ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1998
Seria: AUTOMATYKA z. 123 Nr kol. 1389

Jacek BŁAŻEWICZ*, Erwin PESCH**, Małgorzata STERNA*
* Instytut Informatyki, Politechnika Poznańska
** Institut für Gesellschafts- und Wirtschaftwissenschaften, Universität Bonn

NOWA REPREZENTACJA MASZYNOWA GRAFU DYSJUNKCYJNEGO DLA PROBLEMU SZEREGOWANIA W OGÓLNYM SYSTEMIE OBSŁUGI

Streszczenie. W pracy przedstawiono nową reprezentację maszynową grafu dysjunkcyjnego dla problemu szeregowania w ogólnym systemie obsługi - macierz grafu, charakteryzującą się korzystną złożonością pamięciową i wysoką efektywnością czasową procedur ją obsługujących. Łączy ona zalety trzech klasycznych reprezentacji struktur grafowych: macierzy sąsiedztwa, listy poprzedników i listy następników, umożliwiając łatwy dostęp do różnego rodzaju informacji opisujących operacje w ogólnym systemie obsługi.

THE NEW MACHINE REPRESENTATION OF THE DISJUNCTIVE GRAPH FOR THE JOB SHOP SCHEDULING PROBLEM

Summary. This paper is concerned with a new time and memory efficient representation of the disjunctive graph - the graph matrix, used for describing instances of the job shop scheduling problem. The proposed data structure combines advantages of the classical graph representations like a neighborhood matrix and predecessors' and successors' lists delivering combined information on a job shop and enabling easy manipulation of the problem data.

1. Wstęp

Graf dysjunkcyjny [4] jest jednym z najpopularniejszych modeli wykorzystywanych do reprezentacji instancji problemu szeregowania w ogólnym systemie obsługi [1]. Problem ten polega na znalezieniu uszeregowania zbioru n zadań $J = \{J_1, ..., J_j, ..., J_n\}$ na zbiorze m dedykowanych maszyn $M = \{M_1, ..., M_k, ..., M_m\}$. Poszczególne zadania J_j składają się z uporządkowanego zbioru operacji, które muszą być wykonane w określonej kolejności przez określone maszyny. Poszczególne operacje $T_i \in J_1 \cup ... \cup J_n$ charakteryzują się czasem przetwarzania p_i i powinny być realizowane bez przerwań w sekwencji nie naruszającej ograniczeń problemu. Oprócz ograniczeń kolejnościowych, typu ($T_i \rightarrow T_j$) determinujących

kolejność wykonania par operacji T_i, T_j w zadaniu, klasyczna teoria szeregowania zadań wymaga, aby każda operacja była wykonywana przez co najwyżej jedną maszynę i każda maszyna realizowała co najwyżej jedną operację w danej chwili czasu. Celem szeregowania jest znalezienie dopuszczalnego i przeważnie optymalnego, ze względu na wybrane kryterium, przydziału operacji do maszyn.

Graf dysjunkcyjny, $G = (V, C \cup D)$, modelujący instancje problemu szeregowania zadań w ogólnym systemie obsługi, jest grafem skierowanym składającym się ze zbioru wierzchołków V oraz zbioru łuków C i krawędzi D. Poszczególne wierzchołki grafu odpowiadają operacjom w ogólnym systemie obsługi. Ponadto zbiór V zawiera dwie dodatkowe operacje o zerowym czasie przetwarzania: źródło i ujście, reprezentujące początek i koniec każdego uszeregowania. Poszczególne ograniczenia kolejnościowe modelowane są za pomocą łuków koniunkcyjnych tworzących zbiór C. Natomiast operacje ubiegające się o tę samą maszynę połączone są krawędziami dysjunkcyjnymi ze zbioru D (parą łuków o przeciwstawnym zwrocie). Ponadto wszystkie łuki posiadają wagi o wartości równej czasowi przetwarzania operacji, w której dany łuk ma swój początek. Uszeregowanie operacji w ogólnym systemie obsługi odpowiada ustaleniu zwrotu wszystkich krawędzi dysjunkcyjnych w grafie dysjunkcyjnym w sposób gwarantujący jego acykliczność.

Szeregowanie zadań w ogólnym systemie obsługi jest problemem silnie NP - zupełnym, którego optymalne rozwiązanie wymaga zastosowania metod o wykładniczej złożoności czasowej. Tym samym bardzo istotne staje się wykorzystanie efektywnej reprezentacji instancji problemu, czyli wydajnej reprezentacji maszynowej grafu dysjunkcyjnego, która umożliwia efektywne działanie metod rozwiązujących analizowane zagadnienie. W pracy przedstawiono nową reprezentację maszynową grafu dysjunkcyjnego w postaci macierzy grafu, charakteryzującą się korzystną złożonością pamięciową i wysoką efektywnością czasową procedur ją obsługujących, a także łatwym dostępem do różnego typu informacji opisujących system obsługi.

Struktura pracy jest następująca. W rozdziale 2 przedstawiono definicję macierzy grafu wraz z ilustrującym ją przykładem oraz analizą złożoności pamięciowej i czasowej. W rozdziale 3 zaprezentowano wyniki eksperymentu obliczeniowego, którego celem było porównanie nowej struktury z klasycznymi reprezentacjami grafu. Pracę zakończono sformułowaniem wniosków z przeprowadzonych badań, które ujęto w rozdziale 4.

2. Macierz grafu

Macierz grafu jest strukturą danych przeznaczoną do efektywnego reprezentowania grafu dysjunkcyjnego, modelującego problem szeregowania zadań, łączącą w sobie zalety klasycznych struktur danych, takich jak macierz sąsiedztwa wierzchołków oraz listy następników i poprzedników. W $(n+1)^2$ komórkach pamięci (gdzie $n+1$ oznacza liczbę operacji w systemie z uwzględnieniem źródła T_0 i ujścia grafu T_n) przechowywane są informacje o wzajemnym uporządkowaniu dowolnej pary operacji równocześnie umożliwiające przeglądanie list operacji poprzedzających wybraną operację, następujących po niej oraz listy operacji dotychczas nie uporządkowanych względem wybranej operacji.

Elementy macierzy grafu $G = [g_{ij}]_{(n+1)\times(n+1)}$ reprezentują zależności między poszczególnymi parami operacji T_i, T_j zgodnie z poniższą definicją ($0 < i < n$, $0 < j < n$).

$-n \leq g_{ij} < 0$ $\Leftrightarrow$ kolejność operacji T_i, T_j nie jest ustalona w grafie dysjunkcyjnym,

$0 \leq g_{ij} \leq n-1$ $\Leftrightarrow$ operacja T_j poprzedza operację T_i w grafie dysjunkcyjnym,

$n-1 < g_{ij} \leq 2*n-1$ $\Leftrightarrow$ operacja T_j następuje po operacji T_i w grafie dysjunkcyjnym.

Uwzględniając fakt, że źródło grafu T_0 poprzedza wszystkie pozostałe operacje w systemie, a ujście T_n następuje po wszystkich operacjach w systemie, elementy g_{0i}, g_{i0}, g_{in} i g_{ni} dla każdej operacji T_i mogą być wykorzystane do przechowywania dodatkowej informacji o rozważanym problemie. Podobnie, ponieważ graf nie zawiera pętli własnych dla poszczególnych wierzchołków T_i, elementy g_{ii} mogą służyć innym celom niż określenie relacji danej operacji względem niej samej. Wspomniane fragmenty macierzy grafu są wykorzystane do organizacji trzech struktur listowych dla poszczególnych operacji T_i: listy operacji poprzedzających T_i, następujących po T_i oraz listy operacji, których porządek wykonania względem T_i nie został jeszcze ustalony. Listy te stanowią integralną część macierzy grafu i są tworzone przez odpowiednie powiązanie jej elementów ze sobą, zgodnie z poniższymi regułami.

a) Lista poprzedników T_i - Indeks f pierwszej operacji T_f znajdującej się na liście poprzedników T_i zapamiętany jest w g_{i0}, a indeks l ostatniej operacji T_l z tej listy przechowywany jest w g_{0i}. Jeżeli $g_{i0} = g_{0i} = 0$, to omawiana lista jest pusta; w przeciwnym wypadku każdy jej element $g_{ij} = k$ przechowuje indeks k kolejnej operacji T_k do niej należącej. Dla ostatniego elementu listy T_l zachodzi $g_{ll} = l$.

b) Lista następników T_i - Indeks f pierwszej operacji T_f znajdującej się na liście następników T_i zapamiętany jest w g_{in}, a indeks l ostatniej operacji T_l w g_{ni}. Jeżeli $g_{in} = g_{ni} = 0$, to lista następników operacji T_i jest pusta; w przeciwnym wypadku każdy jej element

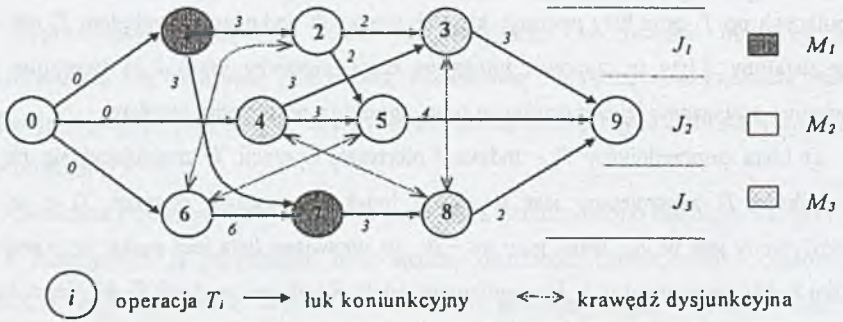
$g_{ij} = n\text{-}1+k$ przechowuje indeks k kolejnej operacji T_k do niej należącej powiększony o liczbę rzeczywistych operacji w systemie n-1, dzięki temu elementy listy następników są w pełni odróżnialne od elementów listy poprzedników. Listę kończy operacja T_l, dla której $g_{il} = n - 1+l$.

c) Lista operacji o nie ustalonej kolejności względem operacji T_i - Zanegowany indeks f pierwszej operacji T_f znajdującej się na wspomnianej liście zapamiętany jest w g_{ii} ($g_{ii} = -f$). Jeżeli $g_{ii} = -1$; analizowana lista jest pusta; w przeciwnym wypadku każdy z jej elementów $g_{ij} = -k$ przechowuje zanegowany indeks k kolejnej operacji T_k do niej należącej. Ponieważ elementy listy są liczbami ujemnymi, są one łatwo odróżnialne od elementów dwóch pozostałych list. Listę kończy operacja T_l, dla której $g_{il} = -l$.

Powyższa definicja macierzy grafu umożliwia zawarcie w jednej macierzy 3 różnych list operacji bez utraty informacji o wzajemnej relacji pomiędzy poszczególnymi parami operacji. Ponadto, definicja macierzy grafu może być łatwo dostosowana do specyficznych wymagań metody rozwiązującej problem szeregowania zadań w ogólnym systemie obsługi.

2.1. Przykładowa macierz grafu

Dany jest ogólny system obsługi składający się ze zbioru 3 maszyn $M=\{M_1, M_2, M_3\}$ i zbioru 3 zadań $J=\{J_1, J_2, J_3\}$ zdefiniowanych jako następujące łańcuchy operacji:
$J_1: T_1(M_1,3) \rightarrow T_2(M_2,2) \rightarrow T_3(M_3,3)$, $J_2: T_4(M_3,3) \rightarrow T_5(M_2,4)$, $J_3: T_6(M_2,6) \rightarrow T_7(M_1,3) \rightarrow T_8(M_3,2)$.
W nawiasach dla każdej operacji podano wymaganą maszynę $M(T_i)$ i czas przetwarzania p_i.



Rys.1. Graf dysjunkcyjny reprezentujący uszeregowanie częściowe
Fig.1. The disjunctive graph representing a partial schedule

Na rys.1. podano częściowe uszeregowanie operacji w przedstawionym ogólnym systemie obsługi powstałe w wyniku ustalenia zwrotu części krawędzi dysjunkcyjnych. Macierz grafu G reprezentującą powyższy graf dysjunkcyjny przedstawiono na rys.2.

g_{ij}	0	1	2	3	4	5	6	7	8	9
0	0	0	1	4	0	1	0	1	1	0
1	0	-4	11	15	-6	13	-6	16	13	2
2	1	1	-4	13	-6	13	-7	-8	-8	3
3	2	4	1	-5	4	-6	-7	-8	-8	0
4	0	-2	-6	11	-1	11	-7	-8	-8	5
5	4	1	1	-6	2	-3	-7	-8	-8	0
6	0	-2	-3	-4	-5	-5	-1	16	16	7
7	6	1	-3	-4	-5	-5	1	-2	16	8
8	7	1	-3	-4	-5	-5	1	6	-2	0
9	0	5	5	0	3	0	8	8	0	-9

Rys.2. Macierz grafu
Fig.2. Graph matrix

2.2. Złożoność pamięciowa i czasowa macierzy grafu

Macierz grafu jest wygodną reprezentacją grafu dysjunkcyjnego, łączącą w sobie zalety klasycznych reprezentacji maszynowych struktur grafowych [2]. Zajmuje ona $(n + 1)^2$ komórek pamięci, analogicznie jak macierz sąsiedztwa wierzchołków (tę reprezentację grafu z rys.1. przedstawiono na rys.3.), a proces pozyskiwania informacji o wzajemnej relacji między parami operacji jest realizowany również w stałym czasie (rzędu $O(1)$) poprzez weryfikację warunków przytoczonych na początku rozdziału 2.

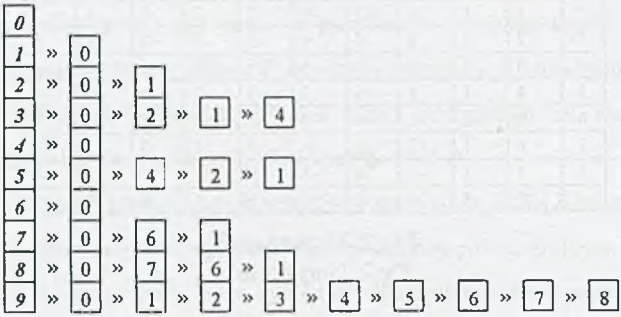
a_{ij}	0	1	2	3	4	5	6	7	8	9
0	0	1	1	1	1	1	1	1	1	1
1	-1	0	1	1	0	1	0	1	1	1
2	-1	-1	0	1	0	1	0	0	0	1
3	-1	-1	-1	0	-1	0	0	0	0	1
4	-1	0	0	1	0	1	0	0	0	1
5	-1	-1	-1	0	-1	0	0	0	0	1
6	-1	0	0	0	0	0	0	1	1	1
7	-1	-1	0	0	0	0	-1	0	1	1
8	-1	-1	0	0	0	0	-1	-1	0	1
9	-1	-1	-1	-1	-1	-1	-1	-1	-1	0

Rys.3. Macierz sąsiedztwa wierzchołków
Fig.3. Neighborhood matrix

Zachowując tę podstawową zaletę macierzy sąsiedztwa wierzchołków, nowa reprezentacja grafu gwarantuje również szybki dostęp do listy następników i poprzedników

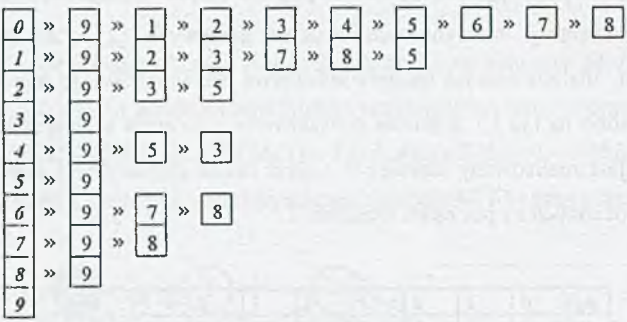
poszczególnych operacji. Nie wymaga bowiem, w odróżnieniu od macierzy sąsiedztwa, przejrzenia całego wiersza macierzy (czas rzędu $O(n)$), ponieważ poszczególne operacje poprzedzające lub następujące po wybranej operacji tworzą spójne ciągi równoważne dwóm innym klasycznym reprezentacjom grafu: liście poprzedników (rys.4.) i liście następników (rys.5.).

indeks operacji T_i » indeksy operacji poprzedzających operację T_i w grafie dysjunkcyjnym



Rys.4. Lista poprzedników
Fig.4. Predecessor list

indeks operacji T_i » indeksy operacji następujących po operacji T_i w grafie dysjunkcyjnym



Rys.5. Lista następników
Fig.5. Successor list

Obie listy zajmują $O(n + m)$ komórek pamięci (m oznacza liczbę łuków w grafie) przy założeniu ich implementacji w postaci dynamicznie alokowanych struktur danych, które jednak, jak wykazują eksperymenty obliczeniowe, charakteryzują się niską efektywnością czasową i wysoką zajętością pamięci. Ponadto, dysponując tylko jedną z list, uzyskuje się możliwość szybkiego dostępu do zbioru tylko jednego typu operacji poprzedzających albo następujących po wybranej operacji. Wyznaczenie zbioru komplementarnego wymaga już przejrzenia całej struktury danych. Macierz grafu nie wykazuje powyższej dysproporcji,

ponieważ obie listy poprzedników i następników są dostępne równocześnie, w ramach tej samej struktury danych. Dodatkowo, jak już wspomniano, sprawdzenie relacji pomiędzy parą operacji dokonywane jest w macierzy grafu w czasie stałym i nie wymaga przejrzenia całej listy operacji związanych z daną operacją relacją poprzedzania lub następowania, co w najgorszym przypadku zajmuje $O(m)$ jednostek czasu dla klasycznych reprezentacji listowych.

Kolejną zaletą macierzy grafu jest zapewnienie natychmiastowego dostępu do listy operacji o nie ustalonej kolejności wykonania, często wykorzystywanej podczas rozwiązywanie problemu szeregowania w ogólnym systemie obsługi Proces ten wymaga przejrzenia całej struktury danych w przypadku list następników i poprzedników lub całego wiersza macierzy sąsiedztwa wierzchołków.

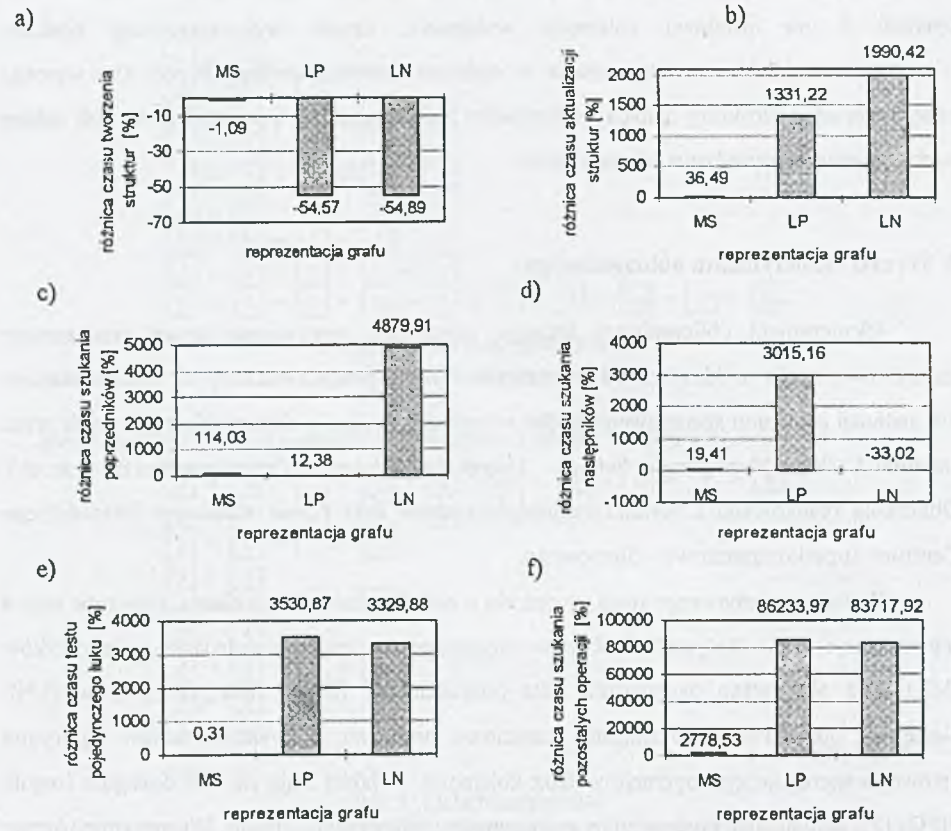
3. Wyniki eksperymentu obliczeniowego

Eksperyment obliczeniowy, którego celem było porównanie nowej reprezentacji maszynowej grafu z klasycznymi strukturami, został przeprowadzony z wykorzystaniem 30 instancji problemu szeregowania zadań w ogólnym systemie obsługi udostępnianych przez Imperial College Management School - University of London (http://mscmga.ms.ic.ac.uk). Obliczenia realizowano z wykorzystaniem komputera SGI Power Challenge Poznańskiego Centrum Superkomputerowo - Sieciowego.

Podczas pojedynczego testu, w oparciu o definicję instancji problemu, tworzone były 4 reprezentacje grafu: statycznie alokowane macierz grafu i macierz sąsiedztwa wierzchołków (MS) oraz alokowane dynamicznie lista poprzedników (LP) i lista następników (LN). Następnie, generowano rozwiązanie częściowe problemu, z wykorzystaniem algorytmu listowego szeregującego operacje według kolejności, w jakiej stają się one dostępne (reguła FIFO) [3], aktualizując poszczególne reprezentacje grafu dysjunkcyjnego. Wyznaczone różnice czasów tworzenia, aktualizacji i pozyskiwania różnego typu informacji między strukturami klasycznymi a macierzą grafu przedstawiono na rys.6.

Eksperymenty obliczeniowe wykazały, że czas tworzenia macierzy grafu jest porównywalny z czasem tworzenia drugiej ze struktur statycznych - macierzy sąsiedztwa wierzchołków (rys.6.a). Wydłużenie tego procesu o około 1% wynika z konieczności dodatkowego połączenia poszczególnych elementów macierzy grafu w 3 listy, które nie występują w klasycznej macierzy sąsiedztwa. Dynamiczne listy poprzedników i następników są inicjalizowane w czasie blisko o połowę krótszym, ponieważ zawierają informacje tylko o

jednym typie relacji między operacjami. Dalsze zarządzanie tymi strukturami (rys.6.b-f) jest jednak znacznie bardziej czasochłonne niż wykorzystywanie struktur statycznych. Ponadto, pomimo teoretycznie niskich wymagań pamięciowych, zajmowały one średnio o 83,81% bajtów więcej, które były konieczne dla przechowywania adresów kolejnych komórek tworzących listę.



Rys.6. Wyniki eksperymentu obliczeniowego
Fig.6. Computational experiment results

Wbudowanie w macierz grafu 3 różnych list dla poszczególnych operacji spowodowało, że aktualizacja grafu dysjunkcyjnego przebiega szybciej, niż obserwuje się to dla struktur klasycznych (rys.6.b). Ponadto, proces przeglądania wszystkich operacji poprzedzających wybraną operacje odbywa się co najmniej tak efektywnie jak z wykorzystaniem specjalizowanej do tego celu listy poprzedników (rys.6.c), a operacji następujących po

wybranej operacji równie efektywnie jak w oparciu o listę następników (rys.6.d). Macierz grafu nie preferuje jednak, w odróżnieniu od struktur listowych, żadnego z tych procesów, a także umożliwia bardzo szybkie sprawdzenie relacji pomiędzy parą operacji, analogicznie jak macierz sąsiedztwa (rys.6.e). Natomiast wyznaczenie wszystkich dotychczas nie uporządkowanych operacji z wykorzystaniem proponowanej macierzy odbywa się znacznie szybciej niż przy zastosowaniu klasycznych reprezentacji maszynowych grafu (rys.6.f).

4. Wnioski

Wyniki eksperymentu obliczeniowego dowodzą wysokiej efektywności nowej reprezentacji maszynowej grafu dysjunkcyjnego w porównaniu z klasycznymi strukturami danych. Gwarantuje ona szybki dostęp do różnego typu informacji opisującej ogólny system obsługi, nie preferując żadnej z nich, bez konieczności zwiększenia obszaru pamięci wykorzystywanego do zapamiętania kompletnego opisu grafu. Przy całej różnorodności sposobów udostępniania informacji o grafie dysjunkcyjnym, a tym samym o ogólnym systemie obsługi, tworzenie i aktualizowanie macierzy grafu realizowane jest przez bardzo proste procedury, łatwo dostosowywalne do specyficznych wymagań implementowanej metody.

Macierz grafu jest strukturą specjalizowaną do przechowywania informacji o grafie dysjunkcyjnym, wykorzystuje bowiem specyficzne cechy tego modelu ogólnego systemu obsługi. Przechowując początki poszczególnych list w postaci odrębnych wektorów, można macierz grafu wykorzystać do efektywnej reprezentacji dowolnych struktur grafowych.

LITERATURA

1. Błażewicz J., Ecker K., Pesch E., Schmidt G., Węglarz J.: Scheduling Computer and Manufacturing Processes, Springer, Heidelberg, 1996.
2. Deo N.: Graph Theory with Applications to Engineering and Computer Science, Prentice Hall, Inc., Englewood Cliffs, New Jersey, USA, 1974.
3. Haupt R.: A survey of priority rule-based scheduling, OR Spektrum, vol. 11, 1989, pp. 3-16.
4. Roy B., Sussmann B.: Les problémes d'ordonnancement avec constraintes disjoncties, SEMA, Note D.S., No 9, Paris, 1964.

Recenzent: Prof.dr hab.inż. Marek Kubale

Abstract

The work is concerned with a new time and memory efficient representation of the disjunctive graph which is a popular model used for describing instances of the job shop scheduling problem. Hence, the proper representation of the graph significantly influences the efficiency of an algorithm solving the considered problem. The proposed data structure for the disjunctive graph's description has the form of a graph matrix and combines advantages of classical graph representations enabling easy manipulation of the problem data. The graph matrix, of the size $n \times n$ where n is the number of tasks, delivers combined information on a job shop in four different ways: as a classical neighborhood matrix, as lists of predecessors, successors and moreover as a list of tasks for which no precedence relation has been disclosed during the solution process. Using this data structures it is possible to obtain information about the mutual order of any pair of tasks in a constant time and to get an immediate access to the mentioned three lists for each task. The computational experiments showed the superiority of the proposed machine representation of the graph over classical ones, especially with regard to the time of extracting information on different relations among tasks. The flexible definition of the graph matrix allows one to easily adjust this data structure for particular requirements of an implemented algorithm solving the job shop scheduling problem. In addition, notwithstanding it is a data structure specialized for the disjunctive graph model, it can be also used to represent any graph structure.