ZESZYTY NAUKOWE POLITECHNIKI ŚLASKIEJ 2008
Seria: AUTOMATYKA z. 151 Nr kol. 1797

Jarosław PEMPERA
Politechnika Wrocławska

ALGORYTMY WSPOMAGAJĄCE PLANOWANIE CZYNNOŚCI TOWAROWANIA PRODUKCJI

Streszczenie. W pracy rozważany jest proces produkcyjny, który zaopatrywany jest dużą liczbą małych części. Zwykle producenci wykorzystują kuwety do ich transportu oraz organizacji technicznej procesu produkcyjnego. Aby zapobiec zwiększaniu czasu obsługi towarzyszącemu zmianie układu kuwet, ładowanie i rozładowywanie wózka transportowego odbywa się zgodnie z regułą LIFO. W pracy zaprezentowano model matematyczny problemu, efektywne algorytmy wyszukujące harmonogramy minimalizujące czas zakończenia wykonywania wszystkich zadań, jak również wyniki eksperymentu komputerowego.

ALGORITHMS AIDED SUPPLY OF MANAGEMENT DURING MANUFACTURING PROCESS

Summary. This paper considers the manufacturing process supplied with a lot of small parts. Usually, the manufacturers use tray for the transportation of the parts and organize the production. To avoid use of excessive time to reorder tray, the vehicle loading-unloading procedures should be performed in accordance LIFO rule. In the paper, there are presented mathematical model, efficient algorithms of finding schedule minimizing makespan as well as some experimental results.

1. Wstęp

Jednym z najistotniejszych ogniw w łańcuchu dostaw jest magazyn centralny. Jego sprawne funkcjonowanie ma decydujący wpływ na terminowość dostaw oraz koszty związane z magazynowaniem i/lub obsługą magazynu. Prace badawcze dotyczące usprawniania funkcjonowania magazynu dedykowane są metodom projektowania rozkładu magazynu [1] oraz magazynów wielopoziomowych [2], planowania tras magazynierów [3] oraz zarządzania wielkością zapasów. Obszerny przegląd literatury dotyczącej tej problematyki można znaleźć w pracy [4].

W niniejszej pracy rozważany jest proces produkcyjny, który zaopatrywany jest dużą liczbą małych części przewożonych w kuwetach przez pojedynczy wózek transportowy. Ładowanie i rozładowywanie kuwet odbywa się zgodnie z regułą LIFO.

2. Sformułowanie problemu i model matematyczny

W problemie towarowania produkcji należy wykonać n zadań ze zbioru $J=\{1,...n\}$ przy wykorzystaniu jednego operatora. Każe zadanie $j \in J$ składa się z trzech czynności (operacji): pierwsza operacja polega załadowaniu na wózek kuwety z częściami, druga na przewiezieniu do miejsca rozładunku, natomiast trzecia na rozładunku kuwety. Zadania wykonywane przez operatora dotyczą przewożenia kuwet z częściami do stanowisk, półproduktów pomiędzy stanowiskami oraz gotowych produktów do magazynu centralnego. Liczba wszystkich stanowisk obsługiwanych przez operatora wynosi m. Dla każdej czynności załadunku i rozładunku określony jest czas wykonywania i miejsca realizacji tych czynności, dane są również czasy przejazdu pomiędzy poszczególnymi stanowiskami (i/lub magazynem). Środkiem transportu można przewozić co najwyżej B kuwet jednocześnie, przy czym kuwety muszą być układane jedna na drugiej, natomiast rozładowywane w kolejności odwrotnej (reguła LIFO).

Kolejność wykonania wszystkich czynności realizowanych przez operatora można opisać za pomocą $2n$-elementowej permutacji elementów ze zbioru $O=\{1,...,2n\}$. Operację załadowania odpowiadającą zadaniu j, $j \in J$ będziemy numerowali indeksem j, natomiast rozładunku indeksem $j+n$. Dla każdej czynności $i \in O$ określone jest miejsce wykonania czynności m_i, $m_i \in \{1,...,m\}$ oraz czas wykonania czynności $p_i>0$. Czas przejazdu pomiędzy obsługiwanymi punktami k oraz l wynosi $d_{k,l} \geq 0$, $k,l \in \{0,1,...,m\}$. Magazynowi centralnemu (przedsionkowi) przyporządkowano numer 0.

Harmonogram wykonywania czynności operatora możemy opisać za pomocą terminów rozpoczęcia S_i i/lub zakończenia operacji C_i, $i \in O$. Czynności transportowe rozpoczynają się bezzwłocznie po zakończeniu operacji załadunku i/lub rozładunku. Kolejność wykonywania operacji π jest dopuszczalna ze względu na ograniczoną liczbę kuwet, którą można przewozić środkiem transportu, jeżeli dla każdego $i=1,...,2n$ spełniony jest następujący warunek:

$$|Z_i|-|R_i| \leq B, \tag{1}$$

gdzie: $Z_i=\{\pi(s):\pi(s) \leq n,\ s=1,...,i\}$, $R_i=\{\pi(s):\pi(s)>n,\ s=1,...,i\}$; oczywiście, liczby $|Z_i|$, $|R_i|$ są odpowiednio liczbą załadunków i rozładunków po wykonaniu i-tej czynności w kolejności π. Łatwo można sprawdzić, że kolejność realizacji operacji spełnia regułę LIFO, gdy spełniony jest następujący warunek:

$$ps(i)<ps(j)<ps(i+n) \Rightarrow ps(j+n)<ps(i+n),\ i,j=1,...,n, \tag{2}$$

gdzie: $ps(i)$ oznacza pozycję elementu i w π.

Dla dopuszczalnej kolejności π wykonywania czynności (spełniającej warunki (1) oraz (2)) harmonogram ich wykonywania musi spełniać następujące warunki:

$$S_i \geq 0,\ i \in O, \tag{3}$$

$$C_i=S_i+p_i,\ i \in O, \tag{4}$$

$$S_{\pi(1)} \geq d_{0,\pi(1)}, \tag{5}$$

$$S_{\pi(i)} \geq C_{\pi(i-1)} + d_{\pi(i-1),\pi(i)}, \quad i=2,3,\ldots,2n. \tag{6}$$

Dla dopuszczalnej kolejności π, najwcześniejszy moment zakończenia realizacji wszystkich operacji spełniający warunki (3-6), można wyznaczyć z następującego wzoru:

$$C_{\max}(\pi) = S_{\pi(2n)} + p_{\pi(2n)} + d_{\pi(2n),0}, \tag{7}$$

gdzie: $S_{\pi(i)} = S_{\pi(i-1)} + p_{\pi(i-1)} + d_{\pi(i-1),\pi(i)}$, $\pi(0)=0$, $S_0=0$, $p_0=0$. Odpowiednio przekształcając (3-7), otrzymujemy:

$$C_{\max}(\pi) = d_{0,\pi(1)} + \sum_{s=1}^{2n} t_s + \sum_{s=1}^{2n-1} d_{\pi(s),\pi(s+1)} + d_{\pi(2n),0}. \tag{8}$$

Niech Π będzie zbiorem wszystkich permutacji określonych na zbiorze $\{1,\ldots,2n\}$ spełniających warunki (1) i (2). Problem optymalizacji polega na wyznaczeniu takiej dopuszczalnej permutacji $\pi \in \Pi$, dla której $C_{\max}(\pi)$ będzie jak najmniejszy.

3. Algorytmy oparte na metodach popraw

Moc obliczeniowa współczesnych komputerów osobistych pozwala na konstruowanie efektywnych (tj. generujących dobre rozwiązania, w akceptowalnym przez praktyków czasie obliczeniowym) algorytmów opartych na metodach przeszukiwań lokalnych. W każdej iteracji, w tego typu algorytmach, dla pewnego rozwiązania zwanego bazowym definiuje się zbiór rozwiązań sąsiednich. Zbiór ten jest losowo próbkowany lub przeglądany w całości celem wyboru nowego rozwiązania bazowego dla następnej iteracji. Wybór rozwiązania zależy od zastosowanej metody, przy czym w większości metod rozwiązania lepsze od najlepszych dotąd znalezionych przyjmowane są w następnej iteracji. Jakość generowanych rozwiązań oraz czas działania algorytmów w głównej mierze zależą od definicji i liczności sąsiedztwa oraz sposobu jego przeglądania.

Jednym z najbardziej znanych i efektywnych ruchów dla problemów harmonogramowania jest ruch typu wstaw (*ang. insert*), który dla rozważanego problemu możemy zdefiniować za pomocą trójki $v=(j,x,y)$, $x<y$, który polega na usunięciu operacji j oraz $j+n$ z permutacji π i ich wstawieniu odpowiednio na pozycje x oraz y w tej permutacji. Ruch $v=(j,x,y)$ generuje nową permutację π_v z π w dwóch etapach:

I $\quad \gamma=(\pi(1),\ldots,\pi(a-1),\pi(a+1),\ldots,\pi(b),\pi(b+1),\ldots,\pi(2n))$,

II $\quad \pi_v=(\gamma(1),\ldots,\gamma(x-1),j,\gamma(x),\ldots,\gamma(y-2),j+n,\gamma(y-1),\ldots,\gamma(2n-2))$,

gdzie: a oraz b są pozycjami operacji j oraz $j+n$ w π.

Otoczenie rozwiązania bazowego π składa się ze zbioru permutacji otrzymanych przez wykonanie ruchów ze zbioru $V = \bigcup_{j \in u} V_j$, gdzie $V_j=\{(j,x,y): j \in J, x<y, x,y=1,\ldots,2n\}$. Zbiór V składa się z $n^2(2n-1)$ ruchów w zdecydowanej większości generujących rozwiązania niedopuszczalne.

Dla zadanego zadania j oraz pozycji x, po wykonaniu pewnych obliczeń wstępnych wymagających $O(n)$ czasu, można wykazać, że w czasie $O(1)$ możemy:

(*i*) wyznaczyć pierwszą oraz każdą następną dopuszczalną pozycję *y*, (*ii*) dla każdego dopuszczalnego ruchu wyznaczyć wartość funkcji celu rozwiązania sąsiedniego wynikającego z tego ruchu (bez konieczności generowania tego rozwiązania).

Bardzo duża liczba rozwiązań w sąsiedztwie znacznie ogranicza, ze względu na czas obliczeń, zastosowanie algorytmów przeszukujących całe otoczenie. Liczne badania dotyczące problemów harmonogramowania pokazują, że najefektywniejsze algorytmy próbkujące otoczenie oparte są na metodach termodynamicznych, w szczególności metodzie symulowanego wyżarzania SA [5] (ang. *simulated annealing*).

W metodzie tej nowe rozwiązanie jest akceptowane, staje się rozwiązaniem bazowym w następnej iteracji, bezwarunkowo w przypadku, gdy jest rozwiązaniem nie gorszym od rozwiązania bazowego lub z prawdopodobieństwem $p=\exp(-\Delta/T)$, gdzie: Δ jest różnicą wartości funkcji celu nowego rozwiązania i rozwiązania bazowego, natomiast T jest parametrem zwanym temperaturą, który jest zmniejszany zgodnie z przyjętym schematem chłodzenia. W danej temperaturze wykonuje się określoną liczbę iteracji. Wyróżniamy dwa podstawowe schematy: logarytmiczny ($T_{i+1}=T_i/(1+\lambda T_i)$) oraz geometryczny ($T_{i+1}=\lambda T_i$), gdzie T_0 jest temperaturą początkową, natomiast λ parametrem schematu chłodzenia. W pracy [5] zaproponowano metodę pozwalającą w sposób automatyczny wyznaczyć temperaturę początkową T_0 oraz współczynnik λ dla logarytmicznego schematu chłodzenia.

Dla problemów o dużej liczbie rozwiązań sąsiednich efektywność algorytmów opartych na metodzie SA można znacznie poprawić, stosując podział sąsiedztwa na podzbiory, w których znajduje się rozwiązanie najlepsze zwane reprezentantem [6]. Zredukowane sąsiedztwo algorytmu SA składa się wówczas z reprezentantów.

Niech zbiór $W_{j,x,p}=\{v=(j,x,y)$: $p\le y$, $y=1,\ldots,2n$, π_v–dopuszczalna$\}$, $j\in J$, $x<p$, $x,p=1,\ldots,2n-1$ będzie podziałem zbioru V. Odpowiadający temu podziałowi zbiór reprezentantów definiujemy w następujący sposób: $W=\{w_{j,x,p}$: $j\in J$, $x<p$, $x,p=1,\ldots,2n-1\}$, gdzie $w_{j,x,p}=\text{argmin}_{v\in W_{j,x,p}} C_{\max}(\pi_v)$ jest reprezentantem zbioru $W_{j,x,p}$ i oznacza najlepszy ruch spośród wszystkich ruchów wykonanych zadaniem j, polegających na wstawieniu operacji j na pozycję x, natomiast operacji $j+n$ na wszystkie dopuszczalne pozycje, począwszy od pozycji p. Natomiast drugi zbiór reprezentantów Z definiujemy na podstawie podziału $Z_j=\{v=(j,x,y)$: $x<y$, $x,y=1,\ldots,2n$, π_v–dopuszczalna$\}$, $j\in J$. Reprezentant $z_j=\text{argmin}_{v\in Z_j} C_{\max}(\pi_v)$ zbioru $Z_{j,x}$ oznacza najlepszy ruch spośród wszystkich ruchów wykonanych zadaniem j. Zauważmy, że złożoność obliczeniowa wyznaczenia dopuszczalnego rozwiązania w zbiorze V oraz reprezentanta w zbiorze W jest taka sama i wynosi $O(n)$, natomiast wyznaczenie reprezentanta ze zbioru Z wymaga $O(n^2)$ czasu.

4. Badania testowe algorytmów

Celem badań eksperymentalnych algorytmów był wybór najlepszego z proponowanych zbiorów ruchów. W tym celu zaimplementowano w języku C++ trzy algorytmy SA: SA(V), SA(W) i SA(Z) z otoczeniami generowanymi przez ruchy V, W oraz Z. Test algorytmów został przeprowadzony na komputerze z procesorem Pentium 1.5 GHz na dwunastu grupach składających się z 5 instancji o rozmiarach $n\times m\in\{10\times2$, 10×3, 10×5, 10×10, 20×2, 20×3, 20×5, 20×10, 50×2, 50×3, 50×5,

50×10} oraz wartościach $B \in \{2,3,5,10,15\}$. Wartości p_i oraz d_{ij} zostały wygenerowane wg rozkładu jednostajnego z przedziału [1,99]. W przypadku punktów obsługi magazyn centralny był wybierany z prawdopodobieństwem 0.5, natomiast pozostałe z prawdopodobieństwem $0.5/m$. W testowanych algorytmach SA zastosowano geometryczny schemat chłodzenia, parametry zostały wyznaczone automatycznie, w danej temperaturze wykonywanych było $n^2/4$ iteracji, podczas przebiegu algorytmu temperatura zmieniana była 1000 razy.

Dla każdego rozwiązania π wygenerowanego przez testowane algorytmy wyznaczono błąd względny $E(\pi)=100\%(C_{\max}(\pi)-C_{\max}(\pi^*))/C_{\max}(\pi^*)$, gdzie π^* jest najlepszym rozwiązaniem dla danej instancji wygenerowanym podczas badań algorytmów. Każdy z algorytmów dla każdej instancji uruchomiono 5 razy, dla każdego w ten sposób otrzymanego zestawu rozwiązań wyznaczono minimalny i średni błąd.

Tabela 1

Błąd względny algorytmów SA z różnym sąsiedztwem

B	Algor.	E	$n \times m$												ave
			10×2	10×3	10×5	10×10	20×2	20×3	20×5	20×10	50×2	50×3	50×5	50×10	
2	SA(V)	min	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,1	0,3	1,0	3,7	5,3	0,9
		ave	0,0	0,0	0,5	0,3	0,3	0,7	0,8	2,8	1,9	3,3	8,5	10,1	2,4
	SA(W)	min	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,1	0,0	0,1	0,8	1,4	**0,2**
		ave	1,8	0,0	0,7	0,0	0,0	0,2	0,7	1,2	0,5	1,3	4,3	6,0	**1,4**
	SA(Z)	min	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,2	0,7	0,3	0,1
		ave	0,0	0,0	0,0	0,1	0,7	0,2	0,3	0,8	0,5	1,5	4,7	3,5	1,0
3	SA(V)	min	0,0	0,0	0,0	0,0	0,0	0,2	0,3	3,4	3,8	5,0	11,9	9,3	2,8
		ave	0,0	0,0	0,0	0,3	3,4	0,0	2,6	7,4	7,2	9,5	20,1	17,5	5,7
	SA(W)	min	0,0	0,0	0,0	0,0	0,7	0,0	0,8	0,4	0,0	0,9	0,3	3,0	**0,5**
		ave	1,3	0,0	0,0	0,2	2,6	0,0	1,5	3,0	0,7	2,5	4,4	9,0	**2,1**
	SA(Z)	min	0,0	0,0	0,0	0,0	2,4	0,0	0,5	0,8	0,0	0,0	0,6	0,2	0,4
		ave	1,2	1,5	0,1	0,8	3,4	0,0	3,6	2,7	2,1	4,0	7,7	5,1	2,7
5	SA(V)	min	0,0	0,0	0,0	0,3	0,0	0,0	0,0	3,1	4,7	5,0	17,0	12,5	3,5
		ave	0,0	3,3	0,1	2,2	4,0	2,3	5,2	10,2	10,9	11,8	24,7	22,3	8,1
	SA(W)	min	0,0	0,0	0,0	0,0	3,8	0,0	1,2	1,9	0,0	0,9	0,0	1,7	**0,8**
		ave	4,5	3,9	0,2	0,6	4,4	2,4	3,2	5,7	2,1	4,6	7,0	9,7	**4,0**
	SA(Z)	min	0,0	0,0	0,0	0,0	15,0	0,0	0,4	0,0	0,0	4,6	5,3	2,2	2,3
		ave	0,2	3,6	0,3	0,5	15,7	16,6	7,9	5,0	5,3	12,0	12,2	9,0	7,4
10	SA(V)	min	0,0	0,0	0,0	0,3	0,0	0,2	2,1	0,5	13,8	3,4	16,5	15,5	4,4
		ave	0,0	3,2	0,4	1,1	0,9	2,0	4,8	8,6	24,6	24,4	37,0	27,9	11,2
	SA(W)	min	0,0	0,0	0,0	0,3	3,8	0,0	0,5	1,3	0,2	0,0	1,7	1,2	**0,8**
		ave	3,2	0,0	2,0	0,3	16,1	4,6	2,3	5,4	10,4	19,2	14,3	13,1	**7,6**
	SA(Z)	min	0,0	0,0	0,5	0,0	7,1	0,4	2,1	4,6	16,0	27,7	10,5	6,0	6,2
		ave	2,4	8,2	4,2	3,1	44,8	24,8	13,0	13,0	50,8	52,9	41,1	27,1	23,8
15	SA(V)	min	0,0	0,0	0,0	0,0	0,0	0,2	0,0	0,0	17,3	9,1	30,6	11,0	5,7
		ave	0,0	1,8	2,0	1,3	1,2	2,0	5,3	7,1	47,0	26,3	53,6	22,7	14,2
	SA(W)	min	0,0	0,0	0,0	0,0	3,8	0,0	0,0	4,6	0,0	0,0	0,0	2,8	**0,9**
		ave	6,4	2,6	0,5	0,9	4,0	0,7	4,6	7,3	2,3	4,0	13,0	10,7	**4,7**
	SA(Z)	min	0,0	0,0	0,5	0,0	30,5	8,6	2,1	6,0	57,9	40,6	28,8	4,2	14,9
		ave	0,0	1,2	1,7	1,0	48,9	23,1	9,5	10,5	97,8	59,1	44,5	12,7	25,8

W tabeli 1 przedstawiono uśrednione względem grup instancji wartości błędów algorytmów. Na podstawie otrzymanych wyników można stwierdzić, że najlepsze rozwiązania generuje algorytm SA bazujący na otoczeniu W. Minimalny z pięciu uruchomień błąd algorytmu SA(W) uśredniony względem wszystkich instancji dla danej wartości B nie przekraczał 1%, natomiast w przypadku SA(V) oraz SA(W) przyjmował on wartości odpowiednio 0.9–5.7 i 0.1–11.1. Biorąc pod uwagę średni błąd algorytmów, można zauważyć, że błędy tych algorytmów są jeszcze większe. Dla małej pojemności bufora B algorytm SA(Z) generuje nieznacznie lepsze rozwiązania niż SA(W). Jest to jednak okupione znacznie większym czasem obliczeń, który dla instancji o największych liczbie zadań n=50 wynosił 2.3 s, podczas gdy czas działania algorytmu SA(W) 1.5 s. Czas działania algorytmu SA(V) jest o połowę krótszy od czasu działania algorytmu SA(W).

BIBLIOGRAFIA

1. Larson N., March, H., Kusiak A.: A heuristic approach to warehouse layout with class-based storage. IIE Transactions (1997) 29, 1997, p. 337–348.
2. Yang L., Feng Y.: Fuzzy multi-level warehouse layout problem: New model and algorithm. Systems Engineering Society of China & Springer-Verlag 2006, p. 493–503.
3. Koster R., Poort E.: Routing orderpickers in a warehouse: a comparison between optima and heuristic solutions; IEE Transactions, 30, 1998, p. 469–480.
4. Berg J.P.: A literature survey on planning and control of warehousing systems. IIE Transactions 31, 1999, p. 751–762.
5. 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, Berlin.
6. Nowicki E., Smutnicki C.: A fast tabu search algorithm for the permutation flow shop problem. European Journal of Operational Research, 91, 1996, p. 160–175.

Recenzent: Dr inż. Jolanta Krystek

Abstract

In the many production systems, the overall problem is how to supply the manufacturing processes of a lot of small parts. Mostly, manufacturers use unified development tray/dish for the transportation (production). This paper considers the case when the first tray loads are placed in board of vehicle and the each new is picked up trays loaded. The loading-unloading procedures should be performed in accordance LIFO rule. This paper presents mathematical model, efficient algorithms of finding minimizing makespan of schedule as well as some experimental results.