OPTYMALIZACJA W KARUZELOWYCH SYSTEMACH PRZEPŁYWOWYCH

Jarosław PEMPERA

Streszczenie: Praca poświęcona jest harmonogramowaniu zadań produkcyjnych w karuzelowym systemie produkcyjnym. W systemie należy wyznaczyć harmonogram wykonywania operacji na maszynach minimalizujący czas realizacji wszystkich zadań. W pracy zaproponowano opis matematyczny problemu oraz model grafowy. W oparciu o model grafowy sformułowano szereg własności problemu, które zostały wykorzystane w konstrukcji algorytmów opartych na metodach przeszukiwań lokalnych. Algorytmy zostały poddane testom komputerowym na literaturowych instancjach problemu przepływowego zaproponowanych przez Tailarda.

Słowa kluczowe: karuzelowy system przepływowy, optymalizacja, algorytmy heurystyczne.

1. Wstęp

Karuzelowe systemy produkcyjne charakteryzują się specyficzną formą transportu przetwarzanych produktów pomiędzy kolejnymi maszynami w ciągu technologicznym. Transport produktów odbywa się na obrotowej platformie. Produkty przymocowywane są na czas obróbki do uchwytów rozmieszczonych równomiernie na obwodzie platformy. Podczas obrotu platformy wszystkie produkty znajdujące się na platformie przemieszczają się jednocześnie z jednej maszyny do innej stanowiącej kolejny etap produkcyjny. W systemie wydzielone jest stanowisko załadowczo-wyładowcze. Systemem karuzelowym możemy nazwać również system produkcyjny, w którym transport realizowany jest przez zsynchronizowane działanie innego typu środków transportowych np. taśmociągów, podajników, wózków AGV itp.

Maszyny i systemy karuzelowe są często używane w praktyce, w szczególności w produkcji masowej. Maszyny karuzelowe stosuje się na przykład w przemyśle spożywczym – rozlewanie do opakowań produktów płynnych, AGD – formowanie plastikowych elementów urządzeń, tekstylnym – drukowanie ozdób metodą sitodruku, elektronicznym – montaż podzespołów elektronicznych, samochodowym – montaż samochodów.

Do niewątpliwych zalet tego typu systemów należy prosty system transportowy, niewielka powierzchnia robocza, prosty sposób zasilania systemu w narzędzia i materiały, duże, równomierne i zdyscyplinowane wykorzystanie siły roboczej. W dobrze zbalansowanym karuzelowym systemie produkcyjnym wytwarzanie podzielone jest na jednakowej długości okresy (takty), w których czas intensywnej pracy maszyn i operatorów przerywany jest w jednakowych momentach przez obrót platformy.

Jednymi z najbardziej atrakcyjnych systemów karuzelowych są systemy montażu podzespołów elektronicznych na płytkach drukowanych. Optymalizacja w takich systemach obejmuje przydział elementów do urządzeń montujących oraz ustalenie kolejności ich

montowania. W systemach tego typu występuje wiele ograniczeń np. skończona liczba podajników przy każdej maszynie, dedykowane głowice do określonych typów elementów, itp. Różne aspekty optymalizacyjne w różnych typach systemów montowania przedstawione są w pracach [1−3]. Optymalizacji kolejności wykonywania produktów obuwniczych w karuzelowym systemie produkcyjnym poświęcona jest praca [4].

2. Opis problemu

W karuzelowym systemie produkcyjnym składającym się z m maszyn technologicznych ze zbioru $M=\{1,...,m\}$ należy wykonać n zadań ze zbioru $J=\{1,...,n\}$. Zadanie $j\in J$ wykonywane jest na maszynie $k\in M$ przez czas $p_{jk}\geq 0$. Dopuszczalne jest pominięcie przez zadanie pewnych etapów produkcyjnych. Wtedy żadne czynności technologiczne nie są wykonywane w czasie, gdy zadanie znajduje się przy maszynie realizującej dany etap.

W danej chwili jedna maszyna może realizować tylko jedno zadanie oraz tylko jedna operacja danego zadania może być wykonywana. Transport wszystkich zadań znajdujących się na platformie następuje jednocześnie. Przyjmuje się, że wszystkie zadania są dostępne w okresie wyznaczania harmonogramu, czas wprowadzenia i usunięcia zadania z systemu jest pomijalnie mały.

Niech $\pi=(\pi(1),...,\pi(n))$ będzie permutacją określoną na zbiorze $\{1,...,n\}$. Permutacja π określa kolejność ładowania zadań do systemu karuzelowego. Niech S_{jk} (C_{jk}) będzie momentem rozpoczęcia (zakończenia) realizacji zadania $j\in J$ na maszynie $k\in M$. Przez T_s, $s=1,...,n+m$ oznaczmy moment rozpoczęcia s-tego taktu systemu produkcyjnego. Oczywiście dla $s=1$ pierwsze zadanie z π transportowane jest ze stanowiska załadowczego do maszyny pierwszej, natomiast dla $s=n+m$ ostatnie zadanie transportowane jest do stanowiska wyładowczego.

Dla kolejności *ładującej* π, zdarzenia S_{jk} oraz C_{jk}, $j\in J$, $k\in M$, określające harmonogram wykonywania zadań na maszynach oraz zdarzenia T_s, $s=1,...,n+m$ muszą spełniać następujące nierówności:

$$T_1 = 0, \tag{1}$$

$$S_{j,k} \geq C_{j,k-1} \qquad j\in J,\ k=2,...,m, \tag{2}$$

$$C_{j,k} = S_{j,k} + p_{j,k} \qquad j\in J,\ k\in M, \tag{3}$$

$$S_{\pi(s+1-k),k} \geq T_s + t, \qquad \begin{cases} s=1,..,m-1,\ k=1,..,s, \\ s=m,..,n,\ k=1,..,m, \\ s=n+1,..,n+m,\ k=s-n,..,m, \end{cases} \tag{4}$$

$$T_s \geq C_{\pi(s-k),k}, \qquad \begin{cases} s=2,..,m,\ k=1,..,s, \\ s=m+1,..,n+1,\ k=1,..,m, \\ s=n+1,..,n+m,\ k=s-n,..,m, \end{cases} \tag{5}$$

gdzie: $t > 0$ - oznacza czas obrotu platformy.

Podstawienie (1) kotwiczy harmonogram zdarzeń w momencie 0. Ograniczenia (2−3)

modelują wymagania technologiczne. Nierówność (2) oznacza, że moment rozpoczęcia zadania na maszynie realizującej pewien etap produkcyjny nie może być wcześniejszy od momentu zakończenia realizacji zadania na maszynie w poprzednim etapie. Równość (3) wiąże moment rozpoczęcia i zakończenia realizacji zadania.

Ograniczenia (4−5) wynikają bezpośrednio z karuzelowej struktury systemu transportowego. Moment rozpoczęcia dowolnego zadania przetwarzanego w *s*-tym takcie systemu produkcyjnego nie może być wcześniejszy od momentu zakończenia obrotu platformy (4) oraz moment rozpoczęcia obrotu platformy nie może być wcześniejszy od najpóźniejszego z momentów zakończenia zadań realizowanych w danym takcie (5).

Celem optymalizacji jest wyznaczenie permutacji ładującej π^* takiej, że

$$C_{\max}(\pi^*) = \min_{\pi\in\Pi} C_{\max}(\pi) \text{ , gdzie } C_{\max}(\pi) = \max_{j=1,..,n} C_{\pi(j),m} \,. \tag{6}$$

Moment $C_{\pi(s),k}$ oznacza najwcześniejszy moment zakończenia realizacji *s*-tego w kolejności π zadania na maszynie *k*, natomiast funkcja $C_{\max}(\pi)$ wyznacza najwcześniejszy moment zakończenia realizacji wszystkich zadań wprowadzanych do systemu w kolejności π. Łatwo można zauważyć, że $C_{\max}(\pi)=C_{\pi(n),m}$.

3. Model grafowy

Rozważany system karuzelowy wygodnie jest modelować w postaci obciążonego grafu skierowanego $G(\pi)$ zdefiniowanego dla zadanej permutacji ładującej π. W grafie $G(\pi)=(V,E)$ węzły ze zbioru *V* odpowiadają zdarzeniom w systemie, natomiast łuki ze zbioru *E* relacjom kolejnościowym pomiędzy nimi.

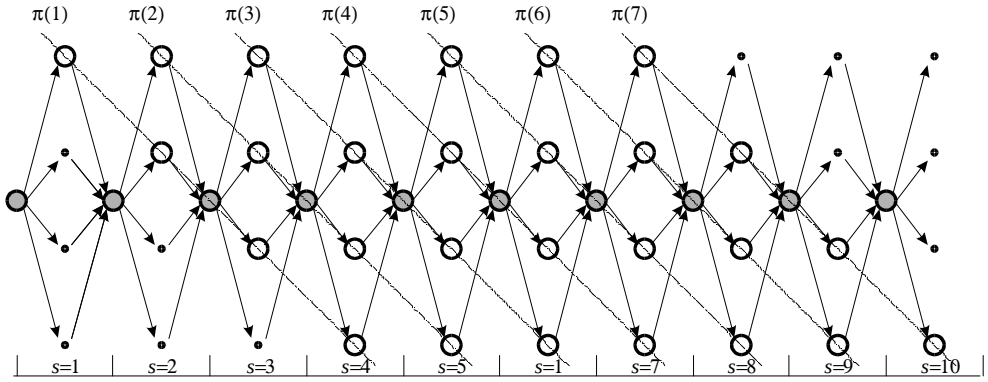
Zbiór węzłów $V=\{1,...,n+m\}\times\{1,...,m+1\}$ można podzielić na trzy rozłączne podzbiory: V^o- odpowiadające momentom rozpoczęcia wykonywania poszczególnych operacji zadań na maszynach, V^t- odpowiadające momentom rozpoczęcia wykonywania czynności transportowych, V^*- momentom rozpoczęcia czynności fikcyjnych, które zostały dodane do modelu w celu usunięcia warunków brzegowych i zwiększenia regularności grafu. Precyzyjne definicje zbiorów są następujące:

$$V^o = \bigcup_{s=1}^{m-1}\bigcup_{k=1}^{s}\{(s,k)\} \cup \bigcup_{s=m}^{n}\bigcup_{k=1}^{m}\{(s,k)\} \cup \bigcup_{s=n+1}^{s=n+m-1}\bigcup_{k=s-n-1}^{m}\{(s,k)\}, \tag{7}$$

gdzie:
węzeł $(s,k)\in V^o$ obciążony jest wagą $p_{\pi(s+1-k),k}$ i odpowiada momentowi rozpoczęcia wykonywania zadania wykonywanego na maszynie *k* w takcie produkcyjnym *s*,

$$V^t = \bigcup_{s=1}^{n+m-1}\{(s,m+1)\}, \tag{8}$$

węzeł $(s,k)\in V^t$ obciążony jest wagą *t* i odpowiada momentowi rozpoczęcia transportu w takcie produkcyjnym *s*,



Rys. 1. Struktura grafu $G(\pi)$

$$V^* = V \setminus (V^o \cup V^t)\,, \tag{9}$$

węzeł $(s,k) \in V^*$ obciążony jest wagą zero.

Zbiór łuków $E = E^o \cup E^i$ składa się z dwóch podzbiorów nieobciążonych łuków. Łuki ze zbioru

$$E^o = \bigcup_{s=1}^{n+m-1} \bigcup_{k=1}^{m} \{((s,m+1),(s,k))\}\,, \tag{10}$$

generowane są przez ograniczenia (4) oraz fikcyjne ograniczenia tego typu pochodzące od fikcyjnych operacji. Natomiast łuki ze zbioru

$$E^i = \bigcup_{s=1}^{n+m-1} \bigcup_{k=1}^{m} \{((s,k),(s,m+1))\}\,, \tag{11}$$

generowane są przez ograniczenia (5) oraz fikcyjne ograniczenia tego typu pochodzące od fikcyjnych operacji. Struktura grafu $G(\pi)$ nie zależy od permutacji ładującej. Od tej permutacji zależy jedynie obciążenie węzłów. Na Rys. 1 przedstawiono strukturę grafu dla systemu składającego się z m=4 maszyn, na których należy wykonać n=7 zadań. Węzły fikcyjne zostały istotnie zmniejszone, natomiast węzły odpowiadające obrotom systemu transportowego zostały zaznaczone szarym kolorem.

Opierając się na ogólnie znanych własnościach grafowych oraz po przeanalizowaniu struktury grafu $G(\pi)$, możemy sformułować następujące własności dla rozważanego problemu:

Własność 1: *Dla zadanej permutacji π, zdarzenie reprezentowane przez węzeł $(s,k) \in V$ może najwcześniej nastąpić w momencie równym długości najdłuższej drogi dochodzącej do danego węzła (bez obciążenia tego węzła) w grafie $G(\pi)$.*

Własność 2: *Najkrótszy czas realizacji wszystkich zadań wykonywanych w kolejności π jest równy długości najdłuższej drogi dochodzącej do węzła $(n+m,m)$ w $G(\pi)$ z obciążeniem tego węzła.*

Własność 3*: Długości najdłuższych dróg dochodzących do wszystkich węzłów można wyznaczyć w czasie $O((n+m)m)$.*

W przypadku dowodu własności 3 wystarczy zauważyć, że graf $G(\pi)$ jest grafem acyklicznym oraz składa się z $(m+n-1)(m+1)$ węzłów oraz $2m(n+m-1)$ łuków.

Analizując strukturę grafu $G(\pi)$ możemy zauważyć, że do każdego węzła reprezentującego operacje wykonywane na maszynach dochodzi tylko jeden łuk. Zatem najwcześniejszy moment zakończenia s-tego zadania w kolejności π wykonywanego na k−tej maszynie możemy wyrazić przy pomocy momentu zakończenia obrotu platformy w takcie bezpośrednio poprzedzającym w następujący sposób :

$$C_{\pi(s),k} = T_{s-k+1} + t + p_{\pi(s),k} \tag{12}$$

natomiast do każdego węzła reprezentującego operację transportową dochodzi m łuków od operacji wykonywanych w poprzednim takcie systemu. Najwcześniejszy moment rozpoczęcia obrotu w s-tym cyklu można wyznaczyć ze wzoru :

$$T_s = \max_{1\le k\le m} C_{\pi(s-k),k}\,, \tag{13}$$

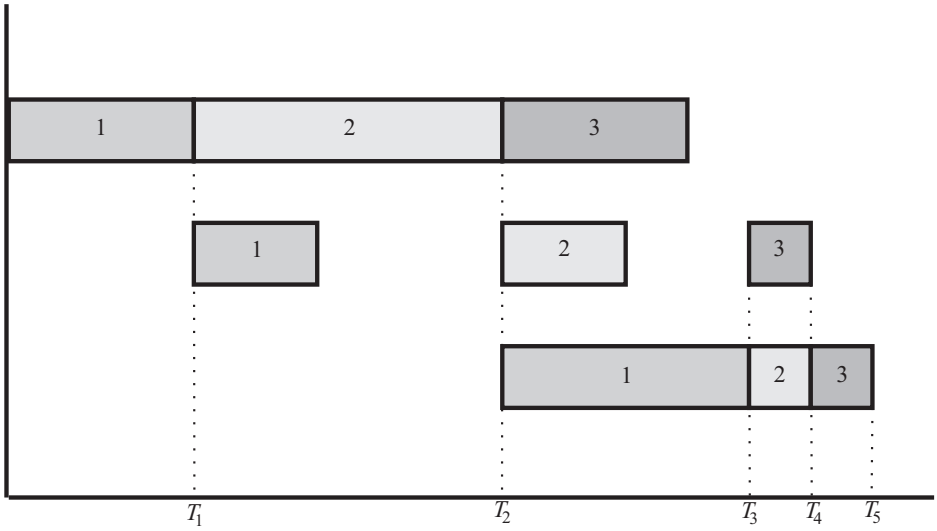
gdzie $s=2,...,n+m-1$ oraz $\pi(-m),...,\ \pi(0),\ \pi(n+1),...,\ \pi(n+m-1)=0$, $p_{0,k}=0$ dla $k=1,...,m$. Ostatecznie z (12) oraz (13) otrzymujemy wzór rekurencyjny :

$$T_s = T_{s-1} + t + \max_{1\le k\le m} p_{\pi(s-k),k}\,, \tag{14}$$

przy warunkach brzegowych takich jak w (13). Wielkości T_s można wyznaczyć w sposób nierekurencyjny wykonując obliczenia w kolejności $T_1, T_2,\ldots, T_{n+m}$.

Przykład 1. *W karuzelowym systemie produkcyjnym składającym się z $m=3$ maszyn należy wykonać $n=3$ zadania w kolejności 1,2,3. Czasy wykonywania zadań na maszynach wynoszą: 3,2,4 dla pierwszego zadania, 5,2,1 dla drugiego zadania oraz 3,1,1 dla trzeciego zadania.*

Dla $t=0$ (czas obrotu platformy), obroty platformy następują w chwilach: (*i*) $T_0=0$, (*ii*) $T_1=T_0+t+\max\{3,0,0\}=0+0+3=3$, (*iii*) $T_2=3+\max\{5,2,0\}=8$, (*iv*) $T_3=8+\max\{3,2,4\}=12$, (*v*) $T_4=12+\max\{0,1,1\}=13$, (*vi*) $T_5=13+\max\{0,0,1\}=14$. Na Rys. 2 przedstawiono harmonogram wykonywania zadań na maszynach w postaci wykresu Gantta.



Rys. 2. Struktura grafu $G(\pi)$

4. Algorytmy symulowanego wyżarzania

Rozpatrywany w pracy problem jest problemem NP-trudnym. W pracy [5] pokazano, że już jednomaszynowy problem tego typu jest problemem, dla którego nie istnieją wielomianowe algorytmy rozwiązania. W szczególnym przypadku tj. w systemie karuzelowym składającym się z dwóch maszyn technologicznych, w którym czasy załadunku produktów do systemu i ich wyładunku są pomijalnie małe rozwiązanie dokładne można uzyskać w czasie wielomianowym algorytmem z pracy [6].

Ze względu na NP-trudny charakter problemu oraz stosunkowo dużą liczbę zadań wykonywanych w rzeczywistych systemach produkcyjnych zrezygnowano z konstruowania algorytmu dokładnego. W literaturze można spotkać bardzo wiele metod konstruowania algorytmów heurystycznych dla kombinatorycznych problemów optymalizacyjnych.

Obecnie, najbardziej atrakcyjnymi dla badaczy i praktyków są metody przeszukiwań lokalnych. Są to metody uniwersalne i wymagają z reguły zdefiniowania tylko kilku komponentów, w których najistotniejszym jest definicja sąsiedztwa. W problemach, w których rozwiązanie reprezentowane jest w postaci permutacji, najczęściej spotykane są dwa typy sąsiedztw: sąsiedztwo wstaw (ang. inset) oraz sąsiedztwo zamień (ang. interchange). Sąsiedztwo typu wstaw składa się z wszystkich permutacji powstałych z bieżącej przez przesunięcie każdego zadania na każdą pozycję, natomiast w przypadku drugiego typu sąsiedztwa rozwiązania sąsiednie powstają przez zamianę pozycji każdego zadania z pozycją każdego innego zadania z permutacji.

W celu rozwiązania problemu zaproponowano algorytm oparty na metodzie symulowanego wyżarzania [7]. Metoda symulowanego wyżarzania należy do jednych z najefektywniejszych metod i jednocześnie jest stosunkowo prosta w implementacji. W algorytmie zastosowano metodę automatycznego doboru temperatury początkowej oraz adaptacyjną metodę wyznaczenia współczynnika chłodzenia (szczegóły można znaleźć w pracy [8]). Algorytm rozpoczynał działania od losowej permutacji początkowej. W danej

temperaturze algorytm wykonywał k iteracji, natomiast kończył swoje działanie po wykonaniu zadanej liczby iteracji lub wykryciu stagnacji obliczeń. W celu zwiększenia szans znalezienia rozwiązania dokładnego algorytm SA został uruchamiany wielokrotnie tj. *rep* razy. Algorytm symulowanego wyżarzania został zaimplementowany w dwóch wersjach: SA-INS z otoczeniem typu wstaw oraz SA-ICH z otoczeniem typu zamień.

5. Badania eksperymentalne

Celem badań eksperymentalnych była ocena efektywności zaproponowanych algorytmów oraz ocena restrykcyjności karuzelowego systemu wytwarzania. Algorytmy SA zostały przetestowane na 9-ciu grupach instancji zaproponowanych przez Taillarda [9]. W zestawie tym dla każdej pary $n \times m$: 20×5, 20×10, 20×20, 50×5, 50×10, 50×20, 100×5, 100×10, 100×20 znajduje się 10 trudnych przykładów testujących. Algorytmy SA-INS oraz SA-ICH zostały zaimplementowane w środowisku Visual Studio 2005 oraz uruchomione na komputerze HP Mobile Workstation z procesorem Intel Core Duo 2.6 GHz.

Każdy z algorytmów symulowanego wyżarzania był uruchamiany z liczbą wznowień obliczeń $rep \in \{1,2,3,5,10,15,20,50\}$. W każdym uruchomieniu wykonywał do 1000 iteracji przy czym w każdej temperaturze wykonywanych było $k=n^2/2$ kroków. Dla każdego przykładu wyznaczono najlepsze rozwiązanie $\pi^{SA\text{-}INS}$ wygenerowane przez algorytm *SA-INS* oraz najlepsze rozwiązanie $\pi^{SA\text{-}ICH}$ wygenerowane przez algorytm *SA-ICH*. W oparciu o wartości funkcji $C_{\max}$ wyznaczono najmniejszą z nich i przyjęto ją jako wartość referencyjną C^*. Następnie obliczono błąd względny rozwiązania $\pi \in \{\pi^{SA\text{-}INS}, \pi^{SA\text{-}ICH}\}$ z następującego wzoru :

$$RE(\pi) = \frac{C_{\max}(\pi) - C^*}{C^*} \cdot 100\% , \tag{15}$$

W Tab. 1 i 2 przedstawiono średnie błędy algorytmów oraz czasy ich działania dla różnej liczby uruchomień *rep*.

Z analizy rezultatów badań zebranych w Tab. 1 wynika, że zdecydowanie lepszym algorytmem okazał się algorytm *TS-ICH*. Dla *rep*=50 algorytm znalazł wszystkie najlepsze rozwiązania. Średni błąd drugiego z algorytmów, algorytmu *TS-INS*, dla tej samej liczby uruchomień *rep*=50 wynosi aż 3.5 %. Jego wartość rośnie wraz ze wzrostem liczby zadań, przy czym dla instancji o małej liczbie zadań (n=20) waha się on w okolicach 1 %, natomiast dla instancji o największej liczbie zadań (n=50) wynosi aż 5.0−6.5 %.

Przewaga *TS-ICH* nad algorytmem *TS-INS* jest jeszcze bardziej widoczna w zestawieniu kolumny wyników dla *rep*=1 uruchomienia algorytmu *TS-ICH* oraz *rep*=50 uruchomień algorytmu *TS-INS*. Średni błąd algorytmu *TS-ICH* wynosi 1.8 % i jest o 1.7 % mniejszy od średniego błędu drugiego algorytmu. Dla *rep*=10 średnie błędy algorytmu *SA-ICH* są mniejsze od średnich błędów algorytmu *SA-INS* dla *rep*=50 we wszystkich grupach instancji.

Wraz ze wzrostem liczby uruchomień *rep* zmniejsza się błąd algorytmów. W przypadku algorytmu *SA-ICH* najistotniejsze zmniejszenie obserwuje się dla *rep*=10. Średni błąd tego algorytmu zmienia się w granicach od 1.8−0.4%. Pięciokrotne zwiększenie liczby uruchomień pozwala na zmniejszenie błędu algorytmu tylko o 0.4%.

Tab. 1. Błąd algorytmów SA w zależności od liczby uruchomień

Grupa	Liczba uruchomień (*rep*)							
	1	2	3	5	10	15	20	50
Algorytm SA-ICH								
20×5	3,3	2,5	1,7	1,2	0,7	0,6	0,4	0,0
20×10	2,4	1,9	1,6	1,2	0,5	0,3	0,2	0,0
20×20	1,7	1,3	0,9	0,6	0,3	0,2	0,1	0,0
50×5	2,0	1,5	1,3	1,1	0,6	0,5	0,4	0,0
50×10	1,7	1,3	1,3	1,0	0,5	0,5	0,4	0,0
50×20	1,5	1,1	1,0	0,6	0,4	0,3	0,3	0,0
100×5	1,2	0,9	0,6	0,5	0,4	0,2	0,1	0,0
100×10	1,1	0,8	0,6	0,5	0,3	0,2	0,2	0,0
100×20	0,9	0,7	0,6	0,5	0,2	0,1	0,0	0,0
Średnio	1,8	1,3	1,1	0,8	0,4	0,3	0,2	0,0
Algorytm SA-INS								
20×5	4,1	3,4	2,9	2,5	1,7	1,3	1,3	0,9
20×10	4,3	3,5	3,2	2,6	2,1	1,7	1,5	1,2
20×20	3,4	2,8	2,2	2,0	1,8	1,6	1,5	1,1
50×5	5,9	5,6	5,4	4,8	4,3	4,1	4,0	3,2
50×10	6,6	6,3	6,2	6,0	5,6	5,2	5,1	4,6
50×20	5,5	4,5	4,5	4,4	3,7	3,7	3,5	3,1
100×5	7,4	6,8	6,6	6,1	6,0	5,6	5,6	5,5
100×10	8,2	8,2	7,7	7,5	7,1	6,9	6,7	6,5
100×20	6,6	6,3	6,1	5,7	5,3	5,1	5,0	5,0
Średnio	5,8	5,3	5,0	4,6	4,2	3,9	3,8	3,5

Czas działania algorytmów *SA-INS* i *SA-ICH* jest porównywalny (Tab. 2). W przybliżeniu rośnie on proporcjonalnie do kwadratu liczby zadań, proporcjonalnie do liczby maszyn oraz proporcjonalnie do liczby iteracji. Dla instancji najmniejszych (n=20) wynosi 0.1−0.2s dla jednego uruchomienia oraz 1.1−3.3s dla 50 uruchomień. Dla instancji o największej liczbie zadań czasy te wahają się odpowiednio w granicach 1.0−2.7s oraz 22.9−65.6s.

Tab. 2. Czas działania algorytmów SA w zależności od liczby uruchomień

Grupa	Liczba uruchomień (*rep*)							
	1	2	3	5	10	15	20	50
Algorytm SA-ICH								
20×5	0,1	0,0	0,1	0,1	0,2	0,5	0,5	1,1
20×10	0,1	0,1	0,1	0,2	0,4	0,8	1,0	2,0
20×20	0,2	0,1	0,2	0,4	0,6	1,4	1,7	3,3
50×5	0,3	0,2	0,3	0,6	1,3	2,2	3,2	6,4
50×10	0,5	0,4	0,6	1,2	2,2	3,5	4,7	10,8
50×20	0,7	0,6	0,9	1,9	3,6	5,7	7,6	17,2
100×5	1,0	0,9	1,3	2,5	4,8	7,8	9,8	22,9
100×10	1,9	1,5	2,3	4,3	9,6	12,6	16,8	40,9
100×20	2,7	2,5	3,6	8,1	15,0	20,5	25,8	65,6
Średnio	1,8	1,3	1,1	0,8	0,4	0,3	0,2	0,0
Algorytm SA-INS								
20×5	0,1	0,1	0,1	0,2	0,2	0,5	0,6	1,3
20×10	0,1	0,1	0,1	0,2	0,4	0,8	1,1	2,3
20×20	0,2	0,1	0,2	0,4	0,7	1,4	1,8	3,8
50×5	0,3	0,2	0,3	0,7	1,3	2,4	3,1	6,5
50×10	0,5	0,4	0,6	1,1	2,3	3,7	4,9	10,7
50×20	0,7	0,7	1,0	1,8	3,9	5,9	8,0	17,3
100×5	1,1	0,9	1,3	2,5	5,0	8,3	10,3	23,4
100×10	1,8	1,5	2,2	4,3	9,8	12,4	16,5	40,8
100×20	2,8	2,5	3,7	7,9	15,4	20,4	26,4	66,8
Średnio	5,8	5,3	5,0	4,6	4,2	3,9	3,8	3,5

Zastosowanie karuzelowej formy transportu w oczywisty sposób wprowadza okresy przestoju maszyn. Przedmiotem kolejnych badań było sprawdzenie przyrostu długości harmonogramu spowodowanego tymi przestojami. W tym celu wartości funkcji C_{max} dla rozwiązań wygenerowanych przez algorytm *SA-ICH* dla *rep*=50 dla systemu karuzelowego (*FS-CA*) zestawiono z wartościami funkcji C_{max} dla tych samych instancji dla systemu przepływowego z ograniczeniem bez magazynowania (*FS-NS*). Rozwiązania problemu *FS-NS* zostały zaczerpnięte z pracy [10]. Należy zauważyć, że probem *FS-NS* jest

Tab. 3. Porównanie wartości C_{max} dla różnych ograniczeń

FS-NS	*FS-CA*	*RE*	*FS-NS*	*FS-CA*	*RE*	*FS-NS*	*FS-CA*	*RE*
	20×5			50×5			100×5	
1384	1475	6,6	3151	3192	1,3	6455	6503	0,7
1411	1500	6,3	3348	3371	0,7	6214	6306	1,5
1293	1360	5,2	3173	3205	1,0	6124	6215	1,5
1448	1497	3,4	3277	3361	2,6	5976	6095	2,0
1348	1428	5,9	3272	3375	3,1	6173	6289	1,9
1363	1422	4,3	3330	3374	1,3	6094	6189	1,6
1381	1440	4,3	3168	3246	2,5	6262	6300	0,6
1384	1455	5,1	3228	3267	1,2	6061	6198	2,3
1378	1482	7,5	3068	3074	0,2	6474	6445	-0,4
1283	1359	5,9	3264	3308	1,3	6366	6445	1,2
Średnio		5,5			1,5			1,3
	20×5			50×5			100×5	
1698	1955	15,1	3776	4124	9,2	7320	7825	6,9
1836	2038	11,0	3641	4033	10,8	7108	7641	7,5
1674	1873	11,9	3588	4009	11,7	7233	7719	6,7
1555	1811	16,5	3786	4147	9,5	7413	7952	7,3
1631	1812	11,1	3745	4114	9,9	7168	7668	7,0
1603	1835	14,5	3747	4096	9,3	6993	7471	6,8
1629	1864	14,4	3778	4183	10,7	7092	7622	7,5
1741	1953	12,2	3708	4060	9,5	7143	7711	8,0
1759	1945	10,6	3668	4029	9,8	7327	7883	7,6
1782	1984	11,3	3729	4160	11,6	7299	7778	6,6
Średnio		12,9			10,2			7,2
	20×20			50×20			100×20	
2449	2916	19,1	4627	5352	15,7	8101	9248	14,2
2242	2743	22,3	4411	5161	17,0	8105	9306	14,8
2483	2968	19,5	4388	5199	18,5	8071	9259	14,7
2348	2849	21,3	4479	5244	17,1	8081	9320	15,3
2450	2983	21,8	4359	5184	18,9	8074	9351	15,8
2398	2833	18,1	4372	5173	18,3	8151	9438	15,8
2397	2919	21,8	4402	5242	19,1	8273	9378	13,4
2345	2806	19,7	4444	5193	16,9	8248	9447	14,5
2363	2865	21,2	4423	5204	17,7	8116	9418	16,0
2334	2871	23,0	4609	5279	14,5	8261	9451	14,4
Średnio		20,8			17,4			14,9

relaksacją problemu *FS-CA* co oznacza, że harmonogram optymalny dla *FS-NS* nie jest dłuższy od harmonogramu optymalnego *FS-CA*. Dla każdej instancji wyznaczono relatywne rozszerzenie *RE* harmonogramu:

$$RE = \frac{C_{\max}(\pi^{FS-CA}) - C_{\max}(\pi^{FS-NS})}{C_{\max}(\pi^{FS-NS})} \cdot 100\% \,. \tag{16}$$

Z zestawiania obliczeń w Tab. 3 wynika, że wraz ze wzrostem liczby maszyn relatywne rozszerzenie harmonogramu zwiększa się. Odwrotną tendencję obserwuje się w przypadku zwiększenia liczby zadań. Dla małej liczby zadań (n=20) przyrost sięga 23.0 %, natomiast dla największej liczby zadań spada do 16.0 %.

6. Podsumowanie

W pracy przedstawiono problem optymalizacyjny w przepływowym systemie karuzelowym. Zaproponowano opis matematyczny i model grafowy problemu. Na podstawie analizy grafu sformułowano szereg własności problemu, w szczególności przedstawiono prostą formułę rekurencyjną pozwalającą na wyznaczenie harmonogramu wykonywania zadań.

Do rozwiązania problemu zaproponowano wielowariantowy algorytm heurystyczny oparty na metodzie symulowanego wyżarzania SA. Algorytm SA poddano testom komputerowym na literaturowych instancjach problemu przepływowego. Z analizy rezultatów badań wynika, że zdecydowanie lepsze rozwiązania generowane są algorytmem SA z sąsiedztwem typu zamień.

Ponadto rezultaty badań jednoznacznie wskazują, że zastosowanie karuzelowego systemu transportowego w systemach produkcyjnych, w ogólnym przypadku, istotnie wydłuża czas realizacji zadań. Wydłużenie to zwiększa się wraz ze wzrostem liczby maszyn i zmniejsza wraz ze wzrostem zadań. Jednocześnie pokazują, że możliwe jest skonstruowanie efektywnego algorytmu optymalizacyjnego dla systemów karuzelowych nie w pełni zbalansowanych. Czas działania algorytmu, dla liczby zadań spotykanej w praktyce (100), nie przekracza 15 sekund.

Dodatkowe informacje

Praca częściowo finansowana z projektu badawczego MNiSW nr N N514 232237.

Literatura

1. Grunow M., Gunther H.O, Schleusener M., Yilmaz I.O.: Operations planning for collect-and-place machines in PCB assembly. Computers and Industrial Engineering 47, 2004, 409−429.
2. Ho W., Ji P.: Component scheduling for chip shooter machines: a hybrid genetic algorithm approach. Computers and Operations Research 30, 2003, 2175−2189.
3. Li S., Hu C., Tian F.: Enhancing optimal feeder assignment of the multihead surface mounting machine using genetic algorithms. Applied Soft Computing 8, 2008, 522−529.

4. Suer G.A.,Subramanian A., Huang J.: Heuristic procedures and mathematical models for cell loading and scheduling in a shoe manufacturing company. Computers and Industrial Engineering 56, 2009, 462−475.
5. Soylu B., Kirca O., Azizoglu M.: Flow shop-sequencing problem with synchronous transfer and makespan minimization. International Journal of Production Research, 2007, 45, 15, 3311-3331.
6. Gilmore P.C., Gomory R.E.: Sequencing a state-variable machine: a solvable case of the traveling salesman problem. Operations Research 12, 1964, 655–679.
7. Aarts E.H.L., Laarhoven P.J.M.: Simulated annealing: a pedestrian review of theory and some applications. Pattern Recognition and Applications. Eds. Deviijver P.A. and Kittler J., Springer, 1987.
8. Smutnicki C.: Algorytmy szeregowania. Akademicka Oficyna Wydawnicza, EXIT, Warszawa 2002.
9. Taillard E.: Some efficient heuristic methods for flow-shop sequencing. European Journal of Operational Research, 47, 1990, 65–74.
10. Grabowski J., Pempera J.: The permutation flowshop problem with blocking. A tabu search approach. Omega 35, 2007, 302–311.

Dr inż. Jarosław PEMPERA
Instytut Automatyki, Informatyki i Robotyki
Politechnika Wrocławska
50-372 Wrocław, ul. Wybrzeże Wyspiańskiego 27
tel./fax.: (71) 320-28-34
e-mail: jaroslaw.pempera@pwr.wroc.pl