ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1988
Seria:AUTOMATYKA z.94 Nr kol.970

Stanisław Berka*
Eugeniusz Toczyłowski
Politechnika Warszawska

IMPLEMENTACJA METODY PODZIAŁU I OGRANICZEŃ DLA PEWNEGO UOGÓLNIENIA ZADANIA PLECAKA

Streszczenie. W pracy przedstawiono implementację algorytmu rozwiązującego zadanie załadunku ze stałą dopłatą w funkcji celu i funkcji w ograniczeniu. Ponieważ zadanie jest NP-trudne, algorytm jest realizacją metody podziału i ograniczeń wykorzystującą szczególne właściwości tego zadania. W szczególności bardzo efektywnie wyznaczane jest oszacowanie od dołu. Opisany został efektywny sposób implementacji algorytmu oraz wyniki jego testowania. Algorytm umożliwia dokładne rozwiązywanie zadań na IBM PC z liczbą zmiennych rzędu 40 w czasie rzędu 100 s. Dla liczby zmiennych rzędu 200 rozwiązanie przybliżone o dokładności 8% otrzymywane jest w czasie rzędu 4 s, a z dokładnością 0,08% w czasie rzędu 80 s.

1. Wprowadzenie

Zadanie plecakowe ze stałą dopłatą zarówno w funkcji celu, jak i funkcji w ograniczeniu, zwane dalej uogólnionym zadaniem plecaka, powstaje jako zadanie lokalne przy dekompozycji (uzyskanej poprzez relaksację jednej z grupy ograniczeń) zadania rozdziału zadań w gnieździe produkcyjnym z maszynami równoległymi przy obecności kosztów i czasów wznowień produkcji (opisane w [4], patrz także [3]). Zadanie zostało sformułowane w drugim paragrafie artykułu. Jest to zadanie NP-trudne, gdyż binarne zadanie plecaka jest szczególnym przypadkiem uogólnionego zadania plecaka (patrz [1]). W trzecim paragrafie została w ogólnym zarysie przedstawiona metoda podziału i ograniczeń dla tego zadania. Szczegółowo algorytm jest opisany w [3]. Czwarty paragraf opisuje modyfikacje algorytmu, które są potrzebne, aby można było rozwiązywać zadanie z ograniczeniem równościowym. Główne elementy efektywnej implementacji przedstawiono w piątym paragrafie, a w szóstym - wyniki jego testowania.

2. Sformułowanie zadania

Uogólnionym zadaniem plecakowym będziemy nazywać następujące zadanie programowania liniowego mieszanego:

$$\min_{(x,v)\in X} \sum_{i=1}^{n} (s_i v_i - c_i x_i) \qquad (1a)$$

* obecnie IBS PAN

przy ograniczeniu

$$\sum_{i=1}^{n}(e_i v_i + p_i x_i) \leq Q \qquad (1b)$$

gdzie $x,v \in R^n$, a X określony jest następująco:

$$X = \left\{ (x,v) \mid v_i \in \{0,1\} \;\wedge\; 0 \leq x_i \leq M_i v_i \quad i=1,\ldots,n \right\} \qquad (1c)$$

a s_i, c_i, e_i, p_i, M_i, Q są danymi liczbami nieujemnymi.

Zadanie można interpretować jako wybór spośród zamówień produktów do produkcji. Wtedy s_i i e_i oznaczają odpowiednio koszt i czas przezbrojenia do produkcji wyrobu i-tego typu; c_i i p_i oznaczają odpowiednio zysk i czas przy produkcji pewnej jednostki wyrobu i-tego typu; M_i jest wielkością zamówienia na wyrób i-tego typu, a Q - maksymalnym czasem pracy linii produkcyjnej.

3. Algorytm podziału i ograniczeń

W rozdziale tym przedstawiono krótki opis algorytmu, podanego w [3]. Tam też znajduje się dowód poprawności algorytmu. Algorytm jest prostą metodą podziału i ograniczeń działającą na drzewie binarnym i znajdującą rozwiązanie ε-dokładne z zadaną dokładnością ε. Drzewo przeszukiwane jest wgłąb. W każdym wierzchołku znajdowane jest oszacowanie od dołu wartości funkcji celu stanowiące górne oszacowanie problemu. Gałąź wychodząca z danego wierzchołka jest zamykana, gdy błąd względny aktualnie najlepszego znalezionego rozwiązania liczony względem dolnego oszacowanie jest nie większy niż zadane ε.

Znalezienie dolnego oszacowania w każdym wierzchołku drzewa polega na rozwiązaniu zadania relaksacji Lagrange'a powstałego po relaksacji ograniczenia (1b). Zadanie to rozwiązywane jest szczególnie efektywnie dzięki wykorzystaniu specjalnych właściwości zadania (1). Można pokazać [3], że rozwiązanie zadania relaksacji Lagrange'a sprowadza się do "załadowania plecaka" produktami w kolejności uszeregowanych nierosnąco współczynników ψ_i określonych wzorem

$$\psi_i = \frac{-s_i + c_i M_i}{e_i + p_i M_i} \qquad (2)$$

aż do wypełnienia (ewentualnie przepełnienia) plecaka dla pewnego $i=i^*$.

Powyższa metoda znajdowania oszacowania dolnego jest bardzo efektywna; wymaga tylko O(n) operacji. Dla porównania zauważmy, że relaksacja liniowa zadania (1) jest zadaniem programowania liniowego o specjalnej strukturze z (n+1) ograniczeniami i jej obliczenie jest kosztowniejsze.

Heurystyka znajdująca rozwiązanie dopuszczalne jest oparta na algorytmie zachłannym. Produkty uporządkowane tak, że: $\psi_1 \geq \psi_2 \geq \ldots \geq \psi_n$, ładowane są podczas rozwiązywania zadania relaksacji Lagrange'a do plecaka aż do

przepełnienia go. Następnie włożone produkty są przeglądane i wyrzucany z plecaka jest ten produkt, który ma najmniejszy udział w wartości funkcji celu. O ile plecak jest nadal przepełniony, powtarza się operację. Jeśli plecak wskutek wyrzucenia z niego produktu jest niepełny, to sprawdzany jest zbiór nie włożonych produktów, aby ewentualnie wybrać najlepszy do włożenia, nie przepełniając jednak plecaka.

Podział problemu na podproblemy jest dokonywany przez ustalenie zmiennej v_i pewnego produktu na 0 lub 1. Produktem tym jest $i=i^*$, który przepełni plecak przy rozwiązywaniu zadania relaksacji Lagrange'a. W [3] pokazano, że dzięki wyborowi właśnie tego produktu do podziału, dolne oszacowania w obu podproblemach mogą wzrosnąć, podczas gdy wybór jakiegokolwiek innego wyklucza wzrost oszacowań w obu podproblemach.

4. Modyfikacja algorytmu dla wypadku ograniczenia równościowego

Przedstawiony w rozdziale trzecim algorytm może po niewielkich modyfikacjach służyć do rozwiązywania zadania (1) ze znakiem '=' w ograniczeniu (1b). Przedstawimy obecnie konieczne modyfikacje. O ile w zadaniu z ograniczeniem nierównościowym można z rozwiązania z góry odrzucić produkty spełniające warunek, że $s_i - c_i M_i \geq 0$ ($\hat{x}_i = 0$ dla tych produktów), to dla wersji równościowej algorytmu produktów tych nie można odrzucić.

W wersji równościowej nie w każdym wierzchołku drzewa podziału i ograniczeń daje się znaleźć rozwiązanie dopuszczalne za pomocą zastosowanego algorytmu. Metoda podziału i ograniczeń działa jednak poprawnie - do momentu znalezienia pierwszego rozwiązania dopuszczalnego (wartość górnego oszacowania wynosi ∞).

Rozwiązanie optymalne zadania w wersji nierównościowej spełnia warunek:

$$m_i \leq x_i \leq M_i v_i \quad \text{albo} \quad x_i = 0 \quad \text{dla} \quad i=1,\ldots,n, \tag{3}$$

gdzie $m_i = \frac{s_i}{c_i}$, ponieważ dla $0 \leq x_i \leq m_i$ i-ty składnik funkcji celu 1a jest dodatni, a dla $x_i = 0$ jest równy zeru.

Dla wersji równościowej taki warunek nie jest prawdziwy (x_i może przyjmować wartości z $[0; M_i]$), jest to uwzględniane podczas wyboru produktu do wyrzucenia (wrzucenia) w algorytmie poszukiwania rozwiązania dopuszczalnego.

Przy obliczaniu dolnego oszacowania przeglądane są w wersji równościowej zarówno dodatnie, jak i ujemne wartości mnożnika ψ_x (ponieważ rozwiązanie zadania dualnego powstałego przez relaksację ograniczenia równościowego wymaga znalezienia $\max L_D(\psi)$ dla $\psi \in R$).

5. Implementacja algorytmu

Struktura danych drzewa podziału i ograniczeń zorganizowana jest w następujący sposób. Każdy wierzchołek drzewa jest rekordem, który zawiera: wskaźniki na wierzchołki synów, jak też na ojca oraz wszelkie informacje, które wraz z informacjami zawartymi w opisanej poniżej liście produktów, wystarczają zarówno do rozwiązania problemu i/lub zamknięcia wierzchołka, jak też do przywrócenia liście produktów stanu pierwotnego, co jest potrzebne do powrotu i rozwiązania pominiętych przy schodzeniu w głąb podproblemów.

W każdej chwili działania programu w pamięci istnieją jedynie rekordy odpowiadające wierzchołkom tworzącym ścieżkę od korzenia do sondowanego wierzchołka. Ilość pamięci potrzebna do realizacji drzewa jest $O(n)$, gdyż głębokość drzewa nigdy nie przekracza n ze względu na sposób podziału na podproblemy przyjęty w algorytmie.

Do wszystkich operacji na produktach w wykonanej implementacji służy jedna dwukierunkowa lista produktów (na tych samych rekordach tworzących listę tworzone są okresowo dwie rozłączne listy produktów włożonych i niewłożonych do plecaka - wystarcza na to 4n dodatkowych komórek pamięci). Każdy element tej listy odpowiada jednemu produktowi i zawiera wszystkie dane dotyczące tego produktu $(s_i, c_i, e_i, p_i, M_i, \psi_i)$. Lista jest na początku uporządkowana nierosnąco według ψ_i i ten porządek jest utrzymany przez cały czas. W każdym wierzchołku modyfikowane jest ψ_i tylko dla jednego i (w związku z ustaleniem v_i na 1 lub na 0), więc wystarczy przestawić ten jeden element, aby przywrócić porządek listy. Można to zrealizować następująco.

1° Uporządkuj listę zgodnie z niemalejącym porządkiem (2).

2° Jeśli rozwiązujesz podproblem z $v_i=0$, to:

Podstaw $\psi_i = 0$ ($-\infty$ dla wersji=) oraz

Przestaw element na koniec listy, aby nie był przeglądany.

W przeciwnym razie:

Podstaw $\psi_i = \bar{\psi}_i$, gdzie $\bar{\psi}_i = \frac{c_i}{p_i}$ oraz wstaw element w takie miejsce listy, aby był zachowany porządek listy.

3° Przy powrocie wykonaj: przywróć liście stan pierwotny.

Wszystkie operacje wykorzystujące algorytm zachłanny (algorytm obliczania dolnego oszacowania i algorytm znajdujący rozwiązanie dopuszczalne) przetwarzają kolejne elementy listy. Informacje niezbędne do przywrócenia poprzedniego stanu listy zawarte są w rekordzie wierzchołka.

Przy obliczaniu oszacowań dla podproblemów tworzona jest czasowo kopia i-tego produktu. Wtedy każdy element listy odpowiada punktowi wierzchołkowemu funkcji Lagrange'a dla podproblemu. Przeglądając listę i obliczając nachylenie funkcji $L_D(\psi)$ znajdujemy w co najwyżej n+1 krokach $\hat{\psi}$ oraz

$L_D(\hat{Y})$, czyli dolne oszacowanie zadania.

Ogółem dla pakietu potrzebne jest ok. 80n słów pamięci. Ilość operacji związanych z sondowaniem jednego wierzchołka jest O(n+1). Algorytm został zaimplementowany w TURBO PASCALu na IBM PC.

6. Testowanie algorytmu

Dane dla zadań testowych były generowane losowo (rozkład jednostajny) z zadanych przedziałów. Testowanie było przeprowadzane na zadaniach o liczbie produktów 10, 40, 100 dla dwóch klas podobieństwa produktów.

1. Parametry produktów losowane z przedziału [10;100] - przykłady oznaczone rzymską cyfrą I.
2. Parametry produktów losowane z przedziału [90;100] - przykłady oznaczone rzymską cyfrą II.

Porównywane były wyniki dla równościowej i nierównościowej wersji zadania. Badany był nakład obliczeń potrzebny do znalezienia ε-dokładnego rozwiązania dla różnych wartości ε.

Nakład obliczeń oceniany był poprzez ilość wierzchołków, które trzeba było przebadać, aby znaleźć rozwiązanie. Wyniki testowania zostały przedstawione w tabelach 1, 2, 3.

Liczby w kolumnach oznaczonych przez min, śr, max określają minimalną, średnią i maksymalną ilość wierzchołków dla serii zadań różniących się losowo wygenerowaną wartością prawej strony ograniczenia Q z przedziału

$$\left[\min_i \ (e_i + p_i M_i) \ ; \ \sum_{i=1}^{n} (e_i + p_i M_i) \right]$$

W kolumnie 1 przedstawiono liczby wierzchołków potrzebnych do rozwiązania całego zadania; w kolumnie 2 - aby znaleźć rozwiązanie optymalne. W kolumnie 3 podano ilość wysondowanych wierzchołków potrzebnych do udowodnienia optymalności znalezionego rozwiązania.

Tabela 1.

Wyniki dla serii 10-20 zadań	ε=0 ogranicz '='	Pełny przebieg algorytmu			Znalezienie rozw.opt.			Dowód optymalności rozw.		
	Przykład	min	śr	max	min	śr	max	min	śr	max
Produkty o parametrach z [10,100]	(I) n=10	0	1	3	0	0	2	0	0	1
Produkty o parametrach [90,100]	(II) n=10	0	133	441	0	50	389	0	80	301

Tabela 2.

=0 ogranicz.	Pełny przebieg algorytmu			Znalezienie rozw. opt.			Dowód optymalności rozw.		
Przykład	min	śr	max	min	śr	max	min	śr	max
(I) n=10	0	1	3	0	0	2	0	0	1
(I) n=40	0	5	29	0	4	26	0	1	3
(I) n=100	0	22	84	0	15	61	0	6	23

Tabela 3.

Ogranicz.	Pełny przebieg algorytmu			Znalezienie rozw. opt.			Dowód optymalności rozw.		
Przykład	min	śr	max	min	śr	max	min	śr	max
$\varepsilon = 0\%$	0	97	430	0	42	182	0	55	248
$\varepsilon = 3\%$	0	44	283	0	24	212	0	20	96
$\varepsilon = 7\%$	0	15	93	0	1	7	0	14	93

Tabela 1 zawiera porównanie nakładów obliczeń dla zadań o różnym stopniu podobieństwa. Porównując różnice w efektywności dla n=10, I oraz II widać, że warto dokonać agregacji produktów podobnych biorąc średnie wartości parametrów dla całej grupy zawierającej produkty podobne.

W Tabeli 2 zostały porównane wyniki dla zadań o różnych rozmiarach. Widać, że średnio liczba wierzchołków przeglądanych przez algorytm wzrasta szybciej niż n. Tabela 3 zawiera porównanie nakładów obliczeń związanych ze znalezieniem ε-dokładnego rozwiązania dla różnych wartości ε. Do wyników podanych w tej tabeli należy dodać komentarz, że przy zwiększaniu czas obliczeń spada szybciej dla zadań 'łatwych' (o parametrach istotnie różniących się między sobą) niż dla zadań 'trudnych'.

Zadanie z klasy zadań trudnych z 40 produktami (n=40) dla pewnej wartości Q nie zostało rozwiązane pomimo wysondowania 3.400 wierzchołków przy $\varepsilon = 0\%$. To samo zadanie przy $\varepsilon = 7\%$ zostało rozwiązane w 526 wierzchołkach. Zadanie z ograniczeniem '=' wymaga średnio o 80% większego czasu niż zadanie z ograniczeniem '$\leq$'.

Dla zadań z klasy trudnych, rozwiązanie dokładne zadań o $n \geq 40$ wydaje się niemożliwe tym algorytmem w akceptowalnym czasie. Jednakże średnio metoda zachowuje się bardzo dobrze, gdy stosujemy ją jako heurystykę i zadowalamy się rozwiązaniem suboptymalnym. Jeżeli przerwiemy obliczenia po wysondowaniu np. 3n wierzchołków, tzn 40 na 43 rozwiązanych zadań błąd rozwiązania był nie większy niż 1% wartości funkcji celu. Wyniki dotyczące takich rozważań zebrano w tabeli 4.

Tabela 4.

Przykład	II n=10	IB n=40	I n=100	Σ
Ilość zadań w serii	22	10	9	41
Ilość zadań z rozwiązaniem optym. znalezionym w wierzch. 0-wym	12	2	1	15
Ilość zadań z rozwiązaniem optm. znalezionym w n/3 wierzchołków	13	7	6	26
Ilość zadań, gdzie w 3n wierzchołkach znaleziono rozwiąz. o błędzie 1%	21	10	9	10
Ilość zadań o błędzie pierwszego rozwiąz. dopuszczalnego nie większym niż 1%	19	8	7	34

Zaobserwowano, że jeżeli produkty są podobne, to najtrudniejsze do rozwiązania są zadania, w których Q jest bliskie połowy

$$Q_{max} = \sum_{i=1}^{n} (e_i + p_i M_i)$$

Jeżeli chodzi o czas obliczeń, to na IBM PC/XT sondowanie jednego wierzchołka dla zadania z 40 zmiennymi trwało nie dłużej niż 3 s.

Ciekawa jest obserwacja, że zmiana Q może zmienić charakter rozwiązań oraz przebieg algorytmu.

7. Podsumowanie

W pracy przedstawiono algorytm rozwiązywania uogólnionego zadania plecaka, efektywną pod względem czasu i pamięci implementację oraz szczegółowe wyniki testowania. Wyniki obliczeń wskazują na to, że opisany algorytm może służyć do znajdowania ε- dokładnych rozwiązań nawet stosunkowo dużych zadań tego typu (rzędu kilkuset zmiennych) oraz do znajdowania rozwiązań dokładnych dla zadań z kilkudziesięcioma zmiennymi.

W opisanym algorytmie przypuszczalnie można usprawnić heurystykę znajdującą rozwiązanie dopuszczalne, przepełniając plecak o pewną ilość produktów, a następnie wyrzucając je, jak to opisano w paragrafie 3. Inna możliwość poprawy leży w wyborze innego produktu do podziału problemu na podproblemy (można zastosować heurystyki pseudokosztowe). Nieodzowne jest wyposażenie pakietu w mechanizm redukcji liczby zmiennych przed przystąpieniem do metody podziału i oszacowań.

Praca była częściowo finansowana w ramach programu badawczego R.P.I.02 w temacie 5.3.

LITERATURA

[1] S. Berka: Algorytmy rozwiązywania wybranych zadań harmonogramowania produkcji, Praca magisterska, Instytut Automatyki PW 1986.

[2] R. S. Garfinkel, G.L. Nemhauser: Programowanie całkowitoliczbowe. PWN, Warszawa 1978.

[3] E. Toczyłowski, S. Berka: Efektywna metoda podziału i ograniczeń dla uogólnionego zadania plecaka. W przygotowaniu.

[4] E. Toczyłowski: Two phase algorithm for lot size scheduling in single stage production systems with parallel facilities. Biulletin of Polish Academy of Sciences - Technical Sciences, Vol. 35, 1987, No 1-2,

[5] S. Walukiewicz: Programowanie dyskretne. PWN, Warszawa 1986.

Recenzent: Doc.dr hab.inż.M.Zaborowski

Wpłynęło do Redakcji do 1988-04-30.

ИМПЛЕМЕНТАЦИЯ МЕТОДА РАЗДЕЛА И ОГРАНИЧЕНИЙ ДЛЯ НЕКОТОРОЙ ОБОБЩЁННОЙ ЗАДАЧИ РЮКЗАКА

Р е з ю м е

В работе дана имплементация алгоритма решающего задачу загрузки с постоянным доплачиванием как функция цели и ограничений. Так как задача является P – трудной , алгоритм реализован по методу раздела и ограничений . Эффективным образом получена нижняя оценка . Алгоритм даёт возможность точного решения задачи на IBM – PC с числом переменных порядка 40 во время до 100 с. Для числа переменных порядка 200 приближённое решение с точностью 8% получается за примерно 4 с а с точностью 0,08 % – за 80 с.

IMPLEMENTATION OF THE BRANCH-AND-BOUND ALGORITHM FOR A GENERALIZED KNAPSACK PROBLEM

S u m m a r y

An algorithm for the generalized knapsack problem, in which the cost function and the constraints are in the fixed-change forms, is presented. The problem is NP-hard, therefore the algorithm is an implementation of the branch-and-bound method which takes into account specific properties of the problem. It is based on a very efficient calculation of the Lagrangean-relaxation lower bound.problems with 40 binary and 40 continuous variables are solved to optimality in 100 sec on IBM PC. Approximate solutions of problems with 200 binary and 200 continnous variables are computed with the eocurecy 0,08% in 80 sec.