ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2008
Seria: AUTOMATYKA z. 150 Nr kol. 1796

Wojciech BOŻEJKO
Politechnika Wrocławska

Maciej WALCZYŃSKI, Mieczysław WODECKI
Uniwersytet Wrocławski

PROBLEM DYSTRYBUCJI Z TERMINAMI DOSTAW

Streszczenie. W pracy przedstawiono algorytm oparty na metodzie przeszukiwania z tabu, rozwiązywania problemu dystrybucji z terminami dostaw. Jest on równoważny pewnemu jednomaszynowemu problemowi szeregowania, który w literaturze jest oznaczany przez $1|s_{ij}|\Sigma w_i T_i$ i należy do klasy problemów silnie NP-trudnych. Wykonano obliczenia na reprezentatywnej grupie danych, a otrzymane wyniki porównano z najlepszymi znanymi w literaturze.

DISTRIBUTION WITH DUE DATES PROBLEM

Summary. A tabu search algorithm is proposed in the paper to solve a distribution problem with due dates. It is equivalent to a single machine scheduling problem, which is described by $1|s_{ij}|\Sigma w_i T_i$ in the literature and it belongs to strongly NP-hard class. Calculations were done on representative group of test instances, obtained results were compared to the best known solutions from the literature.

1. Wprowadzenie

Jedną z dynamicznie rozwijających się dziedzin gospodarki jest przemysł spożywczy związany z produkcją wyrobów bez konserwantów o krótkim okresie przydatności do spożycia. Są to produkty niemagazynowane, które producent bezpośrednio dostarcza do odbiorcy. Stosowane są tam bardzo rygorystyczne normy i standardy dotyczące procesu produkcji, który często jest w pełni zdeterminowany przez technologię. Stąd obniżenie kosztów może nastąpić głównie przez optymalizację dystrybucji. Tym bardziej że czasami stanowią one nawet kilkadziesiąt procent ogólnych kosztów produktu. Prowadzi to do interesujących, ale i trudnych problemów optymalizacyjnych bazujących na istotnych uogólnieniach klasycznego problemu komiwojażera.

W pracy rozpatrujemy problem, w którym należy jednym kursem rozwieźć pewien produkt z magazynu do wielu odbiorców. Znane są czasy przejazdu pomiędzy magazynem a każdym z odbiorców oraz pomiędzy odbiorcami. Ze względu na specyfikę produktu (czas niezbędny na rozprowadzenie), należy go dostarczyć do

odbiorcy przed z góry ustalonym terminem dostawy. W przypadku jego przekroczenia naliczana jest kara za spóźnienie. Należy tak zaplanować trasę pojazdu, aby dostarczyć produkt każdemu z odbiorców, a suma ewentualnych kar była minimalna. W dalszej części przedstawimy algorytm rozwiązywania opisanego problemu oparty na metodzie przeszukiwania z tabu (*Tabu Search*, w skrócie *TS*). W jego konstrukcji zastosowano metody przeglądu pośredniego, eliminujące nawet do 50% elementów otoczenia generowanego przez klasyczne ruchy typu wstaw (*insert*). Eksperymenty obliczeniowe wykonano, korzystając z danych referencyjnych zamieszczonych na stronie [5], a otrzymane wyniki porównano z wynikami najlepszych obecnie znanych algorytmów.

2. Problem dystrybucji

Dany jest zbiór n odbiorców $S = \{1,2,...,n\}$ oraz spełniająca warunek trójkąta macierz *czasów przejazdów* pomiędzy odbiorcami $C = [c_{ij}]_{nxn}$. Dla ustalenia uwagi zakładamy, że pierwszym „odbiorcą" jest magazyn. Niech p_i ($i \in S$) będzie *czasem rozładunku*, d_i - *najpóźniejszym żądanym terminem* zakończenia rozładunku, a w_i - *współczynnikiem kary* za spóźnienie. Jeżeli dana jest pewna trasa (kolejność odbiorców) oraz C_i ($i = 2,3,...,n$) jest *terminem zakończenia* rozładunku, to $T_i = \max\{0, C_i - d_i\}$ nazywamy *spóźnieniem*, a w_iT_i *karą* za spóźnienie dostawy. W rozpatrywanym problemie dystrybucji należy wyznaczyć drogę (ciąg elementów zbioru S, którego pierwszym elementem jest 1), dla której suma kar jest minimalna.

Ponieważ macierz czasów C spełnia warunek trójkąta, więc rozwiązaniem optymalnym problemu dystrybucji jest droga Hamiltona (tj. droga zawierająca każdego odbiorcę dokładnie jeden raz). Rozpatrywane zagadnienie jest więc jednym z wielu uogólnień problemu komiwojażera (*TSP* - Traveling Salesman Problem) i należy do klasy problemów silnie NP-trudnych. Łatwo także zauważyć, że jest ono równoważne pewnemu problemowi szeregowania na jednej maszynie zadań z przezbrojeniami oraz żądanymi terminami zakończenia.

3. Problem szeregowania zadań z przezbrojeniami

Jednomaszynowy problem szeregowania zadań z najpóźniejszymi terminami zakończenia oraz przezbrojeniami (*Single Machine Scheduling Problem with Total Tardiness Criterion and Sequence-Dependent Setup Times*, w skrócie *STD*) oznaczany w literaturze przez $1|s_{ij}|\sum w_iT_i$ można sformułować następująco:

Problem: Niech $J = \{1,2,...,n\}$ będzie zbiorem zadań, które należy wykonać na jednej maszynie. W dowolnej chwili maszyna wykonuje co najwyżej jedno zadanie oraz wykonywanie zadania nie może być przerwane. Dla zadania $i \in J$ niech p_i, w_i, d_i będą odpowiednio: czasem wykonywania, wagą funkcji kary za spóźnienie oraz żądanym terminem zakończenia. Dane są także wartości $s_{i,j}$ reprezentujące czas potrzebny na przygotowanie maszyny do wykonywania zadania j, jeżeli bezpośrednio przed j było wykonywane zadanie i ($i, j \in J$), przy tym $s_{0,i}$ jest czasem przygotowania (uzbrojenia) maszyny, jeżeli zadanie i będzie wykonywane jako pierwsze. Należy

wyznaczyć kolejność wykonywania (permutację) zadań minimalizującą sumę kar za spóźnienia.

Niech Π będzie zbiorem permutacji elementów z J. Dla dowolnej kolejności wykonywania zadań (permutacji) $\pi \in \Pi$ czas zakończenia zadania $\pi(i) \in J$

$$C_{\pi(i)} = s_{0,\pi(1)} + \sum_{j=1}^{i-1} (p_{\pi(j)} + s_{\pi(j),\pi(j+1)}) + p_{\pi(i)}.$$

Wobec tego koszt (suma kar za spóźnienia) permutacji $\pi \in \Pi$

$$WT(\pi) = \sum_{i=1}^{n} w_{\pi(i)} T_{\pi(i)}. \tag{1}$$

Problem *STD* (z czasami przezbrojeń i żądanymi terminami zakończenia zadań) polega na wyznaczeniu w zbiorze permutacji Π minimalizującej wartość funkcji (1). W literaturze jest on oznaczany przez $1|s_{ij}|\sum w_i T_i$ i należy do klasy problemów silnie NP-trudnych. Algorytmy dokładne jego rozwiązywania są zazwyczaj adaptacjami algorytmów rozwiązywania problemu $1||\sum w_i T_i$, tj. bez przezbrojeń. Umożliwiają one rozwiązywanie (w rozsądnym czasie) przykładów z liczbą zadań nie większą niż 40. W pracy Lee i in. [6] zamieszczono szybkie algorytmy konstrukcyjne dla problemu *STD*, które wyznaczają rozwiązania różniące się, niestety, nawet o kilkaset procent od optymalnych. Najlepsze wyniki otrzymano, stosując metaheurystyki: algorytmu genetycznego (Armentano i Mazzini [1]), symulowanego wyżarzania (Tan i in. [8]), przeszukiwania z tabu (Sun i in. [7]), algorytmu mrówkowego (Gagne i in. [4]) oraz próbkowania stochastycznego (Cicirello i Smith [3]). W pracy Bożejko i Wodecki [2] przedstawiono algorytm hybrydowy oparty na idei analizy minimów lokalnych, wyznaczanych przez algorytm zstępujący. Wykonano obliczenia na referencyjnej grupie przykładów i dla wielu z nich otrzymano rozwiązania lepsze od najlepszych obecnie znanych.

W najlepszych obecnie algorytmach rozwiązywania problemu *STD* do intensyfikacji oraz dywersyfikacji obliczeń stosowane są metody probabilistyczne. Nie ma więc gwarancji powtarzalności wyników. Przedstawiony w dalszej części algorytm oparty na metodzie przeszukiwania z tabu jest w pełni deterministyczny.

4. Metoda przeszukiwania z tabu

Algorytmy oparte na metodzie przeszukiwania z tabu (*tabu search, TS*) są z powodzeniem stosowane do rozwiązywania NP-trudnych problemów optymalizacji kombinatorycznej. Metoda ta polega na iteracyjnym polepszaniu bieżącego rozwiązania przez lokalne przeszukiwanie. Rozpoczyna się od pewnego rozwiązania startowego x^0. W każdej iteracji, dla bieżącego rozwiązania x^i, wyznacza się jego otoczenie $N(x^i)$ - podzbiór zbioru rozwiązań dopuszczalnych. Otoczenie jest generowane przez ruchy, tj. pewne przekształcenia rozwiązania x^i. Następnie z otoczenia wyznaczamy najlepszy element x^{i+1}, który przyjmuje się za bieżące rozwiązanie w następnej iteracji. Aby zapobiec generowaniu, w nowych iteracjach, rozwiązań niedawno rozpatrywanych (powstawaniu cykli), zapamiętuje się ruchy (ich atrybuty) na liście rozwiązań zakazanych, tzw. liście tabu. Poniżej opiszemy główne elementy tej metody.

Ruch i otoczenie. W każdej iteracji algorytmu *TS* jest wyznaczane otoczenie, tj. podzbiór zbioru rozwiązań dopuszczalnych. Jest ono generowane przez ruchy - przekształcenia określone na zbiorze wszystkich permutacji. Niech k i l ($k < l$) będzie parą pozycji w permutacji $\pi \in \Pi$. Ruch typu *wstaw* (*insert*), oznaczany przez i_l^k, polega na przestawieniu zadania $\pi(k)$ z pozycji k na pozycję l (elementy znajdujące się na pozycjach pomiędzy k i l są przesuwane o jedną pozycję). Generuje on permutację $i_l^k(\pi) = \pi_l^k \in \Pi$. Wszystkich takich ruchów jest $n(n-1)$. Jeżeli

$$\pi^B = (\pi(a), \pi(a+1), ..., \pi(b-1), \pi(b)), \tag{2}$$

jest pewną subpermutacją ($1 \le a < b \le n$) permutacji $\pi \in \Pi$, to

$$WT(\pi^B) = \sum_{i=a}^{b} w_{\pi(i)} \cdot T_{\pi(i)},$$

nazywamy *kosztem*, a $L(\pi^B) = C_{\pi(b)} + s_{\pi(b),\pi(b+1)}$ *długością* π^B, gdzie

$$C_{\pi(j)} = s_{0,\pi(1)} + \sum_{i=1}^{j-1} (p_{\pi(i)} + s_{\pi(i),\pi(i+1)}) + p_{\pi(j)}, \quad i = 1, 2, ..., n.$$

Dalej, przez $\Pi(\pi^B)$ oznaczamy podzbiór permutacji z Π, które różnią się od π jedynie kolejnością elementów na pozycjach $a, a+1, ..., b$.

Dla subpermutacji π^B niech

$$F^-(\pi^B) = \min\{F(\delta) : \delta \in \Pi(\pi^B)\}, \; F^+(\pi^B) = \max\{F(\delta) : \delta \in \Pi(\pi^B)\},$$

gdzie F jest funkcją będącą kosztem WT lub długością L. Jest to więc odpowiednio dolne i górne ograniczenie wartości funkcji F dla permutacji ze zbioru $\Pi(\pi^B)$. Wobec tego, jeżeli permutacja $\gamma \in \Pi(\pi^B)$, to $F^-(\pi^B) \le F(\gamma) \le F^+(\pi^B)$. Oczywiście, dla funkcji WT i L powyższe minimum (tj. odpowiednio WT^- i L^-) oraz maksimum (tj. WT^+ i L^+) realizują zazwyczaj różne permutacje.

Subpermutacja π^B jest θ*-bliska* optymalnej w π, jeżeli

$$F(\pi^B) \in [AF^-, AF^- + \theta \cdot (AF^+ - AF^-)], \tag{3}$$

gdzie para AF^- i AF^+ jest pewnym przybliżeniem odpowiednio wartości $F^-(\pi^B)$ oraz $F^+(\pi^B)$. Parametr $\theta \in [0,1]$ wyznacza się zazwyczaj eksperymentalnie.

W problemie *STD* należy zminimalizować funkcję celu, która jest sumą kosztów opóźnień. Dla dowolnej permutacji $\pi \in \Pi$

$$WT(\pi) = \sum_{i=1}^{n} w_{\pi(i)} T_{\pi(i)} = \sum_{i=1}^{a-1} w_{\pi(i)} T_{\pi(i)} + \sum_{i=a}^{b} w_{\pi(i)} T_{\pi(i)} + \sum_{i=b+1}^{n} w_{\pi(i)} T_{\pi(i)},$$

gdzie π^B jest pewną subpermutacją określoną w (2). Zmiana kolejności elementów w subpermutacji π^B **może** wygenerować z π permutację o mniejszej wartości funkcji celu, jeżeli:

- zmniejszy się koszt wykonywania elementów z π^B, lub
- zmniejszy się moment rozpoczęcia wykonywania pierwszego zadania za π^B (długość π^B), czyli zadania $\pi(b+1)$.

Dowolną permutację (rozwiązanie dopuszczalne problemu *STD*) można rozbić na subpermutacje, zwane blokami, zawierające odpowiednio zadania terminowe lub zadania spóźnione. Subpermutacja π^B jest w permutacji $\pi \in \Pi$ odpowiednio *blokiem*:

1) *zadań terminowych*, jeżeli:

(a) każde zadanie z π^B jest w permutacji π terminowe,

(b) ze względu na długość L, subpermutacja π^B jest θ-bliska optymalnej,

2) *zadań spóźnionych*, jeżeli:

(a) dowolne zadanie z π^B przestawione na pozycję pierwszą w permutacji π jest spóźnione,

(b) subpermutacja π^B jest θ-bliska optymalnej ze względu na funkcję długości L oraz kosztów WT.

W pracy Wodeckiego [9] jest rozpatrywany jednomaszynowy problem szeregowania z najwcześniejszymi terminami rozpoczęcia i najpóźniejszymi terminami zakończenia zadań. Przedstawione tam metody i algorytmy zostały zaadaptowane (w algorytmie rozwiązywania problemu *STD*) do obliczania występujących w definicji θ-bliskości (3), przybliżeń F^- oraz F^+, a także rozbicia permutacji na bloki.

Niech

$$\Omega = [B^2, B^2, \ldots, B^v],$$

będzie rozbiciem permutacji $\pi \in \Pi$ na bloki. Generując otoczenie $N(\pi)$, pomijamy każdy z ruchów, który zmienia kolejność zadań jedynie w pewnym bloku. Przy odpowiednio dobranej wartości parametru θ pozwala to na eliminację średnio około 40% pełnego otoczenia, generowanego przez ruchy typu wstaw, bez pogarszania wartości wyznaczanych rozwiązań. Ponadto przyśpiesza to zbieżność algorytmu.

Lista tabu. Aby zapobiec powstawaniu cykli (powrotowi do tej samej permutacji, po niewielkiej liczbie iteracji algorytmu), pewne atrybuty każdego ruchu zapamiętuje się na liście ruchów zakazanych. Wykonując ruch i_j^r (tj. generując z π permutację π_j^r), na listę tabu zapisujemy atrybuty tego ruchu, trójkę: $(\pi(r), j, WT(\pi_j^r))$. Załóżmy, że rozpatrujemy ruch i_l^k generujący permutację β_l^k. Jeżeli na liście tabu jest trójka (r, j, Ψ) taka, że $\beta(k) = r$, $l = j$ oraz $F^{n-i}(\beta_l^k) \geq \Psi$, to ruch jest pomijany.

5. Eksperymenty obliczeniowe

Algorytm rozwiązywania jednomaszynowego problemu szeregowania zadań z przezbrojeniami i najpóźniejszymi terminami zakończenia został zaimplementowany w języku C++ i uruchomiony na komputerze z procesorem Pentium IV 1,8 GHz. Eksperymenty obliczeniowe przeprowadzono na 120 przykładach (każdy o rozmiarze 60) zamieszczonych na stronie [5]. Otrzymane wyniki porównano z obecnie najlepszymi z pracy Cicirello i Smitha [3] oraz Bożejki i Wodeckiego [2] (oba te algorytmy nie są deterministyczne). W stosunku do wartości referencyjnych (najlepszych obecnie znanych rozwiązań wyznaczonych przez różne algorytmy) średni błąd względny wynosi 3.81%, a błąd maksymalny 8.57%. Dla 14 przykładów otrzymano rozwiązanie optymalne. Średni czas obliczeń jednego przykładu (dla 3000 iteracji) wynosił 16.3 s, a maksymalny 31.8 s. Przedstawiony algorytm jest w pełni deterministyczny. Okazał się nieznacznie gorszy od algorytmu zamieszczonego w pracy [2] (średni błąd względny wynosi 2.06%), a jednocześnie zdecydowanie lepszy od algorytmu z [3] (średni błąd względny wynosi -8.47%). Otrzymane wyniki, z punktu widzenia zastosowań, są w pełni zadowalające. Zasadniczo wyeliminowanie

bloków z algorytmu nie wpływa na czas obliczeń, jednak średni błąd względny wzrasta o około 15%.

BIBLIOGRAFIA

1. Armentano V.A., Mazzini R.: A genetic algorithm for scheduling on a single machine set-up times and due dates. Prod. Plan. & Control, 11, 2000, p.713-720.
2. Bożejko W., Wodecki M.: A parallel metaheuristics for the single machine total weighted tardiness problem with sequence-dependent setup times, Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 2007), p. 96-103.
3. Cicirello V.A., Smith S.F.: Enhancing stochastic search performance by value-based randomization of heuristics. Journal of Heuristics, 11, 2005, p.5-34.
4. Gagné C., Price W.L., Gravel M.: Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times. Journal of the Operational Research Society, 53, 2002, p.895-906.
5. http://www.ozone.ri.cmu.edu/benchmarks.html
6. Lee Y.H., Bhaskaran K., Pinedo M.: A heuristic to minimize the total weighted tardiness with sequence-dependent setups. IIE Transactions, 29, 1997, p.45-52.
7. Sun X., Noble J.S., Klein C.M.: Single-machine scheduling with sequence dependent setup to minimize total weighted squared tardiness. IIE Transactions, 31, 1999, p.113-124.
8. Tan K.C., Narasimban R., Rubin P.A., Ragatz G.L.: A comparison on four methods for minimizing total tardiness on a single processor with sequence dependent setup times. Omega, 28, 2000, pp.313-326.
9. Wodecki M.: A block approach to earliness-tardiness scheduling problems, Advanced Manufacturing Technology, (DOI: 10.1007/s00170-008-1395-7), 2008.

Recenzent: Prof. dr hab. inż. Eugeniusz Toczyłowski

Abstract

In the paper we consider a distribution problem in which goods have to be distributed to many receivers before a due date. If this term is exceeded, a penalty is counted. The problem consists in planning a tour such, that a sum of penalties will be as low as possible. A tabu search algorithm is proposed in the paper to solve this problem. It is equivalent to a single machine scheduling problem, which is described by $1|s_{ij}|\Sigma w_i T_i$ in the literature and it belongs to strongly NP-hard class. An idea of so-called blocks is presented. It is possible to reduce calculations by 50% using these block properties. Calculations were done on representative group of test instances, obtained results were compared to the best known solutions from the literature.