w metodach SA, TS, GA. Faktycznie, wyznaczenie szczegółowego harmonogramu jest potrzebne tylko dla kolejności generowanej na wyjściu algorytmu lokalnej optymalizacji. Własność 3 mówi, że $T(\pi)$ można wyznaczyć efektywniej niż kiedykolwiek dotychczas. Po drugie, własność 2, poprzez swój związek z długości drogi w grafie pozwala na wprowadzenie własności blokowych znanych dla problemów z buforowaniem [9–10, 4], wyjątkowo korzystnych dla konstrukcji efektywnego otoczenia w metodzie TS oraz przy budowie tzw. *akceleratora* przyśpieszającego proces obliczeniowy. Własność 4 pozwala znaleźć szczegółowy harmonogram cyklu efektywniej niż kiedykolwiek dotychczas. W końcu, uporządkowanie i usystematyzowania podejścia pozwala na skuteczne zaatakowanie trudniejszych problemów tego typu, takich jak np. problem o strukturze zadaniowej (*job-shop*).

Z analizy wyników prezentowanych w tabeli 1 wynika, że dla rozważanego problemu algorytm SA dostarcza najlepszych wyników spośród testowanych GA, SA, TS. Algorytm TS, głównie ze względu na znaczny czas działania, nie jest w stanie osiągnąć właściwych warunków pracy, bowiem w porównywalnym do SA i GA czasie jest w stanie wykonać zaledwie 50 (lub 10) iteracji, a powinien co najmniej 1000…5000. Zastosowanie w tym przypadku własności blokowych i akceleratora w TS powinno dostarczyć znaczących korzyści. Oddzielnym problemem pozostaje adekwatność testów Taillarda dla zastosowań w systemach bez składowania. Jak już pokazano w testach numerycznych z prac [10, 4], jakość otrzymywanych wyników silnie zależy od cech statystycznych realnych przykładów występujących w praktyce.

Literatura

[1] Grabowski J., Pempera J.: *Sequencing of jobs in some production system*. European Journal of Operational Research 2000, 125, 535–550
[2] Leistein R.: *Flowshop sequencing with limited buffer storage*. International Journal of Production Research 1990, 28, 2085–2100
[3] McCormick M.L., Pinedo M.L., Shenker S., Wolf B.: *Sequencing in an assembly line with blocking to minimize cycle time*. Operations Research 1989, 37, 925–935
[4] Nowicki E.: *The permutation flow shop with buffers: A tabu search approach.* European Journal of Operational Research 1999, 116, 205–219
[5] Pempera J., Smutnicki C.: *Minimalizacja czasu cyklu wytwarzania na linii: Podejście genetyczne z ekspresją genów.* Półrocznik AGH Automatyka 2005, t. 9, z. 1/2, 189–199
[6] Sawik T.: *A scheduling algorithm for flexible flow lines with limited intermediate buffers.* Applied Stochastic Models and Data Analysis, 1993, 9, 127–138
[7] Sawik T.: *Mixed integer programming for scheduling flexible flow lines with limited intermediate buffers*. Mathematical and Computer Modelling, 2000, 31, 39–52
[8] Sawik T.: *An exact approach for batch scheduling in flexible flow lines with limited intermediate buffers*. Mathematical and Computer Modelling, 2002, 36, 461–471
[9] Smutnicki C.: *Some properties of scheduling problem with storage constraints*. Zeszyty Naukowe AGH: Automatyka 1983, 34, 223–232
[10] Smutnicki C.: *A two-machine flow shop scheduling problems with buffers*. OR Spectrum 20, 1999, 229–235
[11] Smutnicki C.: *Algorytmy szeregowania.* Warszawa, EXIT 2002
[12] Taillard E.: *Benchmarks for basic scheduling problems.* European Journal of Operational Research 1993, 64, 278–285