ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2000
Seria: AUTOMATYKA z. 129 Nr kol. 1474

Joanna JÓZEFOWSKA, Marek MIKA, Rafał RÓŻYCKI,
Grzegorz WALIGÓRA, Jan WĘGLARZ
Politechnika Poznańska

ALGORYTM SYMULOWANEGO WYŻARZANIA DLA PROBLEMU ROZDZIAŁU ZASOBÓW Z WIELOMA SPOSOBAMI WYKONYWANIA CZYNNOŚCI

Streszczenie. W pracy rozważa się problem rozdziału zasobów z wieloma sposobami wykonywania czynności. Celem jest znalezienie takiego uszeregowania dopuszczalnego, które minimalizuje czas trwania projektu. Zaproponowano algorytm symulowanego wyżarzania dla tego problemu oraz przedstawiono wyniki eksperymentu obliczeniowego.

A SIMULATED ANNEALING ALGORITHM FOR THE MULTI-MODE RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM

Summary. In this paper a multi-mode resource-constrained project scheduling problem is considered. The objective is to find a feasible solution which minimizes the makespan. A simulated annealing algorithm is developed to solve this problem and a performance analysis of this algorithm is presented on the basis of a comprehensive computational experiment.

1. Wprowadzenie

Rozważa się problem rozdziału zasobów z wieloma sposobami wykonywania czynności. Problem polega na uszeregowaniu zbioru czynności w taki sposób, aby czas trwania projektu był minimalny. Sposób wykonania czynności określa czas jej wykonania oraz niezbędną do tego ilość zasobu. Dla każdej czynności określa się skończony zbiór sposobów jej wykonania. Odpowiada to sytuacji często spotykanej w praktyce, gdy czas wykonania czynności można skrócić dzięki wykorzystaniu dodatkowych zasobów. Ze względu na rodzaj ograniczeń wyróżnia się trzy kategorie zasobów: odnawialne, nieodnawialne i podwójnie ograniczone. Zasób odnawialny to taki, dla którego tylko chwilowa dostępność jest ograniczona, a zasób nieodnawialny to taki, dla którego tylko zużycie jest ograniczone. Dla zasobów nieodnawialnych zarówno chwilowa dostępność, jak i zużycie są ograniczone. W przypadku zasobów dyskretnych (czyli przydzielanych w ilościach będących elementami zbiorów skończonych) można pominąć rozważanie zasobów podwójnie ograniczonych, gdyż każdy taki zasób można zastąpić parą zasobów: odnawialnego i nieodnawialnego. Pomiędzy czynnościami projektu zachodzą ograniczenia kolejnościowe.

Zakłada się, że czynności są niepodzielne oraz że nie można zmienić sposobu wykonywania czynności w trakcie jej realizacji. Celem jest znalezienie takiego zestawu sposobów wykonywania czynności oraz takich momentów rozpoczęcia wykonywania czynności, które minimalizują długość uszeregowania i nie naruszają ograniczeń kolejnościowych ani zasobowych. Problem ten jest oczywiście NP-trudny, a ponadto dla więcej niż jednego zasobu nieodnawialnego problem znalezienia rozwiązania dopuszczalnego jest, jak pokazano w [5], NP-zupełny.

Od wielu lat problem ten jest przedmiotem intensywnych badań, których głównym celem jest poszukiwanie jak najbardziej efektywnych algorytmów zarówno dokładnych, jak i heurystycznych. Podejścia dokładne z reguły opierają się na metodzie podziału i ograniczeń (np. [10]). Jednakże ze względu na złożoność obliczeniową odpowiednich algorytmów zasięg ich stosowania ogranicza się do problemów, w których liczba czynności nie przekracza 20. Wśród podejść heurystycznych można wyróżnić heurystyki oparte na regułach priorytetowych, metodę podziału i ograniczeń oraz metaheurystyki. Przegląd najważniejszych prac dotyczących problemu rozdziału zasobów z wieloma sposobami wykonywania czynności oraz metod jego rozwiązywania można znaleźć między innymi w pracy [1].

W niniejszej pracy przedstawiono algorytm oparty na innej metaheurystyce, mianowicie na symulowanym wyżarzaniu. W rozdziale 2 sformułowano rozważany problem w notacji zaproponowanej przez Bruckera i in. [1]. W rozdziale 3 zaprezentowano szczegóły dotyczące implementacji algorytmu, a rozdział 4 zawiera opis eksperymentu obliczeniowego przeprowadzonego na danych testowych z biblioteki PSPLIB. Wyniki podsumowano w rozdziale 5.

2. Sformułowanie problemu

Rozważany jest problem rozdziału zasobów z wieloma sposobami wykonywania czynności. Dany jest zbiór n niepodzielnych czynności. Dla uproszczenia rozważań wprowadza się dwie dodatkowe czynności pozorne: czynność początkową 0 (tzw. superźródło), stanowiącą początek całego uszeregowania oraz czynność końcową n+1 (tzw. superujście), stanowiącą koniec całego uszeregowania. Przez V oznacza się zbiór wszystkich czynności (łącznie z pozornymi). Pomiędzy czynnościami ze zbioru V zachodzą ograniczenia kolejnościowe reprezentowane przez graf ograniczeń kolejnościowych G=(V,E), gdzie E jest zbiorem relacji kolejnościowych. Graf G jest grafem skierowanym, acyklicznym, bez łuków przechodnich i jest przedstawiany w reprezentacji wierzchołkowej (czynności w wierzchołkach grafu). Przyjmuje się, że wierzchołki grafu są ponumerowane w porządku topologicznym. Każde ograniczenie kolejnościowe ze zbioru E jest reprezentowane w postaci $i \rightarrow j$, co oznacza, że wykonywanie czynności j nie może rozpocząć się przed zakończeniem czynności i. Ponadto Pred(j) oznacza zbiór wszystkich bezpośrednich poprzedników czynności j, a zbiór Succ(i) oznacza zbiór wszystkich bezpośrednich następników czynności i.

Niech R_ρ oznacza zbiór zasobów odnawialnych, a R_v zbiór zasobów nieodnawialnych. Liczność tych zbiorów wynosi odpowiednio R i N. Niech T oznacza horyzont planowania i stanowi jednocześnie górne ograniczenie czasu trwania projektu. Liczba jednostek zasobu odnawialnego k, k = 1, 2,..., R, dostępnych w dowolnej chwili t, t = 0, 1, 2,..., T wynosi R_k^ρ. Łączna liczba jednostek zasobu nieodnawialnego k, k = 1, 2,..., N, dostępnych w przedziale czasu [0, T], wynosi R_k^v.

Każda czynność j, j ∈ V musi być wykonana jednym ze sposobów należących do zbioru M_j, który stanowi zbiór alternatywnych sposobów wykonywania czynności j. Czas wykonania czynności j wykonywanej sposobem m, m ∈ M_j określony jest przez wartość p_{jm}. Każda czynność j, j ∈ V, wykonywana sposobem m, m ∈ M_j, wykorzystuje r_{jkm}^ρ jednostek zasobu odnawialnego k, k = 1, 2,..., R, i zużywa r_{jkm}^v jednostek zasobu nieodnawialnego k, k = 1, 2,..., N.

Żadna z czynności nie może być przerwana w trakcie wykonywania, a sposób wykonywania nie może ulec zmianie. Innymi słowy, jeśli czynność j, j ∈ V zacznie się wykonywać sposobem m, m ∈ M_j musi być wykonana do końca i bez przerw tym samym sposobem. Dla superźródła i superujścia przyjmuje się tylko jeden sposób realizacji, gdzie $p_{0,1}=p_{n+1,1}=0$; $r_{0,k,1}^\rho = r_{n+1,k,1}^\rho = 0$ dla k = 1, 2,..., R; $r_{0,k,1}^v = r_{n+1,k,1}^v = 0$ dla k = 1, 2,..., N. Zakłada się, że wszystkie czynności oraz zasoby są dostępne od chwili 0.

Niech S_j oznacza moment rozpoczęcia wykonywania czynności j. Zatem dla danego zestawu sposobów wykonywania czynności ciąg $S = (S_1, S_2,..., S_n)$ stanowi uszeregowanie. Przez S_T oznaczmy zbiór rozwiązań dopuszczalnych ze względu na ograniczenia kolejnościowe, a przez S_R zbiór uszeregowań dopuszczalnych zasobowo. Zatem zbiór $S = S_R \cap S_T$ stanowi zbiór uszeregowań dopuszczalnych. Niech C_j oznacza moment zakończenia zadania j, a $C = (C_1, C_2,..., C_n)$ ciąg momentów zakończenia zadań.

Problem polega na znalezieniu zestawu sposobów wykonywania czynności oraz uszeregowania $S = (S_1, S_2,..., S_n) \in S$, które minimalizują czas wykonania projektu $C_{max} = \max\{C_i\}$. Wszystkie wartości numeryczne są nieujemne i całkowitoliczbowe. Problem programowania matematycznego dla rozważanego problemu został sformułowany przez Talbota [12], a jego zmodyfikowana wersja została zaprezentowana przez Maniezzo i Mingozziego [9].

3. Algorytm symulowanego wyżarzania

Do rozwiązania analizowanego problemu wykorzystano metaheurystykę symulowanego wyżarzania. W przedstawionej w pracy implementacji tej metaheurystyki można wyróżnić pięć istotnych etapów: wprowadzanie danych, przetwarzanie wstępne, inicjalizacja algorytmu, proces chłodzenia oraz wyprowadzanie wyników. W fazie wprowadzania danych parametry problemu oraz algorytmu są wprowadzane ze

standardowego wejścia i/lub z odpowiedniego pliku. Przetwarzanie wstępne służy przygotowaniu zbioru danych instancji problemu do dalszego przetwarzania. Inicjalizacja algorytmu polega na ustaleniu wartości początkowych parametrów algorytmu symulowanego wyżarzania. Proces chłodzenia stanowi główną fazę rozważanego algorytmu, podczas której odwiedzane są kolejne rozwiązania w przestrzeni rozwiązań w celu znalezienia najlepszego z nich. Na zakończenie informacje dotyczące najlepszego znalezionego rozwiązania są wyprowadzane do pliku lub/i na standardowe wyjście.
Dalej będą przedstawione szczegóły dotyczące prezentowanej implementacji algorytmu i, o ile nie będzie zaznaczone inaczej, będą one dotyczyły całego algorytmu, a nie jego poszczególnych faz.

3.1. Przetwarzanie wstępne

Przetwarzanie wstępne jest pierwszą fazą algorytmu po wczytaniu danych instancji problemu. Procedura ta, zaproponowana przez Sprechera i in. [11], służy zmniejszeniu przestrzeni rozwiązań wszędzie tam, gdzie to możliwe i polega na usunięciu wszystkich niedopuszczalnych i nieefektywnych sposobów wykonywania czynności oraz redundantnych zasobów nieodnawialnych.

3.2. Reprezentacja rozwiązania

W algorytmie symulowanego wyżarzania w dowolnej chwili istotne są dwa rozwiązania: bieżące i sąsiednie. Oba rozwiązania powinny mieć taką samą reprezentację, aby przejście z jednego do drugiego z wykorzystaniem mechanizmu generacji sąsiedztwa jak i kryterium akceptacji było jak najprostsze. W rozważanej implementacji algorytmu symulowanego wyżarzania zbiór rozwiązań odpowiada zbiorowi uszeregowań kolejnościowo dopuszczalnych. A więc rozważane są również uszeregowania niedopuszczalne ze względu na zasób nieodnawialny. W takim przypadku niezbędne jest wprowadzenie mechanizmu, który umożliwia rozróżnienie uszeregowań dopuszczalnych od niedopuszczalnych. W niniejszej pracy dla tych celów zastosowano funkcję kary opisaną w punkcie 3.4.

W rozważanej implementacji algorytmu symulowanego wyżarzania rozwiązanie jest reprezentowane przez dwie n-elementowe listy: AL - listę czynności uporządkowanych kolejnościowo (tj. czynność nie może wystąpić przed żadnym z jej poprzedników) oraz MA – listę sposobów wykonywania czynności.

3.3. Reguła dekodująca

Reguła dekodująca służy do przekształcenia rozwiązania postaci i = (AL,MA) w uszeregowanie dopuszczalne ze względu na ograniczenia kolejnościowe i zasoby odnawialne. Wykorzystano sekwencyjny schemat generacji uszeregowania (SGS), który polega na szeregowaniu kolejnych czynności z listy AL wykonywanych sposobem określonym odpowiednim elementem listy MA w najwcześniejszym możliwym momencie. Musi być

zagwarantowana niepodzielność czynności oraz zachowanie ograniczeń kolejnościowych, jak i ograniczeń dotyczących zasobów odnawialnych. Dopuszcza się naruszenie ograniczenia dotyczącego zasobów nieodnawialnych, gdyż taki przypadek jest rozwiązywany przez wykorzystanie funkcji kary.

3.4. Funkcja kary

Funkcja kary jest stosowana dla rozwiązań, w których dla co najmniej jednego zasobu nieodnawialnego jego zużycie przekracza łączną liczbę jednostek tego zasobu. Dla danego rozwiązania i = (AL,MA) funkcja kary dana jest wzorem:

$$PF(i) = \sum_{k=1}^{N} \max\left\{0; \sum_{j=1}^{n} r_{jkm}^{v} - R_k^{v}\right\},$$

gdzie m jest sposobem wykonania czynności j określonym przez odpowiedni element listy MA. Jeżeli PF(i) = 0, to mamy do czynienia z uszeregowaniem dopuszczalnym również ze względu na ograniczenia dotyczące zasobów nieodnawialnych. Funkcja kary jest obliczana dla każdego rozwiązania, dla którego obliczana jest funkcja celu.

3.5. Funkcja celu

Funkcja celu jest obliczana dla każdego rozwiązania odwiedzanego w fazie chłodzenia. Wykorzystano funkcję celu zaproponowaną przez Hartmanna [3]. Dla danego rozwiązania i = (AL,MA) funkcja celu jest zdefiniowana następującym wzorem:

$$f(i) = \begin{cases} C_{\max} & \text{gdy } PF(i) = 0; \\ T + PF(i) & \text{w przeciwnym razie.} \end{cases}$$

3.6. Rozwiązanie początkowe

Rozwiązanie początkowe jest generowane w fazie inicjalizacji w następujący sposób. Lista AL jest tworzona przez ustawienie wszystkich czynności w porządku rosnącym wynikającym z topologicznego uporządkowania wierzchołków w grafie G, a lista MA przez ustawienie sposobu wykonywania wszystkich czynności na 1.

3.7. Mechanizm generacji sąsiedztwa

Rozwiązania sąsiednie generowane są z rozwiązania bieżącego przy użyciu jednego z trzech operatorów:

Przesunięcie czynności (*AS*), które działa wyłącznie na liście AL, polega na:

- losowym wyborze czynności $j \in V$ z listy AL;
- odnalezieniu na liście AL najbliższego poprzednika p oraz najbliższego następnika s;
- losowym wyborze pozycji x między czynnościami p i s;
- przesunięciu czynności j na pozycję x.

Zmiana sposobu wykonywania zadania (*MCh*) działa wyłącznie na liście MA i polega na:

- losowym wyborze pozycji k z listy MA;

- o ile to możliwe - zmianie sposobu wykonywania czynności k na inny losowo wybrany sposób.

Przejście złożone (*CM*) działa jednocześnie na obu listach i jest połączeniem obu wymienionych operatorów.

Wybór odpowiedniego operatora odbywa się drogą losową z określonymi prawdopodobieństwami, które wynoszą odpowiednio:

$P_{CM} = 0.5$; $P_{AS} = \left(\frac{1}{2} - \frac{|A|}{n\cdot(n-1)}\right)$; $P_{MCh} = 0.5 - P_{AS}$.

Prawdopodobieństwo dla operatora CM jest zawsze stałe bez względu na rozmiar oraz instancję problemu. Prawdopodobieństwa dla operatorów AS i MCh zależą od gęstości grafu G = (V,E) zgodnie z przedstawionym powyżej wzorem, gdzie |A| oznacza liczność zbioru łuków A w domknięciu przechodnim grafu G.

3.8. Schemat chłodzenia

Zastosowano „wielomianowy" adaptacyjny schemat chłodzenia opisany w [1]. Jedynym wyjątkiem jest zastosowanie innego kryterium stopu.

Początkowa wartość T_0 parametru kontrolnego obliczana w fazie inicjalizacji wynosi:

$$T_0 = \frac{\overline{\Delta f}^{(+)}}{\ln\left(\frac{m_2}{m_2\cdot\chi_0 - m_1(1-\chi_0)}\right)},$$

gdzie χ_0 jest początkową wartością współczynnika akceptacji określanego jako stosunek liczby przejść zaakceptowanych do liczby przejść wygenerowanych; m_1 jest liczbą przejść powodujących poprawę wartości funkcji celu, a m_2 jest liczbą przejść, dla których ta wartość nie ulega poprawie spośród wszystkich m_0 przejść wygenerowanych w fazie inicjalizacji $m_0 = m_1 + m_2$. $\overline{\Delta f}^{(+)}$ jest średnią różnicą wartości funkcji celu dla wszystkich m_2 przejść, dla których wartość funkcji celu ulega poprawie. Założono, że $\chi_0 = 0.95$ a $m_0 = 50$.

Następna wartość parametru kontrolnego T_{k+1} obliczana jest ze wzoru:

$$T_{k+1} = \frac{T_k}{1 + \frac{T_k\cdot\ln(1+\delta)}{3\cdot\sigma_{T_k}}}, \quad k = 0, 1, \ldots,$$

gdzie σ_{T_k} jest odchyleniem standardowym wartości funkcji celu dla k-tego łańcucha Markowa, a δ jest pewną liczbą rzeczywistą (zwykle małe wartości tego parametru prowadzą do lepszych rozwiązań kosztem czasu obliczeń). Przyjęto, że $\delta = 0.5$.

Długość łańcuchów Markowa w jednorodnej wersji algorytmu symulowanego wyżarzania określa liczbę przejść wygenerowanych przy stałej wartości parametru kontrolnego. Przyjęto, że wielkość ta jest stała i zależna od rozmiaru problemu wg następującej reguły:

$L_k = L = 5 \cdot n$, $k = 0, 1, \ldots$, gdzie n jest liczbą czynności.

Kryterium stopu jest spełnione, jeśli zachodzi jeden z dwóch przypadków:

- znaleziono dopuszczalne uszeregowanie, którego długość jest równa MPM (długość ścieżki krytycznej dla problemu rozdziału zasobów z wieloma sposobami wykonywania czynności obliczana przy założeniu, że wszystkie czynności są wykonywane sposobami o najkrótszych czasach wykonywania z pominięciem ograniczeń zasobowych);
- odwiedzono założoną z góry liczbę rozwiązań, która wynosi 12 000·n, gdzie n jest liczbą czynności.

4. Eksperyment obliczeniowy

Eksperyment obliczeniowy został przeprowadzony dla wszystkich zbiorów instancji dostępnych w internetowej bibliotece testowych danych dla rozważanego problemu. Biblioteka ta nosi nazwę PSPLIB i jest powszechnie uznawana za najlepsze źródło instancji dla problemu rozdziału zasobów. Wszystkie dane zawarte w tej bibliotece zostały wygenerowane według klucza za pomocą programu o nazwie ProGen. Biblioteka PSPLIB jest dostępna za pomocą protokołu FTP pod adresem ftp.bwl.uni-kiel.de/pub/operations-research/psplib lub z witryny WWW o adresie http://www.bwl.uni-kiel.de/Prod/psplib/index.html. Szczegóły dotyczące zarówno biblioteki PSPLIB, jak i programu ProGen można odnaleźć w pracach [6],[7] i [8].

Rozważany algorytm został zaimplementowany i skompilowany w MS Visual C++ 5.0. Wszystkie testy zostały przeprowadzone na komputerze klasy PS z procesorem Pentium 133MHz i 32MB RAM w środowisku systemu Windows NT 4.0. Ponieważ algorytm symulowanego wyżarzania jest algorytmem losowym, wykorzystano wbudowany w kompilator języka MS Visual C++ generator liczb pseudolosowych. Każdorazowe wywołanie programu dla dowolnej instancji powodowało zainicjowanie generatora liczb pseudolosowych wartością 1.

Wykorzystano następujące zbiory instancji dostępne w PSPLIB: J10, J12, J14, J16, J18, J20, J30, M2, M4, M5, C15, C21, N0, N1, N3, R1, R3, R4 i R5. Dzięki temu można było zaobserwować wpływ poszczególnych parametrów problemu na jakość otrzymywanych wyników, które przedstawiono w tabeli 1. W tabeli tej przedstawiono średni i maksymalny czas obliczeń w sekundach oraz następujące wartości:

- średnie względne odchylenie od optimum (ŚWO) wyrażone w procentach;
- maksymalne względne odchylenie od optimum (MWO) wyrażone w procentach;
- maksymalne bezwzględne odchylenie od optimum (MBO);
- procent znalezionych rozwiązań dopuszczalnych (PRD);
- procent znalezionych rozwiązań optymalnych (PRO).

Tablica 1

Wyniki eksperymentu obliczeniowego

Zbiór instancji	Czas obliczeń		ŚWO	MWO	MBO	PRO	PRD
	średni	maksymalny					
J10	7.0	14.5	0.20	22.2	4	97.6	100.0
J12	10.8	22.0	0.36	134.7	66	97.8	99.8
J14	14.6	30.3	0.21	22.2	8	96.0	100.0
J16	18.3	41.4	0.16	23.5	8	97.3	100.0
J18	24.0	55.3	0.15	21.6	3	96.6	100.0
J20	30.6	75.0	0.15	11.1	2	95.8	100.0
J30*	72.2	200.2	0.34	6.9	3	88.9	100.0
C15	18.1	43.1	0.08	6.9	2	97.8	100.0
C21	19.9	60.3	0.05	8.3	3	98.7	100.0
R1	13.5	34.7	0.08	6.9	2	98.2	100.0
R3	23.9	50.5	0.13	6.9	2	96.8	100.0
R4	28.0	56.4	0.12	7.7	3	96.9	100.0
R5	32.8	81.3	0.13	7.4	2	96.5	100.0
N0	13.9	53.4	0.03	6.7	1	99.4	100.0
N1	19.3	40.7	0.06	5.7	2	98.6	100.0
N3	18.4	43.2	2.81	367.9	103	94.8	99.2
M2	18.9	46.5	0.09	18.2	4	98.8	100.0
M4	19.5	41.5	1.08	286.5	106	95.7	99.6
M5	20.3	42.9	1.06	210.6	99	93.2	99.6

*) Dla zbioru instancji J30 dokonano porównania z najlepszymi znanymi rozwiązaniami ze względu na to, że rozwiązania optymalne nie są znane.

Jeżeli dla któregokolwiek zestawu danych testowych wartość PRD jest mniejsza niż 100%, oznacza to, że dla jednej lub więcej instancji algorytm nie znalazł dopuszczalnego uszeregowania. W takich przypadkach wartości MBO i MWO też są odpowiednio duże. Bierze się to stąd, że wartość funkcji celu została powiększona o wartość kary.

Otrzymane rezultaty pokazują dużą stabilność algorytmu. Procent znalezionych rozwiązań optymalnych jest duży dla wszystkich zbiorów instancji problemu. Algorytm tylko w 10 na 8883 przypadkach nie znalazł uszeregowania dopuszczalnego. Analiza wpływu poszczególnych parametrów problemu na jakość otrzymanych wyników pokazuje, że najistotniejszy wpływ na jakość otrzymanych wyników mają: liczba zasobów nieodnawialnych, której wzrost powoduje zmniejszenie liczby znalezionych rozwiązań optymalnych (patrz N0, N1, J16, N3 w tabeli 1) oraz liczba sposobów wykonywania czynności, której wzrost również powoduje pogorszenie otrzymanych wyników (patrz M2, J16, M4, M5 w tabeli 1). Pozostałe parametry problemu, takie jak: liczba czynności, liczba zasobów odnawialnych oraz gęstość grafu ograniczeń kolejnościowych nie mają większego wpływu na jakość otrzymanych wyników.

Za najlepszy znany dotychczas algorytm genetyczny dla rozpatrywanego problemu uchodzi algorytm opisany w [3], gdzie podano wyniki eksperymentu obliczeniowego dla

zbiorów instancji J10 do J20, gdy czas obliczeń wynosił 1 sek. W porównaniu z tymi wynikami przedstawiony w tej pracy algorytm symulowanego wyżarzania znajduje więcej rozwiązań optymalnych i wykazuje mniejsze średnie odchylenie od optimum dla instancji zawierających powyżej 14 czynności.

5. Podsumowanie

W pracy przedstawiono algorytm symulowanego wyżarzania zaadaptowany do problemu rozdziału zasobów z wieloma sposobami wykonywania czynności. Wydajność tego algorytmu została potwierdzona na podstawie obszernego eksperymentu obliczeniowego korzystającego ze zbioru instancji testowych dostępnych w bibliotece PSPLIB. Wysoki procent znalezionych rozwiązań optymalnych oraz niska wartość średniego odchylenia względnego od optimum pokazują, że opracowany algorytm jest jednym z najlepszych algorytmów służących rozwiązywaniu rozważanego problemu. Pokazano również, że parametry problemu mają raczej niewielki wpływ na jakość otrzymanych rozwiązań z dwoma wyjątkami, gdy zwiększenie liczby zasobów nieodnawialnych bądź zwiększenie liczby sposobów wykonywania zadań powoduje nieznaczne pogorszenie otrzymanych wyników.

LITERATURA

1. Aarts E., Korst J.: Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing, Wiley, Chichester, 1989.
2. Brucker P., Drexl A., Möhring R., Neumann K., Pesch E.: Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research, 112 (1999) 3-41.
3. Hartmann S.: Project Scheduling with Multiple Modes: A Genetic Algorithm. Manuskripte aus den Instituten für Betriebswirtschaftslehre, No.435, University of Kiel, Germany, 1997.
4. Józefowska J., Mika M., Różycki R., Waligóra G., Węglarz J.: *Solving discrete-continuous project scheduling problem via its discretization*, Mathematical Methods of Operations Research, spcial issue on Project Scheduling (przyjęte do druku).
5. Kolisch R.: Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. European Journal of Operational Research, 90 (1996), 320-333.
6. Kolisch R., Schwindt Ch., Sprecher A.: Benchmark instances for project scheduling problems, in J. Węglarz (Ed.) Project Scheduling: Recent Models, Algorithms and Applications, Kluwer Academic Publishers, Dordrecht, 1999, pp. 195-212.
7. Kolisch R., Sprecher A.: PSPLIB - a project scheduling problem library, European Journal of Operational Research, 96 (1996), 205-216.
8. Kolisch R., Sprecher A., Drexl A.: Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science, 41 (1995), 1693-1703.

9. Maniezzo V., Mingozzi A.: A heuristic procedure for the multi-mode project scheduling problem based on Bender's decomposition, in: J. Węglarz (Ed.) Project Scheduling: Recent Models, Algorithms and Applications, Kluwer Academic Publishers, Dordrecht, 1999, pp. 179-196.
10. Sprecher A., Drexl A.: Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. European Journal of Operational Research, 107 (1998), 431-450.
11. Sprecher A., Hartmann S., Drexl A.: An exact algorithm for project scheduling with multiple modes, OR Spektrum, 19 (1997), 195-203.
12. Talbot F. B.: Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case, Management Science, 28 (1982), 1197-1210.

Recenzent: Prof.dr hab.inż. K.Wala

Abstract

In this paper a multi-mode resource-constrained project scheduling problem is considered. In this problem activities can be executed in one of several modes. Moreover, each activity may not be preempted and a mode once selected may not be changed during the execution of this activity. Two types of resources are considered: renewable and non-renewable ones. The objective is to find an assignment of modes to activities as well as precedence- and resource-feasible starting times for all activities such that the makespan of the project is minimized. The problem is NP-hard. Moreover, for more than one non-renewable resource the problem of finding a feasible solution is already NP-complete A simulated annealing algorithm is proposed to solve this problem. Some problem specific parameters of the developed algorithm are presented. The solution is represented by two n-element lists: a precedence feasible list of activities and a mode assignment. A neighbour solution is generated using one of the three neighbourhood generation mechanism operators. A performance analysis is made on the basis of a comprehensive computational experiment performed on a set of standard benchmark instances from project scheduling problem library PSPLIB. The obtained results show that the presented algorithm is very stable for different settings of problem parameters and gives high percentage of optimal solution found for all sets of instances considered.