Job shop scheduling

Opracowanie wybranego problemu optymalizacyjnego

Radosław Wojtczak, numer indeksu: 254607

28.05.2023

Spis treści

Wprowadzenie

1.1 Opis problemu

Job shop scheduling problem (JSSP) jest jednym z popularniejszych problemów optymalizacyjnych. Jego celem jest znalezienie optymalnego sposobu alokacji współdzielonych zasobów w czasie względem konkurujących ze sobą aktywności, tak aby czas realizacji aktywności był jak najmniejszy. W ogólności wskazany problem definiuje się w następujący sposób:

- liczba n określą liczbę prac (ang. jobs), które muszą zostać zrealizowane $(J_1, \ldots, J_n)$
- liczba m okreslą liczbę współdzielonych zasobów (często w odniesieniu do zasobów używa się określenia *maszyny*) $(M_1, \ldots, M_m)$
- celem jest utworzenie takiego harmonogramu, aby czas realizacji zadań był jak najmniejszy (czas ten najczęściej w literaturze określany jest mianem **makespan**)

Termin *job shop* pochodzi od charakterstycznego modelu produkcji, w którym różne zadania (jobs) przechodzą przez różne etapy produkcji w zakładzie produkcyjnym (shop). Na podstawie pochodzenia nazwy problemu łatwo zauważyć, iż jestem to problem, który wywodzi się bezpośrednio z potrzeb życia codziennego, stąd jest on jednym z pierwszych problemów kombinatorycznych, dla którego przeprowadzono analizę konkurencyjności (ang. competetive analysis) [7]
Ze względu na szeroką gamę zastosowań, na przestrzeni lat wytworzyło się wiele odmian wskazanego problemu. W podstawowych rozważaniach każde z zadań składa się ze zbioru k operacji $(O_1, \ldots, O_k)$ które muszą zostać wykonane w odpowiedniej kolejności (co znane jest jako *precedence constraint*). Każda z operacji wykonuje się określony czas oraz każda z maszyn może wykonywać jedną operację w danym momencie. Istnieje wiele odmian problemu, takie jak:

- *flexible job shop with duplicate machines* pozwalające na istnienie takich samych maszyn, czego naturalnym rozszerzeniem jest *flexible job*

shop, FJSSP który dopuszcza istnienie wielu (lub nawet wszystkich) identycznych maszyn, bądź identycznych grup maszyn

- odmiany, w których nakładane są ograniczenia na maszyny (wymuszenie przestojów, przerw w działaniu)
- odmiany, w których nakładane są ograniczenia na prace ($i - ta$ praca może rozpocząć się dopiero, gdy $j - ta$ praca się zakończy)

Przykładowe reprezentacje problemu

2.1 Graf rozłączny

Najpopularniejszy sposób reprezentacji problemu job shop scheduling wykorzystuje strukturę o nazwie **graf rozłączny** (ang.disjunctive graph) [3]. Graf rozłączny jest grafem skierowanym $G = (V, C \cup D)$, gdzie:

- V oznacza zbiór wierzchołków odpowiadający zadaniom (bądź operacjom, które wchodzą w skład zadań, w zależności od rozpatrywanej instancji problemu), który zawiera w sobie dodatkową, sztucznie wytworzoną parę wierzchołków: początkowy (*start*) oraz końcowy (*sink*). Oba wyróżnione wierzchołki mają zerowy czas wykonywania, istnieją jedynie po to, aby ustalić relację wykonywania zadań (zadanie startowe musi zostać wykonany przed wszystkimi innymi zadaniami, natomiast zadanie końcowe zostaje wykonane wtedy, gdy zostaną ukończone wszystkie pozostałe czynności).

- C to zbiór łuków łączących (conjunctive arcs), które odzwierciedlają początkowe ograniczenia kolejności, łączące każde dwie kolejne operacje tego samego zadania

- Nieskierowane krawędzie rozłączne należące do zbioru D łączą wzajemnie nieuporządkowane zadania, które wymagają tej samej maszyny do wykonania (krawędź rozłączna może być reprezentowana przez dwa przeciwstawne łuki skierowane). Każdy łuk jest oznaczony dodatnią wagą równą czasowi wykonania zadania, od którego łuk się zaczyna.

W modelu grafu rozłącznego (disjunctive graph), znalezienie optymalnego porządku wykonaia wszystkich zadań na maszynach równoważne jest z wyborem jednego łuku w każdej dysjunkcji, czyli zamienieniem każdej nieskierowanej krawędzi dysjunkcyjnej na skierowaną krawędź łączącą. Poprzez ustalenie kierunków wszystkich krawędzi dysjunkcyjnych, określa się kolejność wykonywania wszystkich konfliktujących zadań wymagających tej samej maszyny, co prowadzi do uzyskania kompletnego harmonogramu. Ponadto, uzyskany graf musi być acykliczny, a jeśli jest optymalny, to długość najdłuższej ścieżki od źródła do ujścia jest minimalna. Ta najdłuższa ścieżka określa *makespan*

2.2 Zdarzeniowy graf sieciowy

Graf rozłączny zawiera jedynie statyczne informacje o problemie, takie jak czasy trwania operacji, czy ograniczenia odnośnie kolejności, bez uwzględnienia dynamicznie zmieniających się informacji w ramach konstruowania harmonogramu. Zdarzeniowy graf sieciowy (Event-based Network Graph) wprowadza do wezłów grafu dodatkowe informacje, tak aby graf G reprezentował bieżący stan JSSP podczas planowania. [4] [13]

2.3 Model matematyczny

W celu reprezentacji problemu z rodziny job shop popularne jest wykorzystanie modelu matematycznego, który może być pomocny chociażby podczas próby rozwiązania wspomnianego zagadnienia przy pomocy technik programowania matematycznego, w którego skład wchodzi programowanie liniowe, czy programowanie całkowitoliczbowe. Zdefiowawszy następującą instancję problemu:
$jobs = \{1, \ldots, n\}$ - zbiór składający się z n zadań, m- liczba identycznych maszyn, (i, j)- relacja pierwszeństwa między zadanami, która oznacza, iż zadanie j może rozpocząć się dopiero wtedy, gdy zadanie j zostanie ukończone. Celem zadania jest znalezienie dopuszczalnego harmonogramu, który minimalizuje całkowity czas potrzebny do wykonania wszystkich zadań. Czas ten oznaczono jako c_max.
W poniższy sposób prezentuje się jej model programowania liniowego:

- Wymagany czas nie może być mniejszy niż suma rozpoczęcia ostatniej pracy na maszynie oraz czasu jej trwania

$$(\forall j \in jobs)(\sum_{t \in times} x_{jt} * (p_j + t)) \leqslant c\_max \tag{2.1}$$

- Każda praca może mieć tylko jeden moment rozpoczęcia

$$(\forall j \in jobs)(\sum_{t \in times} x_{jt} = 1) \tag{2.2}$$

- Zachowanie relacji pierwszeństwa między zadaniami

$$(\forall (j,k) \in precedence)(\sum_{t \in times} (p_j + t) * x_{jt} \leqslant \sum_{t \in times} t * x_{kt}) \tag{2.3}$$

- W jednej jednostce czasu nie może być więcej aktywny zadań niż dostepnych maszyn

$$(\forall t \in times)(\sum_{j \in jobs} \sum_{s=max\{0, t-p_j+1\}}^{t} x_{js}) \leqslant m \tag{2.4}$$

gdzie zbiór *times* oznacza horyzont zdarzeń, czyli dyskretny podział czasu na sloty, w których dana maszyna może obsłużyć zadanie, a

$$x_{jt} : j \in jobs, t \in times, x_{jt} \in \{0, 1\} \tag{2.5}$$

jest zmienną decyzyjną w formie macierzy przechowującej momenty rozpoczęcia każdego z zadań. Powyższy opis wraz z naturalną definicją funkcji celu jako minimalizacji zmiennej

$$min\ c\_max \tag{2.6}$$

tworzy model programowania liniowego dla pewnej odmiany problemu JSSP.

Zastosowania

3.1 Zarządzanie zadaniami na wielu procesorach

Instrukcje programu komputerowego można traktować jako zadania, a maszyny jako procesory w komputerze. Zgodnie z powyższym możemy założyć, że istnieją pewne zadania, które muszą zostać wykonanie jednocześnie na dwóch bądź więcej procesorach, co sprowadza się do ustalenia harmonogramu zadań tak, aby każdy z procesorów potrzebnych do realizacji danego zdania w danym momencie był dostepny oraz aby żaden procesor nie realizował dwóch zadań jednocześnie. Jest to perfekcyjny przykład, w którym można wykorzystać metody przedstawione w JSSP, co pokazują prace [5], [16]

3.2 Harmonogramowanie w kolejnictwie

Problem harmonogramowania w kolejnictwie można modelować jako problem job-shop poprzez podzielenie sieci kolejowej na segmenty torów. Jako zadanie traktuje się pociąg, który musi dostać się z pewnego miejsca początkowego do celu. Jako operacje uznaje się segmenty torów na trasie, co bezpośrednio wynika z chrakterystyki kolei- pociągi nie mogą wykonywać manewrów wymijania, więc istotnym jest, aby tylko jeden pociąg w danej jednostce czasu okupował daną część trasy. Czas przetwarzania operacji to czas potrzebny na przejazd przez segment toru. Cała problematyka sprowadza się do utworzenia harmonogramu w taki sposób, aby zmimializować czas, w którym pociągi oczekują na możliwość wjechania do odpowiedniego segmentu torów. [10]

3.3 Zarządzanie produkcją w fabryce

Ze względu na wielokrotne wykorzystanie analogii między maszynami a zadaniami, zarządzanie fabryką jest głównym obszarem zastosowania problemu optymalizacyjnego job shop. W kontekście zarządzania fabryką, problem job shop scheduling dotyczy efektywnego harmonogramowania zadań na maszynach w celu minimalizacji takich rzeczy jak czasu realizacji [20], zużycia energii [17] czy produkcji odpadów bądź zanieczyszczeń [1].

Rozwój metodologii rozwiązań

4.1 Programowanie matematyczne

Pierwsze prace nad problemem JSSP skoncentrowały się na rozwoju matematycznych modeli opisujących problem i opracowaniu algorytmów optymalizacyjnych. Jednym z wczesnych podejść było użycie metod programowania całkowitoliczbowego (integer programming) [19] [12], które opierały się na sformułowaniu problemu jako zadania minimalizacji pewnej funkcji celu przy pewnych ograniczeniach. Ponadto wykorzystano takie techniki rozwiązywania problemów jak programowanie liniowe (linear programming), programowanie liniowe całkowitoliczbowe (mixed-integer linear programming) czy programowanie ograniczeń (constraint programming) [6] [11]. Te metody umożliwiały bardziej precyzyjne modelowanie i rozwiązywanie problemu, uwzględniając dodatkowe ograniczenia i zależności między operacjami.

4.2 Heurystyki

Wraz z rozwojem technologii i rosnącą dostępnością komputerów, pojawiły się nowe metody rozwiązywania problemu JSSP. Algorytmy heurystyczne, takie jak przeszukiwanie lokalne (local search) i algorytmy genetyczne (genetic algorithms), [2] [15] zaczęły być wykorzystywane do rozwiązywania tego problemu. Metody te mogły generować satysfakcjonujące rozwiązania w krótszym czasie, choć niekoniecznie optymalne.

4.3 Metaheurystyki, algorytmy ewolucyjne

Wraz z postępem naukowym, zastosowanie innych metod, takich jak metaheurystyki, systemy wieloagentowe i algorytmy ewolucyjne, stało się popularne w rozwiązywaniu problemu JSSP [14] [9]. Te zaawansowane techniki algorytmiczne i sztucznej inteligencji były w stanie generować coraz lepsze i bardziej optymalne harmonogramy, uwzględniając różne cele i ograniczenia.

4.4 Sztuczna inteligencja

Obecnie, wraz z rozwojem technologii i wzrostem ilości dostępnych danych, popularne stało się wykorzystywanie uczenia maszynowego i sztucznej inteligencji do rozwiązywania problemu JSSP. Metody oparte na głębokim uczeniu (deep learning) są stosowane do generowania optymalnych harmonogramów, a także do automatycznego uczenia się z wcześniejszych rozwiązań. [18] [8] [21]

Podsumowanie

Job shop scheduling jest istotnym problemem kombinatorycznym, który znajduje swoje zastosowanie w wielu dziedzinach. Z tego względu wielu specjalistów poświęciło znaczną część swojego zawodowego życia aby rozwijać metody i technologie wykorzystywane przy tworzeniu nowych sposobów rozwiązań opisywanego problemu. Począwszy od opracowania matematycznych modeli, takich jak programowanie całkowitoliczbowe i programowanie liniowe, aż po rozwinięcie algorytmów optymalizacyjnych, heurystyk, czy wykorzystania uczenia maszynowego, możliwe stało się szybkie i skuteczne poszukiwanie optymalnych lub bliskich optymalnych rozwiązań. Osiągnięcia w dziedzinie job shop scheduling mają ogromne znaczenie dla różnych sektorów i branż, takich jak produkcja, logistyka, czy transport. Poprawa planowania i harmonogramowania zadań przekłada się na zwiększenie wydajności, redukcję czasu przestoju maszyn, minimalizację kosztów i poprawę jakości produkcji oraz środowiska, co bezpośrednio wpływa na lepsze i łatwiejsze funkcjonowanie ludzi w otaczającym ich świecie.

Bibliografia

[1] Giuseppina Ambrogio, Rosita Guido, Domenico Palaia, and Luigino Filice. Job shop scheduling model for a sustainable manufacturing. *Procedia Manufacturing*, 42:538–541, 2020. International Conference on Industry 4.0 and Smart Manufacturing (ISM 2019).

[2] Leila Asadzadeh. A local search genetic algorithm for the job shop scheduling problem with intelligent agents. *Computers & Industrial Engineering*, 85:376–383, 2015.

[3] Jacek Błażewicz, Erwin Pesch, and Małgorzata Sterna. The disjunctive graph machine representation of the job shop scheduling problem. *European Journal of Operational Research*, 127(2):317–331, 2000.

[4] Stéphane Dauzère-Pérès, Junwen Ding, Liji Shen, and Karim Tamssaouet. The flexible job shop scheduling problem: A review. *European Journal of Operational Research*, 2023.

[5] Maciej Drozdowski. Scheduling multiprocessor tasks — an overview. *European Journal of Operational Research*, 94(2):215–230, 1996.

[6] Soroush Fatemi-Anaraki, Reza Tavakkoli-Moghaddam, Mehdi Foumani, and Behdin Vahedi-Nouri. Scheduling of multi-robot job shop systems in dynamic environments: Mixed-integer linear programming and constraint programming approaches. *Omega*, 115:102770, 2023.

[7] R. L. Graham. Bounds for certain multiprocessing anomalies. *The Bell System Technical Journal*, 45(9):1563–1581, 1966.

[8] Yong Gui, Dunbing Tang, Haihua Zhu, Yi Zhang, and Zequn Zhang. Dynamic scheduling for flexible job shop using a deep reinforcement learning approach. *Computers & Industrial Engineering*, 180:109255, 2023.

[9] Hui Li, Xi Wang, and Jianbiao Peng. A hybrid differential evolution algorithm for flexible job shop scheduling with outsourcing operations and

job priority constraints. *Expert Systems with Applications*, 201:117182, 2022.

[10] Shi Qiang Liu and Erhan Kozan. Scheduling trains as a blocking parallel-machine job shop scheduling problem. *Computers & Operations Research*, 36(10):2840–2852, 2009.

[11] Leilei Meng, Chaoyong Zhang, Yaping Ren, Biao Zhang, and Chang Lv. Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem. *Computers & Industrial Engineering*, 142:106347, 2020.

[12] CHAO-HSIEN PAN. A study of integer programming formulations for scheduling problems. *International Journal of Systems Science*, 28(1):33–41, 1997.

[13] Junyoung Park, Jaehyeong Chun, Sang Kim, Youngkook Kim, and Jinkyoo Park. Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning. *International Journal of Production Research*, 59:1–18, 01 2021.

[14] Carlos A.S. Passos, Vitor M. Iha, and Rafael B. Dominiquini. A multi-agents approach to solve job shop scheduling problems using meta-heuristics. *IFAC Proceedings Volumes*, 43(17):198–203, 2010. 5th IFAC Conference on Management and Control of Production Logistics.

[15] Bo Peng, Zhipeng Lü, and T.C.E. Cheng. A tabu search/path relinking algorithm to solve the job shop scheduling problem. *Computers & Operations Research*, 53:154–164, 2015.

[16] Brucker Peter and Krämer Andreas. Shop scheduling problems with multiprocessor tasks on dedicated processors.

[17] Liji Shen, Stéphane Dauzère-Pérès, and Söhnke Maecker. Energy cost efficient scheduling in flexible job-shop manufacturing systems. *European Journal of Operational Research*, 2023.

[18] David Tremblet, Simon Thevenin, and Alexandre Dolgui. Predicting makespan in flexible job shop scheduling problem using machine learning. *IFAC-PapersOnLine*, 55(10):1–6, 2022. 10th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2022.

[19] Harvey M. Wagner. An integer linear-programming model for machine scheduling. *Naval Research Logistics Quarterly*, 6(2):131–140, 1959.

[20] Zigao Wu, Shudong Sun, and Shaohua Yu. Optimizing makespan and stability risks in job shop scheduling. *Computers & Operations Research*, 122:104963, 2020.

[21] Erdong Yuan, Shuli Cheng, Liejun Wang, Shiji Song, and Fang Wu. Solving job shop scheduling problems via deep reinforcement learning. *Applied Soft Computing*, 143:110436, 2023.