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

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

OPTYMALIZACJA PRZYDZIAŁU ZASOBÓW CIĄGŁYCH I DYSKRETNYCH W KOMPLEKSIE OPERACJI NIEPODZIELNYCH

Streszczenie. W niniejszej pracy rozważa się zbiór operacji niepodzielnych, w którym określono relację częściowego porządku oraz zbiór zasobów odnawialnych, podzielnych w sposób dyskretny i jeden zasób podzielny w sposób ciągły. Chwilowa szybkość wykonywania operacji jest ciągłą, niemalejącą funkcją ilości zasobu ciągłego przydzielonego do tej operacji w danej chwili. Przedstawiono ogólne własności i sposoby wyznaczania rozwiązań optymalnych ze względu na czas realizacji całego zbioru operacji, oraz trzy algorytmy przybliżone wykorzystujące ideę lokalnego przeszukiwania, których efektywność porównano na podstawie wyników eksperymentu obliczeniowego.

OPTIMIZATION OF CONTINUOUS AND DISCRETE RESOURCE ALLOCATION IN A PROJECT SCHEDULING PROBLEM WITH NONPREEMPTABLE ACTIVITIES

Summary. We consider a project consisting of precedence and resource-constrained activities which require renewable resources of two types: discrete and continuous ones. The processing rate of an activity is a continuous, nondecreasing and concave function of the amount of a continuous resource allotted to this activity at a time. The problem is to find an assignment of discrete resources and, simultaneously, a continuous resource allocation which minimize the project duration. We give basic properties and methods of constructing optimal schedules and present local search metaheuristics adopted for the considered problem. Finally, we provide results of computational experiments.

1. Wstęp

W niniejszej pracy rozważamy system typu kompleks operacji, zawierający zbiór operacji niepodzielnych oraz zbiór zasobów złożony z m zasobów odnawialnych, podzielnych w sposób dyskretny i jednego zasobu podzielnego w sposób ciągły. W zbiorze operacji jest określona relacja częściowego porządku, określająca ograniczenia kolejnościowe. Model operacji ma postać ciągłej, niemalejącej funkcji szybkość wykonania – ilość zasobu ciągłego. Operacja jest scharakteryzowana przez jej model, rozmiar (określający zapotrzebowanie na

przetwarzanie) oraz wektor żądań zasobowych odnośnie do zasobów dyskretnych oraz przedział dopuszczalnych przydziałów zasobu ciągłego. Przedstawiono ogólne własności i sposoby wyznaczania rozwiązań optymalnych ze względu na czas realizacji całego zbioru operacji dla wklęsłych i wypukłych modeli operacji. Znalezienie optymalnego rozdziału zasobu ciągłego dla modeli wklęsłych wymaga rozwiązania nieliniowego problemu programowania matematycznego dla każdego dopuszczalnego ciągu operacji. Ponieważ liczba tych ciągów rośnie wykładniczo ze wzrostem liczby operacji, procedura ta jest wysoce nieefektywna obliczeniowo. Przedstawiono zatem algorytmy przybliżone, wykorzystujące różne koncepcje lokalnego przeszukiwania (przeszukiwanie tabu, symulowane wyżarzanie i algorytm genetyczny), których efektywność porównano na podstawie eksperymentu obliczeniowego.

Problemy tego typu dla przypadku zadań niezależnych z jednym zasobem dyskretnym (zbiór identycznych maszyn równoległych) były rozważane w pracach [7, 9, 10], a dla zadań zależnych w pracy [8]. W tej pracy uogólnimy te rozważania na przypadek m zasobów odnawialnych, podzielnych w sposób dyskretny.

2. Sformułowanie problemu

Rozważmy kompleks n operacji w postaci dowolnego grafu acyklicznego w przedstawieniu wierzchołkowym. Operacje są niepodzielne i żądają do swego wykonania dwóch kategorii zasobów odnawialnych i ograniczonych: podzielnych w sposób dyskretny i podzielnych w sposób ciągły. Zakładamy, że dostępnych jest m rodzajów zasobów dyskretnych i jeden zasób ciągły. Liczba jednostek zasobu j, $j = 1, 2,\ldots, m$, niezbędna do wykonania operacji i, $i = 1, 2,\ldots n$, jest stała i dana wektorem $\mathbf{r}_i = [r_{i1}, r_{i2}, \ldots r_{im}]$. Liczba dostępnych jednostek zasobu j, $j = 1, 2,\ldots, m$, wynosi R_j. Chwilowa szybkość wykonywania operacji i zależy od ilości zasobu ciągłego $u_i(t) \geq 0$ przydzielonego do tej operacji w chwili t. Zależność ta jest opisana następującym równaniem:

$$\dot{x}(t) = \frac{dx_i(t)}{dt} = f_i[u_i(t)] \quad x_i(0) = 0, \quad x_i(C_i) = \bar{x}_i \tag{1}$$

gdzie: $x_i(t)$ jest stanem operacji i w chwili t,

$u_i(t)$ jest ilością zasobu ciągłego przydzielonego do operacji i w chwili t,

f_i jest ciągłą, niemalejącą funkcją, $f_i(0) = 0$,

C_i jest (nieznaną a priori) chwilą zakończenia operacji i,

$\bar{x}_i$ jest stanem końcowym (rozmiarem) operacji i.

Bez utraty ogólności założymy, że $\sum_{i=1}^{n} u_i(t) = 1$.

Problem polega na znalezieniu przydziału zasobów dyskretnych i zasobu ciągłego do operacji minimalizującego czas wykonania całego zbioru operacji $M = \max_i \{C_i$.

3. Znajdowanie rozwiązań optymalnych i suboptymalnych

Podzielmy dowolne dopuszczalne uszeregowanie (czyli rozwiązanie przedstawionego problemu) na $p \le n$ przedziałów wyznaczonych przez chwile zakończenia kolejnych operacji. Niech Z_k oznacza kombinację operacji wykonywanych w k-tym przedziale tego uszeregowania. *Ciągiem dopuszczalnym* S nazwiemy ciąg kombinacji Z_k, $k = 1, 2, \ldots, p$, odpowiadający danemu uszeregowaniu dopuszczalnemu. Aby ciąg kombinacji był dopuszczalny, muszą być spełnione następujące warunki:

1. liczba jednostek zasobu dyskretnego j, $j = 1, 2, \ldots, m$, przydzielonych do operacji w kombinacji Z_k, $k = 1, 2, \ldots, p$, nie przekracza R_j, to znaczy $\sum_{i \in Z_k} r_{ij} \le R_j$, $j = 1, \ldots, m$, $k = 1, \ldots, p$,
2. każda operacja występuje co najmniej w jednej kombinacji,
3. ograniczenia kolejnościowe między operacjami są spełnione,
4. niepodzielność operacji jest zachowana.

Ostatni warunek oznacza, że każda operacja występuje dokładnie w jednej lub tylko w kolejnych kombinacjach w S.

Odpowiednio rozmiar i-tej operacji $\tilde{x}_i$, $i = 1, \ldots, n$, może być podzielony na części $\tilde{x}_{ik} \ge 0$ $i = 1, \ldots, n$, $k = 1, \ldots, p$, wykonywane w k-tym przedziale uszeregowania.

Można wykazać [10], że w przypadku wypukłych funkcji f_i, $i = 1, \ldots, n$, uszeregowanie o minimalnej długości można otrzymać przez uszeregowanie operacji w kolejności zgodnej z grafem ograniczeń kolejnościowych, przydzielając każdej operacji całą dostępną ilość zasobu ciągłego. W takim uszeregowaniu w każdej chwili jest wykonywana dokładnie jedna operacja, co gwarantuje jego dopuszczalność również ze względu na ograniczenia związane z zasobami dyskretnymi. Natomiast w przypadku, gdy funkcje f_i, $i = 1, \ldots, n$, są wklęsłe, optymalne uszeregowanie otrzymuje się przez wykonywanie możliwie jak największej liczby operacji równoległe [10]. W tym przypadku dla danego ciągu dopuszczalnego S można znaleźć optymalny podział $\tilde{x}_{ik}$, $i = 1, \ldots, n$, $k = 1, \ldots, p$, rozmiarów operacji między

poszczególne przedziały uszeregowania rozwiązując następujący problem wypukłego programowania matematycznego:

Problem P. Zminimalizować $$M^*\left(\{\tilde{x}_k\}_{k=1}^{p}\right) = \sum_{k=1}^{p} M_k^*(\tilde{x}_k) \tag{2}$$

przy ograniczeniach $$\sum_{k \in K_i} \tilde{x}_{ik} = \tilde{x}_i, \qquad i = 1,2,\ldots,n \tag{3}$$

$$\tilde{x}_{ik} \geq 0, \qquad i = 1,2,\ldots,n; \quad k \in K_i \tag{4}$$

gdzie $M_k^*(\tilde{x}_k)$ jest jedynym dodatnim pierwiastkiem równania:

$$\sum_{i \in Z_k} f_i^{-1}\left(\tilde{x}_i / M^*\right) = 1$$

które można rozwiązać analitycznie dla niektórych funkcji f_i, ważnych z punktu widzenia zastosowań praktycznych [16].

Zatem uszeregowanie optymalne dla danej instancji można wyznaczyć znajdując optymalne rozwiązanie Problemu P dla każdego ciągu dopuszczalnego dla tej instancji i wybierając spośród tych rozwiązań to, które ma najmniejszą długość. Postępowanie takie jest jednak nieefektywne obliczeniowo, pomijając nawet złożoność Problemu P, gdyż liczba ciągów dopuszczalnych rośnie wykładniczo ze wzrostem liczby operacji. Uzasadnione jest zatem poszukiwanie dobrych algorytmów przybliżonych. W dalszej części pracy zaproponowano adaptacje algorytmów przeszukiwania tabu, symulowanego wyżarzania i algorytmu genetycznego do rozwiązywania tej klasy problemów. Wykazano [10], że aby znaleźć rozwiązanie optymalne, wystarczy rozważać ciągi dopuszczalne o długości n. Zatem ciąg dopuszczalny o długości n nazwano rozwiązaniem dopuszczalnym, a zbiór wszystkich rozwiązań dopuszczalnych stanowi przestrzeń przeszukiwaną przez rozważane algorytmy. Również bez utraty ogólności można przyjąć, że operacje są tak ponumerowane, że jeżeli operacja i poprzedza operację j, to $i \leq j$.

4. Algorytm symulowanego wyżarzania

Algorytm symulowanego wyżarzania należy do algorytmów lokalnego przeszukiwania [2]. Algorytm ten został po raz pierwszy zastosowany do rozwiązywania problemów kombinatorycznych w latach osiemdziesiątych [3, 11], co rozpoczęło jego dalsze liczne zastosowania w tej dziedzinie [1, 12]. Przystosowanie algorytmu do rozwiązywania danej klasy problemów wymaga ustalenia wartości parametrów charakteryzujących tzw. schemat

chłodzenia, reprezentacji rozwiązania, mechanizmu generowania rozwiązania początkowego i sąsiedniego oraz warunku zakończenia obliczeń. W omawianej implementacji przyjęto schemat chłodzenia podany w pracy [2]. Warunkiem zakończenia obliczeń jest odwiedzenie ustalonej a priori liczby rozwiązań. Rozwiązanie dopuszczalne jest reprezentowane w postaci dwóch ciągów [10], z których pierwszy określa kolejność rozpoczynania operacji (zgodną z grafem ograniczeń kolejnościowych), a drugi kolejność kończenia operacji. Informacja zawarta w drugim ciągu jest wykorzystywana podczas tworzenia rozwiązania dopuszczalnego tylko wtedy, gdy ze względu na ograniczenia zasobowe więcej niż jedna operacja musi zostać zakończona w jednej kombinacji. Mechanizm generowania rozwiązań sąsiednich gwarantuje dopuszczalność wszystkich odwiedzanych rozwiązań.

5. Algorytm przeszukiwania tabu

Algorytm przeszukiwania tabu został wprowadzony przez Glovera [4, 5], a przegląd najnowszych wyników można znaleźć w [6]. Podobnie jak w przypadku symulowanego wyżarzania zastosowanie tego algorytmu wymaga określenia reprezentacji rozwiązania dopuszczalnego, mechanizmu generowania rozwiązania początkowego oraz rozwiązań sąsiednich, metody sterowania listą tabu i warunku zakończenia obliczeń. W omawianej implementacji rozwiązanie dopuszczalne jest reprezentowane w postaci n-elementowego ciągu dopuszczalnego, tj. ciągu kombinacji operacji. Zdefiniowano dwa mechanizmy generowania rozwiązań sąsiednich: *zastąpienie* i *zamianę* [10]. Zastąpienie polega na zastąpieniu operacji inną operacją, pod warunkiem że takie zastąpienie nie narusza ograniczeń kolejnościowych ani zasobowych. Zamiana pozwala uniknąć „zablokowania" procesu przeszukiwania, gdy ograniczenia zasobowe nie pozwalają na wykonanie zastąpienia. Polega ona na zamianie dwóch operacji występujących w jednoelementowych kombinacjach w ciągu dopuszczalnym. Listą tabu steruje się zgodnie z metodą REM [5]. Warunkiem zakończenia obliczeń jest odwiedzenie określonej liczby rozwiązań.

6. Algorytm genetyczny

Przegląd zagadnień związanych ogólnie z problematyką algorytmów genetycznych przedstawiono w pracy [14]. W przypadku algorytmu genetycznego należy określić sposób reprezentacji rozwiązania dopuszczalnego oraz generowania populacji początkowej i tzw. operatorów genetycznych prowadzących do kolejnych generacji osobników oraz warunek

zakończenia obliczeń. W naszej implementacji rozwiązanie dopuszczalne jest reprezentowane w postaci dwóch ciągów (chromosomów). Pierwszy chromosom określa kolejność rozpoczynania operacji. Drugi jest wektorem n-1-elementowym, którego współrzędne przyjmują wartości ze zbioru: $\{-n, -n+1, \ldots, -2, -1, 0, 1, 2, \ldots, n-1, n\}$ [10]. Zdefiniowano trzy operatory genetyczne: krzyżowanie, mutację i zamianę (będącą pewną odmianą mutacji). Krzyżowanie polega na wylosowaniu dwóch pozycji w drugim chromosomie rodziców i zamianie elementów między wylosowanymi pozycjami. Mutacja polega na zastąpieniu losowo wybranego dodatniego elementu drugiego chromosomu losowo wybraną wartością ze zbioru $\{1, 2, \ldots, n-1, n\}$. Zamiana polega na zamianie pozycji dwóch sąsiednich elementów w pierwszym chromosomie, jeżeli nie powoduje to naruszenia ograniczeń kolejnościowych. Warunkiem zakończenia obliczeń jest odwiedzenie pewnej liczby rozwiązań.

7. Eksperyment obliczeniowy

Opracowane algorytmy porównano na drodze eksperymentu obliczeniowego przeprowadzonego dla losowo generowanych instancji.

Wszystkie algorytmy zaimplementowano w języku C++ i wykonywano na komputerze SGI Power Challenge XL z 12 procesorami RISC R8000. Do rozwiązania problemu programowania matematycznego optymalizującego przydział zasobu ciągłego zastosowano specjalnie przystosowany solver CFSQP (A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints) 2.3 [13]. Przyjęto, że kryterium stopu dla solvera jest spełnione, gdy bezwzględna różnica wartości funkcji celu w kolejnych iteracjach jest mniejsza lub równa 10^{-3}.

Porównując trzy algorytmy (przeszukiwania tabu TS, symulowanego wyżarzania SA i algorytm genetyczny GA) ustalono wspólne kryterium stopu, zapewniające (w przybliżeniu) równe nakłady obliczeniowe dla każdego algorytmu. Kryterium tym była liczba odwiedzonych rozwiązań. Ustalono, że algorytm kończy przeszukiwanie po odwiedzeniu 1000 rozwiązań. W przypadku przeszukiwania tabu i algorytmu genetycznego zatrzymanie programu dokładnie po odwiedzeniu 1000 rozwiązań nie jest możliwe, gdyż odwiedzają one w każdej iteracji pewien zbiór rozwiązań. Dla tych algorytmów ustalono, że zakończą one działanie zaraz po iteracji, w której odwiedzono tysięczne rozwiązanie.

Tablica 1

Wyniki eksperymentu obliczeniowego

R	*Algorytm*	#	*Średnie odchylenie [%]*	*Maksymalne odchylenie[%]*
2	TS	13	1.3463	12.4451
	GA	23	0.2513	2.9953
	SA	30	0	0
5	TS	10	1.4525	12.5193
	GA	16	0.6318	4.9142
	SA	30	0	0
10	TS	10	1.5564	12.4451
	GA	23	0.1406	2.0287
	SA	30	0	0
15	TS	10	1.7217	11.4951
	GA	20	0.4741	3.7331
	SA	28	0.0100	0.1644

Eksperyment przeprowadzono dla instancji zawierających po 10 operacji z jednym dyskretnym i jednym ciągłym zasobem ograniczonym. Wygenerowano losowo trzy zestawy po 30 instancji i $R = 2, 5, 10$ i 15 odpowiednio w kolejnych zestawach. Modele operacji miały postać $f_i = u_i^{1/\alpha_i}$. Wartości α_i generowano losowo ze zbioru $\{1, 2\}$ z jednakowym prawdopodobieństwem. Rozmiary operacji generowano z przedziału [1, 100] z rozkładem równomiernym. Średnia gęstość grafu ograniczeń kolejnościowych wynosiła 0,5.

Porównanie rozwiązań otrzymanych przez poszczególne algorytmy przedstawiono w tablicy 1. Podano trzy wartości: średnie względne odchylenie od najlepszego znalezionego rozwiązania, maksymalne względne odchylenie od najlepszego znalezionego rozwiązania, oraz liczba instancji, dla których algorytm znalazł rozwiązanie najlepsze spośród badanych algorytmów.

Jak wynika z tablicy 1, najlepsze wyniki uzyskano stosując algorytm symulowanego wyżarzania. Odchylenie długości uszeregowania dla rozwiązań znajdowanych za pomocą algorytmu genetycznego od długości uszeregowań znajdowanych za pomocą algorytmu symulowanego wyżarzania nie przekracza średnio 0.7%, a maksymalne odchylenie nie przekracza 5%. W przypadku algorytmu przeszukiwania tabu odchylenia te są znacznie większe i wynoszą odpowiednio ok. 1.5 % i 12%. Wyniki te są powtarzalne, niezależnie od rozmiaru instancji.

8. Podsumowanie

W pracy rozpatrzono problem znajdowania uszeregowań optymalnych ze względu na czas wykonania zbioru operacji w kompleksie operacji z dwoma kategoriami zasobów odnawialnych: podzielnych w sposób dyskretny i w sposób ciągły. Podano metodykę znajdowania rozwiązań optymalnych (w ogólności nieefektywną obliczeniowo) oraz zaproponowano trzy algorytmy lokalnego przeszukiwania: przeszukiwania tabu, symulowanego wyżarzania i algorytm genetyczny.

Podano wyniki eksperymentu obliczeniowego zrealizowanego w celu porównania efektywności tych trzech algorytmów odpowiednio dostosowanych do powyższych problemów. Eksperyment wykazał, że po odwiedzeniu tej samej liczby rozwiązań najlepsze rozwiązania znajduje algorytm symulowanego wyżarzania. W ramach dalszych prac przewiduje się analizę szczególnych przypadków w celu zdefiniowania warunków w jakich możliwe jest rozwiązanie problemu w czasie wielomianowym. Ponadto przewiduje się modyfikację algorytmów przybliżonych zmierzającą do zwiększenia ich efektywności, zwłaszcza na etapie oceny rozwiązań dopuszczalnych.

LITERATURA

1. Aarts E.H.L., Korst J.H.M.: Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing, Wiley, Chichester, 1989.
2. Aarts E.H.L., Korst J.H.M., van Laarhoven P.J.M.:Simulated Annealing, w: Aarts E.H.L. and Lenstra J.K, ed., Local Search in Combinatorial Optimization, Wiley, Chichester, 1997, 91-120.
3. Černy, V.: Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. Journal of Optimization Theory And Applications 45, 1985 41-51.
4. Glover F., Tabu Search - part 1, ORSA Journal on Computing, 1, 1989, 190-206.
5. Glover F., Tabu Search - part 2, ORSA Journal on Computing, 2, 1990, 4-32.
6. Glover F., Laguna M., Tabu Search, w: Reeves C.R., ed. Modern Heuristic Techniques for Combinatorial Problems. Blackwell Scientific Publications, Oxford, 1993.
7. Józefowska J., Mika M., Różycki R., Waligóra G, Węglarz J., Local search metaheuristics for discrete-continuous scheduling problems, European Journal of Operational Research, 1998 - to appear.
8. Józefowska J., Mika M., Różycki R., Waligóra G, Węglarz J., Scheduling precedence constrained nonpreemptable jobs on parallel machines with additional, discrete-continuous resources, Proc. Int. Conf. MMAR'98, 1998 w druku.
9. Józefowska J., Mika M., Różycki R., Waligóra G, Węglarz J., Project scheduling under discrete and continuous resources, w: J. Węglarz (ed.), Recent Advances in Project

Scheduling, Kluwer, 1998 (w druku).

10. Józefowska J., and Węglarz J., On a methodology for discrete-continuous scheduling, European Journal of Operational Research, 1998 - to appear.
11. Kirkpatrick S., Gelatt Jr. C.D., Vecchi M.P.: Optimization by simulated annealing. Science 220, 1983, 671-680.
12. Laarhoven van P.J.M., Aarts E.H.L.: Simulated Annealing: Theory and Applications, Reidel, Dordrecht, 1987.
13. Lawrence C., Zhou J.L., Tits A.L.: Users guide for CFSQP Version 2.3 (Released August 1995), available by e-mail: andre@eng.umd.edu.
14. Metropolis N., Rosenbluth A., Rosenbluth M., Teller A., Teller E.: Equation of state calculations by fast computing machines. Journal of Chemical Physics 21, 1953, 1087-1092.
15. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, Springer, Berlin, 1992 .
16. Węglarz J. Modelling and control of dynamic resource allocation project scheduling scheduling systems, w: S. G. Tzafestas (ed.), Optimization and Control of Dynamic Operational Research Models, North-Holland, Amsterdam, 1982.

Recenzent: Prof.dr hab.inż.Ryszard Gessing

Abstract

We consider a project consisting of precedence and resource-constrained activities, which require renewable resources of two types: discrete and continuous ones. Discrete resources can be allotted to activities in amounts (numbers of units) from given finite sets only. Continuous resources can be allotted to activities in amounts (unknown in advance) from given intervals. The processing rate of an activity is a continuous, nondecreasing and concave function of the amount of a continuous resource allotted to this activity at a time.

The problem is to find an assignment of discrete resources and, simultaneously, a continuous resource allocation which minimize the project duration. A schedule consists of two components: a feasible sequence and a vector function of the continuous resource allocation. A feasible sequence is a sequence of combinations of activities which fulfils the following conditions:

(i) discrete resource requirements of all activities in any combination can be simultaneously fulfilled,

(ii) each activity appears in at least one combination,

(iii) precedence constrains are satisfied,

(iv) nonpreemptibility of each activity is guaranteed.

For such a feasible sequence an optimal continuous resource allocation can be found by solving a nonlinear mathematical programming problem. However, finding an optimal feasible sequence is a NP-hard combinatorial problem. Thus, we adopted local search metaheuristics (simulated annealing, tabu search and genetic algorithm) in order to find good suboptimal solutions. We provide results of preliminary computational experiments.