AUTOMATYKA • 2005 • Tom 9 • Zeszyt 1–2

Czesław Smutnicki*, Adam Tyński*

Problem gniazdowy z transportem i ograniczoną liczbą niededykowanych wózków AGV**

1. Wstęp

Dzięki wysokiej elastyczności i stosunkowo niskiemu kosztowi pracy, elastyczne systemy produkcyjne wykorzystujące wózki AGV zwiększają swą popularność na całym świecie. Jednym ze sposobów zwiększenia wydajności takiego systemu jest zintegrowanie fazy szeregowania zadań produkcyjnych z fazą planowania trasy przejazdu wózków.

W tej pracy rozważa się elastyczny system produkcyjny o strukturze gniazdowej, w którym czasy transportu zadań pomiędzy poszczególnymi maszynami mają znaczący wpływ na przebieg całego procesu technologicznego.

Do transportu zadań stosuje się identyczne wózki AGV, których przydział do realizacji poszczególnych operacji transportowych nie jest zadany *a priori*. Uwzględnia się też ograniczenia technologiczne, związane z obecnością skończonej liczby wózków w systemie. Problem polega na wyznaczeniu takiego harmonogramu pracy maszyn i wózków, by kryterium optymalizacji – moment zakończenia wykonywania wszystkich zadań – przyjęło wartość minimalną.

Ponieważ postawiony problem jest silnie *NP*-trudny, w pracy wykorzystuje się model matematyczny, dedykowany dla zastosowania algorytmu przybliżonego opartego o technikę poszukiwań lokalnych.

Wpierw wprowadza się model permutacyjno-grafowy problemu, w którym zbiór wózków AGV utożsamia się ze zbiorem dodatkowych maszyn (transportowych), zaś transport zadań pomiędzy maszynami – z operacjami transportowymi. Następnie definiuje się pojęcie bloku operacji na ścieżce krytycznej grafu i określa się dwa podstawowe zbiory ruchów dla zastosowania techniki poszukiwań lokalnych; zbiory te bazują na:

1) ruchach typu *zamień sąsiednie operacje*,
2) ruchach typu *wstaw* i umożliwia przesuwanie operacji transportowych pomiędzy maszynami transportowymi.

* Instytut Informatyki Automatyki i Robotyki Politechniki Wrocławskiej
** Praca była finansowana ze środków KBN w latach 2003–2006, jako projekt badawczy nr T11A01624

W celu przybliżonego rozwiązania problemu proponuje się algorytm wykorzystujący technikę poszukiwania z zabronieniami. Aby określić jakość dostarczanych rozwiązań, algorytm poddaje się badaniom numerycznym na odpowiednio skonstruowanych zestawach instancji testowych.

2. Model matematyczny i permutacyjno-grafowy

Prezentowany poniżej model matematyczny, jak i reprezentacja permutacyjno-grafowa były szczegółowo omawiane np. w pracach [1, 2]. Dlatego poniżej ograniczymy się do przedstawienia ich najistotniejszych elementów. Dany jest elastyczny system produkcyjny ze zbiorem maszyn $M^p = \{1, 2, ..., m^p\}$ i zbiorem identycznych, dwukierunkowych wózków AGV (maszyn transportowych) $M^t = \{m^p+1, m^p+2, ..., m^p+m^t\}$; m^p i m^t stanowi odpowiednio liczbę maszyn produkcyjnych i transportowych; $m = m^p+m^t$. W systemie należy wykonać r zadań ze zbioru $J = \{1, 2, ..., r\}$. Każde zadanie $k \in J$ składa się z $o_k > 0$ operacji produkcyjnych, indeksowanych przez $j_k+1, j_k+2, ..., j_k+o_k$, które powinny być wykonane w tej kolejności; $j_k = \sum_{i=1}^{k-1} o_i$; $j_1 = 0$. Zbiór operacji produkcyjnych oznaczymy przez $O^p = \{1, 2, ..., n^p\}$, gdzie n^p stanowi liczbę wszystkich operacji produkcyjnych. Każdą operację produkcyjną $j \in O^p$ należy wykonać na maszynie $m(j) \in \mu(j) \subseteq M^p$ w czasie $p_j > 0$, przy czym dla $j \in O^p$ przyjmujemy $|\mu(j)| = 1$.

Poniżej omówimy transporty. Przez $e(x, y) \geq 0$ oznaczymy czas przejazdu pustego (przejazdu bez załadunku) każdego wózka $w \in M^t$ pomiędzy parą maszyn $x, y \in M^p$; $e(x, x) = 0$, $e(x, y) > 0$, $x \neq y$. Operacja transportowa powstaje poprzez wstawienie pomiędzy każdą parę operacji produkcyjnych $j_k + i, j_k + i + 1$ operacji oznaczonej przez $[j_k + i]$, $1 \leq i < o_k$, $k \in J$. Zbiór wszystkich operacji transportowych można zapisać jako $O^t = \bigcup_{k\in J} \bigcup_{i=1}^{o_k-1} \{[j_k + i]\}$, zaś zbiór wszystkich operacji jako $O = O^p \cup O^t$. Liczność powyższych zbiorów wynosi odpowiednio $|O^t| = \sum_{k\in J} o_k - 1 = n^p - r = n^t$ i $|O| = n^p + n^t = n$. Każdą operację $a \in O^t$ należy wykonać na jednej z maszyn ze zbioru $\mu(a) \subseteq M^t$, gdzie $|\mu(a)| \geq 1$. Realizacja operacji transportowej $a = [j_k + i_k]$, $1 \leq i_k < o_k$, polega na przewiezieniu przez maszynę transportową $m(a) \in \mu(a)$ palety z elementem wykonywanym w ramach zadania $k \in J$ pomiędzy maszyną $x = m(j_k + i_k)$ i maszyną $y = m(j_k + i_k+1)$ w czasie $p_a = t_k(x, y) \geq e(x, y)$. Po wykonaniu operacji a i przed wykonaniem operacji $b = [j_g + i_g]$, $1 \leq i_g < o_g$, $g \in J$, wózek wykonuje przejazd pusty w czasie $e(y, z)$ pomiędzy maszyną y i maszyną $z = m(j_g + i_g)$. Przejazd pusty można utożsamić z przezbrojeniem maszyny transportowej $m(a) = m(b)$ trwającym $s(a, b) = e(y, z)$ jednostek czasu. Dla uproszczenia notacji, dla każdej pary operacji produkcyjnych $a, b \in O^p$ przyjmujemy $s(a, b) = 0$.

Uszeregowanie można zdefiniować jako zbiór par $(m(j), S(j))$, $j \in O$, gdzie $m(j)$ jest maszyną wybraną do wykonania operacji j, zaś $S(j) \geq 0$ jest momentem jej rozpoczęcia. Uszeregowanie jest dopuszczalne, gdy spełnia ograniczenia typowe dla klasycznego problemu gniazdowego. Problem polega na odnalezieniu takiego uszeregowania dopuszczalnego, by moment zakończenia wykonywania procesu technologicznego, równy $\max_{j\in O} S(j)+p_j$, przyjął wartość minimalną.

Przejdźmy do krótkiego omówienia modelu permutacyjno-grafowego. Zauważamy, że zbiór operacji O można podzielić na m rozłącznych podzbiorów O_1, O_2, ..., O_m takich, że $\forall_{j\in O}\exists!_{l\in\mu(j)}\ j\in O_l$. Zbiory O_1, O_2, ..., O_{mp} są określone jednoznacznie, zaś ustalenie zbiorów $O_{m^p+1}, O_{m^p+2}, ..., O_m$ wymaga ustalenia maszyny transportowej $m(j) \in \mu(j)$ dla każdej operacji $j \in O^t$. Kolejność wykonania operacji w zbiorach O_1, ..., O_m można zapisać za pomocą zestawu permutacji podziału rozłącznego $\pi = (\pi_1, \pi_2, ..., \pi_m)$, gdzie $\pi_l = (\pi_l(1), \pi_l(2), ..., \pi_l(n_l))$, $n_l = |O_l|$, jest permutacją operacji ze zbioru O_l (dla uproszczenia, zestaw permutacji π również będzie nazywany permutacją, bądź rozwiązaniem problemu). Niech $G(\pi) = (O, E^T \cup E^K(\pi))$ będzie skierowanym grafem ze zbiorem wierzchołków O, zbiorem łuków technologicznych E^T i kolejnościowych $E^K(\pi)$,
gdzie

$$E^T = \bigcup_{k\in J}\ \bigcup_{j=j_k+1}^{j_k+o_k-1}\{(j,[j]),([j],j+1)\},\quad E^K(\pi) = \bigcup_{l\in M}\ \bigcup_{i=2}^{n_l}\{(\pi_l(i-1),\pi_l(i))\} \tag{1}$$

Każdy wierzchołek $j \in O$ przyjmuje obciążenie p_j, wszystkie łuki w zbiorze E^T mają zerowe obciążenie, natomiast każdy łuk $(\pi_l(i–1), \pi_l(i)) \in E^K(\pi)$ przyjmuje obciążenie $s(\pi_l(i–1), \pi_l(i))$. Łatwo zauważyć, że dla każdej permutacji π „zgodnej" z dowolnym uszeregowaniem dopuszczalnym (permutację taką będziemy nazywać permutacją dopuszczalną) graf $G(\pi)$ jest acykliczny. Najdłuższa ścieżka $r^j(\pi)$ w grafie $G(\pi)$ dochodząca do wierzchołka $j \in O$ (bez jego obciążenia p_j) jest równa najwcześniejszemu możliwemu momentowi rozpoczęcia wykonywania $s(j)$ operacji j. Analogicznie, przez $g^j(\pi)$ oznaczymy najdłuższą ścieżkę wychodzącą z wierzchołka j (bez jego obciążenia) w grafie $G(\pi)$. Ścieżką krytyczną, o długości $C_{\max}(\pi) = \max_{j\in O}\{r^j(\pi) + p_j + q^j(\pi)\}$, nazywamy najdłuższą ścieżkę w grafie $G(\pi)$. Długość ścieżki krytycznej jest równa wartości przyjętej funkcji kryterialnej uszeregowania „zgodnego" z permutacją dopuszczalną π. Zatem problem sprowadza się do znalezienia takiej permutacji π^*, że graf $G(\pi^*)$ pozostaje acykliczny i

$$C_{\max}(\pi^*) = \min_{\pi\in\Pi} C_{\max}(\pi),$$

gdzie Π jest zbiorem wszystkich permutacji.

3. Struktura sąsiedztwa i jego własności

Bez straty ogólności dalsze rozważania można ograniczyć do ustalonej permutacji dopuszczalnej π i acyklicznego grafu $G(\pi)$. Dla uproszczenia notacji, jeżeli nie będzie to prowadzić do niejednoznaczności, w poniższych definicjach indeks π będzie pomijany.

Przez b_j^T i b_j^K oznaczymy odpowiednio bezpośredniego poprzednika technologicznego i kolejnościowego wierzchołka $j \in O$ w grafie $G(\pi)$. Analogicznie, niech a_j^T i a_j^K oznacza odpowiednio bezpośredniego następnika technologicznego i kolejnościowego wierzchołka j. Jeśli którykolwiek z poprzedników (następników) technologicznych (kolejnościowych) wierzchołka $j \in O$ nie istnieje, to przyjmujemy, że odpowiednia war-

tość $b_j^T, b_j^K, a_j^T, a_j^K$ wynosi 0. Poniżej precyzyjnie definiujemy ścieżkę krytyczną. Niech $u = (u_1, u_2, ..., u_{lw})$ będzie ścieżką krytyczną w grafie $G(\pi)$, gdzie $u_i \in O$, $1 \le i \le lw$, zaś lw jest liczbą wierzchołków na ścieżce. Zbiór wierzchołków na ścieżce krytycznej u można podzielić na lb rozłącznych podzbiorów $u = (B_1, B_2, ..., B_{lb})$, gdzie $B_h = (\pi_{l_h}(e_h), \pi_{l_h}(e_h+1), ..., \pi_{l_h}(f_h))$, $1 \le e_h \le f_h \le n_{l_h}$, $1 \le h \le lb$ jest h-tym blokiem operacji na ścieżce u takim, że $l_h \in M$ oraz $l_h \ne l_{h-1}$, $1 < h \le lb$. Własności blokowe w problematyce szeregowania zadań były szeroko dyskutowane m.in. w książce [3]. W dalszych rozważaniach będzie również użyteczny zbiór $Z = \bigcup_{h=1}^{lb} Z_h$,
gdzie

$$Z_h = \begin{cases} \{\pi_{l_h}(e_h), \pi_{l_h}(f_h)\}, & e_h < f_h \\ \varnothing, & e_h = f_h \end{cases} \qquad h = 1, 2, ..., lb \tag{2}$$

Sąsiedztwo przedstawiane w tej pracy generowane jest w oparciu o ruchy typu wstaw oraz ruchy typu *zamień sąsiednie operacje*. Niech $m^\pi(h)$ oznacza maszynę, na której wykonywana jest operacja $h \in O^t$ w permutacji π, zaś $\mu'(h) = \mu(h) \setminus \{m^\pi(h)\}$. Niech $v(h, k, z)$ będzie ruchem typu wstaw, gdzie $h \in O^t$, $k \in \mu'(h)$, $1 \le z \le n_k + 1$. Zastosowanie ruchu v do permutacji π można podzielić na dwa etapy:

(*i*) pobierana jest operacja $h = \pi_l(x)$ znajdująca się na pozycji $1 \le x \le n_l$ w permutacji π_l, $l = m^\pi(h)$;
(*ii*) operacja h wstawiana jest na pozycję z w permutacji π_k.

Permutację powstałą w wyniku zastosowania ruchu v do π oznaczymy przez π_v. Zauważmy, że każdy ruch typu wstaw $v = (h, l, x + 1)$ taki, że $h = \pi_l(x)$, $1 \le x < n_l$, $l \in M$ może być utożsamiony z ruchem typu zamień sąsiednie operacje. Jednakże, dla rozróżnienia, w dalszej części pracy ruchy typu zamień sąsiednie operacje będą notowane jako para operacji (h, j), gdzie $h = \pi_l(x)$, $j = \pi_l(x+1)$.

Zbiór wszystkich ruchów oznaczymy jako sumę $V = V^S \cup V^I$ zbioru ruchów typu zamień sąsiednie operacje V^S i ruchów typu wstaw V^I, natomiast sąsiedztwem $N = N^S \cup N^I$ rozwiązania π będziemy nazywać zbiór rozwiązań powstały w wyniku zastosowania wszystkich ruchów ze zbioru V do permutacji π. Zbiór V^S (i odpowiadające mu sąsiedztwo N^S) był już przedstawiany w pracy [2] i nie będzie omawiany. W pracy [2] zaprezentowano również efektywną czasowo metodę przeszukiwania sąsiedztwa N^S. W pracy tej dowiedziono, że dla dowolnego rozwiązania sąsiedniego $\delta \in N^S$ rozwiązania π wartość $C_{\max}(\delta)$ można wyznaczyć w czasie $O(\max\{\sum_{j=1}^{r} \log o_j, \sum_{l=1}^{m} \log n_l\})$. Zbiór V^I można przedstawić jako sumę $V^I = \bigcup_{h \in Z} V_h^I$,
gdzie

$$V_h^I = \bigcup_{k \in \mu'(h)} \{(h, k, z) : 1 \le z \le n_k + 1\} \tag{3}$$

Zbiór V_h^I, $h \in Z$, zawiera wszystkie możliwe ruchy polegające na wstawieniu operacji h na maszyny ze zbioru $\mu'(h)$. Zastosowanie dowolnego ruchu ze zbioru V^I do permutacji π

nie gwarantuje zatem dopuszczalności permutacji π_v. Poniżej przedstawione jest praktyczne rozwinięcie metody z pracy [4], dzięki której możliwe jest wyznaczenie najlepszego (w sensie wartości funkcji celu) rozwiązania dopuszczalnego w zbiorze N^I bez konieczności przeglądu całego zbioru. W pracy tej rozważano przypadek transportów zadaniowo-niezależnych i przyjmowano $t_k(x, y) = t(x, y) = e(x, y)$, $k \in J$, $x, y \in M^p$. Jednakże, bazując na dowodach przeprowadzonych w cytowanej pracy, można łatwo dowieść, że uzyskane rezultaty są również prawdziwe w przypadku transportów zadaniowo-zależnych gdy spełniony jest warunek $t_k(x, y) \geq e(x, y)$.

Zdefiniujmy pewne dodatkowe oznaczenia. Dana jest operacja $h \in Z$ oraz ruch $v = (h,k,z) \in V_h^I$.

Niech

$$G(\gamma^h) = \{G(\pi) : E^K(\gamma^h) = E^K(\pi) \cup \{(b_h^K, a_h^K)\} \setminus \{(b_h^K, h), (h, a_h^K)\} \tag{4}$$

będzie grafem powstałym z $G(\pi)$ po wykonaniu etapu (i) ruchu v. Wartość

$$d_v = \max\{R, r^i(\gamma^h) + p_i + s(i,h)\} + p_h + \max\{Q, q^j(\gamma^h) + p_j + s(h, j)\} \tag{5}$$

oznacza długość najdłuższej ścieżki przechodzącej przez wierzchołek h, w grafie $G(\pi_v)$, gdzie $R = r^h(\gamma^h)$, $Q = q^h(\gamma^h)$, $i = \pi_k(z–1)$ oraz $j = \pi_k(z)$. Przyjmujemy również, że $r^0(\gamma^h) = 0$, $q^0(\gamma^h) = 0$. Interpretacja równania

$$d_{\min} = \min_{v \in V_h^I} d_v \tag{6}$$

jest oczywista. Niech ruch $v = (h, k_h, z_h) \in V_h^I$ będzie ruchem takim, że:

$$k_h = \{k \in \mu'(j) : d_v = d_{\min}\} \tag{7}$$

$$z_h = \begin{cases} \min\{c \leq z \leq b : d_{(h,k_h,z)} = d_{\min}\}, & a < c < b \\ b, & b \leq c \\ a, & c \leq a \end{cases} \tag{8}$$

gdzie:

$$a = \min\{1 \leq z \leq n_{k_h} + 1 : d_{(h,k_h,z)} = d_{\min}\} \tag{9}$$

$$b = \max\{1 \leq z \leq n_{k_h} + 1 : d_{(h,k_h,z)} = d_{\min}\} \tag{10}$$

$$c = \max\{1 \leq z \leq n_{k_h} + 1 : r^i(\gamma^h) + p_i \leq R\} \tag{11}$$

oraz $i = \pi_{k_h}(z–1)$.

Bezpośrednią konsekwencją rozważań przeprowadzonych w pracy [4] jest fakt, że ruch v prowadzi do najlepszego rozwiązania w zbiorze N_h^I, zaś $C_{\max}(\pi_v) = \max\{d_{\min}, C_{\max}(\gamma^h)\}$. Powyższa metoda umożliwia wyznaczenie najlepszego rozwiązania w zbiorze N^I w czasie $O(|Z|\cdot n)$.

4. Algorytm poszukiwań z zabronieniami

Technika poszukiwania z zabronieniami, zwana też metodą tabu, jest techniką poszukiwania lokalnego, której charakterystycznym elementem jest tzw. lista tabu. Lista tabu jest krótkoterminową pamięcią, która zabezpiecza algorytm przed powrotem do rozwiązań już wcześniej przeglądanych. Charakter przechowywanej na liście informacji w dużym stopniu zależy od konkretnego problemu i inwencji autora. Na liście takiej często zapamiętywane są ruchy, które uzyskują status zabronionych i których nie można zastosować do żadnej permutacji. Jednakże, w myśl metody przedstawionej w punkcie 3, w trakcie wyznaczania najlepszego rozwiązania w sąsiedztwie N^I, nie przegląda się całego zbioru V^I ustalonego rozwiązania π. Wiąże się to z koniecznością zaprojektowania specyficznej listy tabu. Autorzy pracy rozwiązali ten problem przez podział listy tabu na dwie rozłączne listy.

W proponowanej poniżej procedurze pierwsza z list tabu przechowuje tylko ruchy typu zamień sąsiednie operacje lub ruchy sztuczne $dv = (0,0)$. Formalnie listę tą będziemy notować jako $T^S = (T_1^S, T_2^S, ..., T_{\max\_s}^S)$, gdzie $T_i^S = (h, j)$, $1 \le i \le \max\_s$, $h, j \in$ O zaś max_s jest długością listy. Korzystając z notacji z prac [2, 5], ruch odwrotny do $v = (h, j)$ będzie oznaczany przez $\overline{v} = (j, h)$. Każdorazowo, gdy do ustalonej permutacji π stosowany jest ruch $v = (h, j) \in V^S$, do listy T^S dodawany jest ruch $\overline{v}$. Operację tą będziemy notować jako $T^S \oplus \overline{v}$. Polega ona na położeniu $T_i^S := T_{i+1}^S$, i = 1, 2, ..., max_s – 1 i $T_{\max\_s}^S := \overline{v}$. Drugą listę oznaczymy przez $T^I = (T_1^I, T_2^I, ..., T_{\max\_i}^I)$, gdzie $T_i^I \in O^t \cup \{do\}$, $1 \le i \le \max\_i$ zaś max_i jest długością listy. Lista ta przechowuje operacje transportowe bądź operacje sztuczne $do = 0$. Operację dodawania elementu $h \in O^t \cup \{do\}$ do listy T^I przebiega w sposób analogiczny jak w przypadku listy T^S i będzie notowana $T^I \oplus h$. Zbiór ruchów zabronionych w zbiorze V^I będziemy notować przez $L^I = \{(h,k,z) \in V^I : h \in T^I\}$.

W dalszej części pracy zapis $T = \varnothing$, gdzie $T = T^S \cup T^I$ będzie oznaczać, że na każdej z list T^S, T^I przechowywane są odpowiednio tylko ruchy sztuczne i tylko operacje sztuczne. W powyższy sposób zdefiniowana lista tabu T w pewnych sytuacjach może okazać się zbyt restrykcyjna i może uniemożliwiać osiągniecie relatywnie dobrych regionów przestrzeni rozwiązań. Dlatego w prezentowanym algorytmie dodatkowo dopuszcza się możliwość zastosowania ruchów ze zbioru

$$A = \{v \in V^S \cap T^S : C_{\max}(\pi_v) < C^*\} \cup \{v \in L^I : C_{\max}(\pi_v) < C^*\} \tag{12}$$

gdzie C^* jest najlepszą wartością funkcji celu znalezioną we wcześniejszych iteracjach algorytmu.

Najważniejszym elementem badanego w pracy algorytmu jest przedstawiona poniżej procedura przeszukiwania sąsiedztwa (PPS). Najprostszy algorytm tabu uzyskuje się poprzez iterowanie procedury PPS przez określoną liczbę iteracji. Najlepsze jednak wyniki autorzy pracy uzyskali po osadzeniu procedury PPS w miejscu procedury NSP w algorytmie TSAB, przedstawionym w pracy [5], i jednoczesnym zastąpieniu oryginalnego sąsiedztwa sąsiedztwem *N*, omawianym w punkcie 3. W ten sposób uzyskany algorytm będziemy nazywać algorytmem $TSAM_{AGV}$. Podobnie jak oryginalny algorytm, algorytm $TSAM_{AGV}$ wymaga określenia szeregu parametrów, takich jak *maxiter* – maksymalna liczba iteracji bez poprawy wartości C^*, *maxl* – maksymalna długość listy dla skoków powrotnych, *max*δ i *maxc*, używanych w detektorze cykli. Oryginalny parametr *maxt* (oznaczający długość listy tabu) został oczywiście zastąpiony przez parametry *max_s* i *max_i*, opisywane wcześniej.

Procedura Przeszukiwania Sąsiedztwa

Procedura rozpoczyna działanie z ustaloną permutacją π listą tabu $T = T^S \cup T^I$, niepustym zbiorem ruchów $V = V^S \cup V^I$ i wartością funkcji celu C^*. Procedura zwraca ruch v', zmodyfikowaną listę tabu T' i nową permutację π'.

Krok 1.

Stosując efektywną metodę wyznaczania, dla każdego ruchu $v \in V^S$ wyznacz i zapamiętaj wartość $C_{max}(\pi_v)$.

Krok 2.

Połóż $W := \emptyset$. Dla każdego $h \in Z$ powtarzaj kroki 3 – 4.

Krok 3.

Skonstruuj graf $G(\gamma^h)$, wyznacz wartości $r^i(\gamma^h)$, $q^i(\gamma^h)$, $i \in O$ oraz $C_{max}(\gamma^h)$. Dla każdego $v \in V_h^I$ wyznacz wartość d_v daną równaniem (5). Wyznacz wartość d_{min} daną równaniem (6).

Krok 4.

Wyznacz pozycje *a*, *b*, *c* dane równaniami (9)–(11). Korzystając z równań (7), (8), wyznacz odpowiednio maszynę k_h oraz pozycję z_h. Połóż $w_h := (h, k_h, z_h)$, $W := W \cup \{w_h\}$ oraz $C_{max}(\pi_{wh}) := \max\{C_{max}(\gamma^h), d_{min}\}$.

Krok 5.

Wyznacz zbiór *A* dany równaniem (12). Wyznacz zbiór $B = \{\{V^S \cup W\} \setminus \{T^S \cup L^I\}\} \cup A$. Jeżeli $B \neq \emptyset$, to wybierz $v' \in B$ taki, że $C_{max}(\pi_{v'}) = \min\{C_{max}(\pi_v) : v \in V\}$ i idź do kroku 11.

Krok 6.

Jeżeli $|V| = 1$, to $v' \in V$ wybierz i idź do kroku 11.

Krok 7.

Jeżeli $T^S \neq \emptyset$, to wyznacz $i = \min\{1 \leq j \leq max\_s : T_j^S \neq dv\}$. W przeciwnym przypadku idź do kroku 9.

Krok 8.

Jeżeli $i \neq \max\_s$, to powtarzaj $T^S := T^S \oplus T^S_{\max\_s}$, dopóki $V^S \setminus T^S = \varnothing$. W przeciwnym przypadku powtarzaj $T^S := T^S \oplus dv$ dopóki $V^S \setminus T^S = \varnothing$. Wybierz $v' \in V^S \setminus T^S$ i idź do kroku 11.

Krok 9.

Znajdź $i = \min\{1 \le j \le \max\_i : T^I_j \neq do\}$. Połóż $h := T^I_i$ oraz $v' := w_h$.

Krok 10.

Jeżeli $i \neq \max\_i$, to powtarzaj $T^I := T^I \oplus T^I_{\max\_i}$, dopóki $V^I \setminus L^I = \varnothing$. W przeciwnym przypadku powtarzaj $T^I := T^I \oplus do$, dopóki $V^I \setminus L^I = \varnothing$.

Krok 11.

Jeżeli $v' \in V^S$, to połóż $T^S := T^S \oplus \overline{v'}$ i $T^I := T^I \oplus do$. W przeciwnym przypadku połóż $T^I := T^I \oplus h$ i $T^S := T^S \oplus dv$. Połóż $\pi' := \pi_{v'}$ i $T' := T^S \cup T^I$.

5. Rezultaty testów numerycznych

Ze względu na brak instancji testowych dla problemu gniazdowego z transportem i ograniczoną liczbą wózków (z wyjątkiem szczególnych przypadków, w których $m^t = 1$), zdecydowaliśmy się zaprojektować własne instancje. Proponowane poniżej instancje są modyfikacją 40 instancji zaproponowanych dla klasycznego problemu gniazdowego przez Lawrence'a w pracy [6]. Poprzez uwzględnienie różnych kombinacji układów maszyn, liczby wózków oraz czasów transportów i przejazdów pustych utworzyliśmy 480 instancji. Każda instancja ma swoją unikalną nazwę w postaci TR$a/m^t/b/c/d$, gdzie a, m^t, b, c, d są zmiennymi. Zmienna a, $1 \le a \le 40$ określa numer instancji Lawrence'a użytej do modyfikacji. Ze względu na wartości zmiennej a instancje TR1-40 dzieli się na 8 grup po 60 instancji, zróżnicowanych z racji rozmiaru instancji. Dla wszystkich instancji w grupie TR1-5, TR6-10, ..., TR36-40 mamy odpowiednio $r \times m^p$ = 10×5, 15×5, 20×5, 10×10, 15×10, 20×10, 30×10, 15×15 zadań i maszyn produkcyjnych. Zmienna m^t określa liczbę wózków AGV i przyjmuje jedną z wartości ze zbioru $m^t \in \{2, 3\}$. Zmienne c i d są współczynnikami skalującymi dla odpowiednio czasów przejazdów pustych i czasów transportów i mogą przyjmować jedną z par wartości $(c,d) \in \{(2,2), (2,5), (5,5)\}$. Dla każdej pary maszyn produkcyjnych $x, y \in M^p$ mamy $e(x, y) = c \cdot g(x, y)$ i $t_j(x, y) = d \cdot g(x, y)$, $j \in J$ gdzie $g(x, y)$ jest odległością pomiędzy maszynami x, y zależną od parametru b. Zmienna b, $b \in \{1, 2\}$, określa typ układu maszyn. Gdy $b = 1$ przyjmujemy, że wszystkie maszyny produkcyjne zorganizowane są w pojedynczą linię i mamy $g(x, y) = |y - x|$. Gdy $b = 2$, maszyny są zorganizowane w układ typu pętla i odległości są wyznaczane z równania

$$g(x,y) = \begin{cases} |y-x|, & |y-x| < \left\lceil m^p/2 \right\rceil \\ m^p - |y-x|, & \text{w przeciwnym przypadku} \end{cases} \tag{13}$$

Wszystkie instancje zostały w sposób przybliżony rozwiązane przez algorytmy zaimplementowane w Delphi i uruchamiane na komputerze Athlon XP 2500+ (1833 MHz). Pierwszy algorytm, o nazwie i-TSAB$_{AGV}$, został przedstawiony w pracy [2]. Algorytm ten jest algorytmem poszukiwania z zabronieniami, wyposażony dodatkowo w schemat dywersyfikacji obliczeń. Algorytm oryginalnie został zaprojektowany dla problemu gniazdowego z transportem, w którym zakłada się, że przydział wózków AGV do poszczególnych operacji transportowych dany jest *a priori*. Użycie algorytmu do rozważanego problemu możliwe jest po uprzednim skonstruowaniu przydziału wózków do operacji transportowych przez np. algorytm konstrukcyjny INT1 z pracy [7]. Startując z rozwiązania początkowego dostarczonego przez INT1, algorytm i-TSAB$_{AGV}$ został uruchomiony dwa razy z 10-minutowym ograniczeniem czasowym i wartościami parametrów identycznymi jak w pracy [2]. Wartość funkcji celu najlepszego rozwiązania dostarczonego przez i-TSAB$_{AGV}$ będziemy oznaczać przez C^{TB}.

Startując z rozwiązania dostarczonego przez INT1 algorytm TSAM$_{AGV}$ został uruchomiony dwukrotnie z 10-minutowym ograniczeniem czasowym i wartościami parametrów max$max\delta$ = 100, $maxc$ = 2, max_s = 15, max_i = 5. W pierwszym uruchomieniu przyjęto $maxiter$ = 16000. W drugim uruchomieniu przyjęto $maxiter$ = 30 000 w momencie startu algorytmu i po każdej poprawie wartości funkcji celu oraz $maxiter$ = 10 000 po wykonaniu skoku powrotnego (szczegóły opisane są w pracy [5]). Wartość funkcji celu najlepszego rozwiązania dostarczone przez algorytm TSAM$_{AGV}$ dla poszczególnych instancji będziemy oznaczać symbolem C^{TM}.

Algorytm TSAM$_{AGV}$ dowiódł optymalności rozwiązań dla liczby instancji odpowiednio 24, 48, 60, 0, 3, 7, 29, 0 w grupach TR1-5, TR6-10, ..., TR36-40. Ze względu na zróżnicowany rozmiar i niewielką liczbę instancji rozwiązanych optymalnie najciekawszymi wydają się być grupy TR16-20, TR21-25 i TR36-40, do których ograniczymy nasze dalsze rozważania. Dla każdej instancji z powyższych grup została wyznaczona poprawa względna

$$\Phi(TB,TM) = \frac{C^{TB} - C^{TM}}{C^{TB}} \cdot 100\% \tag{14}$$

algorytmu i-TSAB$_{AGV}$ przez algorytm TSAM$_{AGV}$. W tabeli 1 zostały umieszczone wartości C^{TM}, zaś w tabeli 2 wartości $\Phi(TB, TM)$wyznaczone dla każdej instancji w omawianych grupach instancji.

Algorytm TSAM$_{AGV}$ poprawił większość rozwiązań dostarczonych przez algorytm i-TSAB$_{AGV}$. Najmniejsza i największa wyznaczona poprawa wynosi odpowiednio –2,5% i 10,4%. Algorytm i-TSAB$_{AGV}$ nie jest algorytmem dedykowanym problemowi omawianemu w tej pracy. Mimo to, w sporadycznych przypadkach, algorytm ten dostarczył rozwiązań lepszych, niż algorytm TSAM$_{AGV}$. Brak konieczności przesuwania operacji transportowych pomiędzy wózkami wpłynął na zmniejszenie złożoności obliczeniowej algorytmu w stosunku do algorytmu TSAM$_{AGV}$, co umożliwiło zaimplementowanie dodatkowych, szybkich mechanizmów dywersyfikacji. Jednakże lepsze wyniki zostały osiągnięte jedynie w przypadku instancji charakteryzujących się stosunkowo niewielkimi czasami transportów i przejazdów bez załadunku. W miarę wzrostu czasu transportów i przejazdów pustych w stosunku do czasów wykonania operacji produkcyjnych przewaga algorytmu TSAM$_{AGV}$ nad algorytmem i-TSAB$_{AGV}$ zwiększa się.

Tabela 1
Wartości funkcji celu dostarczone przez algorytm $TSAM_{AGV}$

TRa	2/1/c/d			2/2/c/d			3/1/c/d			3/2/c/d		
	2/2	2/5	5/5	2/2	2/5	5/5	2/2	2/5	5/5	2/2	2/5	5/5
TR16	1000	1106	1157	976	1052	1057	1000	1053	1063	976	1040	1044
TR17	809	1023	1119	805	892	935	809	892	908	805	848	848
TR18	891	1017	1109	884	939	965	889	954	967	881	931	931
TR19	898	1000	1087	874	931	968	886	946	956	874	918	921
TR20	936	1010	1074	931	978	996	936	978	987	931	975	975
TR21	1105	1387	1496	1084	1162	1243	1099	1171	1215	1082	1129	1131
TR22	986	1377	1480	953	1084	1166	955	1056	1120	953	1007	1009
TR23	1060	1439	1584	1032	1130	1225	1032	1109	1174	1032	1047	1047
TR24	1010	1519	1686	990	1177	1276	1006	1195	1207	967	1023	1038
TR25	1018	1344	1471	1009	1127	1211	1018	1100	1131	1004	1041	1071
TR36	1615	3210	3559	1418	2479	2758	1453	2245	2433	1363	1751	1970
TR37	1668	3267	3597	1542	2383	2699	1532	2244	2517	1523	1753	1928
TR38	1512	3074	3427	1361	2445	2703	1340	2110	2346	1314	1723	1916
TR39	1487	2826	3229	1373	2268	2556	1337	1963	2228	1350	1652	1803
TR40	1553	3232	3590	1375	2326	2598	1381	2212	2433	1306	1665	1854

Tabela 2
Poprawy względne algorytmu *i*-$TSAB_{AGV}$ przez algorytm $TSAM_{AGV}$

TRa	2/1/c/d			2/2/c/d			3/1/c/d			3/2/c/d		
	2/2	2/5	5/5	2/2	2/5	5/5	2/2	2/5	5/5	2/2	2/5	5/5
TR16	0,30	3,41	7,96	0,00	0,28	4,52	0,00	3,13	4,66	0,00	0,67	1,88
TR17	0,37	3,58	5,01	0,00	1,55	6,59	0,61	4,60	7,72	0,00	1,62	4,72
TR18	0,56	4,33	6,65	–0,34	2,09	6,22	0,78	2,55	5,20	0,00	0,53	2,31
TR19	0,11	5,84	10,39	0,57	1,69	4,35	0,45	4,44	6,37	0,57	1,50	3,26
TR20	0,53	2,23	5,46	0,75	0,51	3,21	0,53	2,30	4,82	0,00	0,41	1,42
TR21	–0,55	1,70	4,83	0,00	1,19	2,81	0,09	2,98	6,68	0,18	0,79	2,33
TR22	–1,75	2,13	5,49	0,00	3,39	5,82	0,52	5,12	8,57	0,00	1,66	5,96
TR23	–2,51	2,18	4,46	0,00	1,74	4,60	0,00	5,05	6,97	0,00	0,38	5,08
TR24	–1,10	1,68	2,37	–1,64	0,25	1,69	–0,70	0,33	7,93	0,00	2,29	6,32

Tabela 2 cd.

TRa	2/1/c/d			2/2/c/d			3/1/c/d			3/2/c/d		
	2/2	2/5	5/5	2/2	2/5	5/5	2/2	2/5	5/5	2/2	2/5	5/5
TR25	0,68	3,10	6,84	–0,70	2,42	3,89	0,10	1,70	7,90	–0,20	2,16	3,60
TR36	6,05	1,65	3,45	2,74	0,16	3,77	1,82	2,86	4,55	1,09	4,94	4,42
TR37	4,47	0,76	1,86	0,64	2,01	3,85	3,16	3,81	3,71	–0,46	4,00	6,45
TR38	4,00	1,79	2,31	5,16	1,57	3,01	4,90	3,52	3,54	1,35	4,06	5,10
TR39	4,25	0,18	1,40	3,17	2,79	3,87	4,77	3,96	2,96	–1,35	4,40	9,67
TR40	6,84	0,49	3,08	2,69	1,32	4,49	3,22	3,95	6,03	2,17	2,86	5,79
średnia	**1,48**	**2,34**	**4,77**	**0,87**	**1,53**	**4,18**	**1,35**	**3,35**	**5,84**	**0,22**	**2,15**	**4,55**

6. Zakończenie

W tej pracy rozważano elastyczny system produkcyjny o strukturze gniazdowej, w którym uwzględniono niezerowe czasy transportów zadań pomiędzy poszczególnymi maszynami. Do transportów stosuje się zbiór identycznych, dwukierunkowych wózków AGV, których przydział do poszczególnych transportów nie jest z góry ustalony.

Dla problemu zaproponowano algorytm popraw, wykorzystujący technikę poszukiwania za zabronieniami oraz dodatkowy mechanizm umożliwiający przesuwanie operacji transportowych pomiędzy poszczególnymi wózkami. Zaproponowany algorytm został poddany badaniom numerycznym przy użyciu specjalnie skonstruowanych zestawów instancji testowych. Algorytm dostarczył rozwiązań średnio znacznie lepszych, niż podobny algorytm, nie posiadający mechanizmu przesuwania operacji pomiędzy wózkami.

Literatura

[1] Nowicki E., Tyński A.: *Problem gniazdowy z transportem.* W: Badania operacyjne i systemowe wobec wyzwań XXI wieku, pod red. Z. Bubnickiego, O. Hryniewicza i J. Węglarza, Warszawa, EXIT 2004

[2] Smutnicki C., Tyński A.: *Simultaneous machine scheduling and AGV route planning in the FMS.* Praca zgłoszona na 11 międzynarodową konferencję "Methods and Models in Automation and Robotics 2005"

[3] Grabowski J., Nowicki E., Smutnicki C.: *Metoda blokowa w zagadnieniach szeregowania zadań.* Warszawa, EXIT 2003

[4] Nowicki E., Tyński A.: *Analiza sąsiedztwa w problemie gniazdowym z ograniczoną liczbą wózków AGV.* W: Automatyzacja procesów dyskretnych, pod red. R. Gessinga i T. Szkodnego, Warszawa, WNT 2004

[5] Nowicki E., Smutnicki C.: *A fast taboo search algorithm for the job shop problem.* Management Science, 42, 1996, 797

[6] Lawrence S.: *Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement).* Pittsburg, Pensylwania, Graduate School of Industrial Administration, Carnegie-Mellon University, 1984

[7] Nowicki E., Tyński A.: *Algorytmy konstrukcyjne dla problemu gniazdowego z czasami transportu.* W: Komputerowo zintegrowane zarządzanie, pod red. T. Knosali, tom II, Warszawa, WNT 2003