ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1997
Seria: INFORMATYKA z. 33 Nr kol. 1377

Sebastian DEOROWICZ
Politechnika Śląska

ROZWIĄZYWANIE PROBLEMU DOSTAWY ZA POMOCĄ ALGORYTMÓW K-OPTYMALNYCH

Streszczenie. Problem dostawy jest przykładem złożonej optymalizacji kombinatorycznej i należy do grupy problemów NP-zupełnych. Ponieważ do rozwiązywania innych problemów tej klasy dobrze nadają się algorytmy k-optymalne, podjęto próbę zaadaptowania tych algorytmów do problemu dostawy. W pracy przedstawiono wyniki działania wybranych modyfikacji algorytmu k-optymalnego dla problemu dostawy dla zbiorów o różnych rozmiarach.

SOLVING THE DELIVERY PROBLEM USING K-OPTIMAL ALGORITHMS

Summary. The Delivery Problem is a difficult combinatorial optimization problem that belongs to the NP-complete group. Since the k-optimal algorithms are suitable for solving the problems of this class, we have tried to adopt these algorithms to the delivery problem. In the paper the computational results of some modifications of the k-optimal algorithm for solving the delivery problem for the sets of various sizes are presented.

1. Wprowadzenie

Problem rozbicia zbioru, który wraz z pokrewnymi problemami pokrycia i upakowania tworzy grupę zagadnień pokrycia, należy do klasy tzw. problemów NP-zupełnych, tj. takich, dla których nie jest znany deterministyczny algorytm o złożoności wielomianowej znajdujący rozwiązanie optymalne. W niniejszym artykule zajmiemy się szczególnym przypadkiem problemu rozbicia zbioru, jakim jest problem dostawy. Dla tego problemu istnieje kilka metod poszukiwania rozwiązania optymalnego, jednak charakteryzują się one wykładniczą złożono-

ścią czasową, w związku z czym metody te tracą zastosowanie dla zbiorów danych wejściowych o dużej mocy.

W niniejszym artykule przedstawiono inne podejście do tego problemu. Ponieważ w praktycznych zastosowaniach najczęściej nie jest konieczne znalezienie rozwiązania optymalnego i wystarczające jest znalezienie rozwiązania do niego zbliżonego, zdecydowano się więc na zastosowanie sprawdzonych w innych problemach NP-zupełnych algorytmów k-optymalnych.

2. Sformułowanie problemu dostawy

Niech dany będzie zbiór bazowy $N = \{1,2,3,...,n\}$ oraz rodzina podzbiorów tego zbioru $P = \{P_j\}_{j \in M}$, gdzie $M = \{1,2,3,...,m\}$. Zdefiniujmy tzw. *funkcję kosztu F* przyjmującą wartości nieujemne dla każdego podzbioru zbioru bazowego N:

$$\forall_{P_j \in P} F(P_j) = c_j, \text{ gdzie } c_j \geq 0 \tag{1}$$

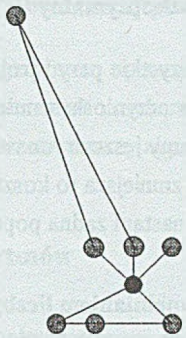
Rozbicie zbioru N definiujemy jako podzbiór Q rodziny P, której wszystkie elementy są rozłączne i pokrywają wszystkie elementy zbioru N.

Funkcja kosztu dla rozbicia zbioru jest zdefiniowana jako suma wartości funkcji kosztów dla podzbiorów wchodzących w skład rozbicia:

$$F(Q) = \sum_{i=1}^{k} F(Q_i), \tag{2}$$

gdzie $Q_i \in P$. Problem rozbicia zbioru polega na znalezieniu rozbicia Q o minimalnym koszcie.

Szczególnym przypadkiem rozbicia zbioru, rozważanym w tym artykule, jest zagadnienie dostawy, gdzie zbiór bazowy N reprezentuje zbiór klientów (miast, węzłów), którzy mają być obsłużeni przez bazę-magazyn. Zakładamy, że magazyn dysponuje nieograniczoną liczbą samochodów dostawczych, służących do dostarczania towarów do klientów. Ograniczona jest jedynie liczba klientów, którzy mogą być obsłużeni w ramach jednej trasy; liczbę tę oznaczamy przez k (dalsze rozważania dotyczą przypadku, gdy $k = 3$). Każdej możliwej trasie obejmującej od 1 do k klientów przyporządkowany jest koszt będący długością trasy. Zadanie polega na znalezieniu takiej kombinacji tras, której koszt będzie minimalny, a każdy element zbioru bazowego (węzeł) pokryty będzie jednokrotnie. Na rys. 1 przedstawiony jest optymalny zbiór tras dla $n = 7$ i $k = 3$ (czarny punkt oznacza magazyn, zaś szare punkty - klientów).



Rys. 1. Przykład optymalnego rozwiązania dla $n = 7$ i $k = 3$
Fig.1. Sample of optimal solution for $n = 7$, $k = 3$

3. Algorytmy

W dalszej części przedstawimy kilka algorytmów i wyniki ich działania dla pewnego zbioru danych testowych. Wszystkie z omówionych algorytmów są modyfikacjami algorytmów 2- i 3-optymalnego dla problemu komiwojażera [1] zaadaptowanymi do problemu dostawy.

3.1. Wstępne grupowanie w trasy

Jako że opisane poniżej algorytmy zakładają znajomość pewnego rozwiązania i starają się go zoptymalizować, więc konieczne jest wstępne pogrupowanie miast w trasy o rozmiarze nie większym niż 3. Tak przygotowany zbiór tras można następnie poddać optymalizacji. Sprawdzono doświadczalnie, że dla wszystkich z algorytmów z wyjątkiem 3-OPTU (patrz pkt. 3.4) nie jest istotne, jak to grupowanie będzie wykonane. W związku z tym w tej pracy założono, że miasta są losowo grupowane w trasy o losowym rozmiarze (od 1 do 3).

3.2. Algorytm „Nowa trasa" (NT)

W algorytmie tym przeglądamy wszystkie pary i trójki miast i dla każdej z nich próbujemy utworzyć nową trasę (skracając jednocześnie trasy, do których te miasta wcześniej należały). Jeżeli ta zmiana przyniosła zmniejszenie kosztu, to ją zatwierdzamy, a jeżeli nie, cofamy. Algorytm wykonujemy tak długo, aż w trakcie jednego przebiegu nie nastąpi żadna poprawa kosztu.

3.3. Algorytm „Nowa trasa" plus 3-optymalny (NT-3)

W algorytmie tym przeglądamy wszystkie pary i trójki miast i dla każdej z nich próbujemy utworzyć nową trasę. Jeżeli zmiana ta przyniosła zmniejszenie kosztu, to ją zatwierdzamy, a jeżeli nie, cofamy. Następnie przeglądamy jeszcze raz wszystkie trójki miast i zamieniamy miasta ze sobą miejscami w trasach, jeżeli zmniejsza to koszt. Algorytm wykonujemy cyklicznie do momentu, gdy w jednym przejściu nie nastąpi żadna poprawa kosztu.

3.4. Algorytm 3-optymalny z uzupełnianiem liczby miast (3-OPTU)

W algorytmie tym, inaczej niż we wszystkich innych tu przedstawionych, istotne jest wstępne pogrupowanie miast w trasy. Miasta grupujemy w trasy o rozmiarze 3, a jeżeli liczba miast nie jest podzielna przez 3, to uzupełniamy ostatnią trasę jednym lub dwoma miastami pomocniczymi, które pokrywają się z punktem oznaczającym magazyn. Następnie przeglądamy wszystkie trójki miast i zamieniamy miasta ze sobą miejscami w trasach, jeżeli zmniejsza to koszt. Algorytm wykonujemy cyklicznie do momentu, gdy w jednym przejściu nie nastąpi żadna poprawa kosztu.

Należy podkreślić, że dla pewnych zbiorów danych algorytm może nie być w stanie znaleźć optymalnego rozwiązania. Wynika to z tego, że zakładamy tu arbitralnie pewną maksymalną liczbę tras, z których może składać się rozwiązanie.

```
1  dodaj 3⌈n/3⌉ − n pseudo-klientów pokrywających się z magazynem;
2  min_koszt := ∞;
3  for i := 1 to liczba_iteracji do
4      ułóż klientów w losowym porządku;
5      stwórz trasę dla każdej kolejnej trójki klientów;
6      repeat
7          zmalał_koszt := false;
8          for wszystkich trójek i, j, k klientów do
9              zamień klientów i, j, k w trasach używając rotacji i ← j ← k ← i oraz
                   i → j → k → i i oblicz nowy_koszt jako mniejszy koszt z obu rozwiązań;
10             if nowy_koszt < min_koszt then
11                 zapamiętaj rozwiązanie; min_koszt := nowy_koszt;
12                 zmalał_koszt := true;
13             else przywróć poprzednie rozwiązanie; end if;
14         end for;
15     until zmalał_koszt = false;
16 end for;
```

3.5. Algorytm Heurystyka plus "Nowa trasa" plus 3-optymalny (HNT-3)

Algorytm ten jest identyczny z algorytmem opisanym w punkcie 3.3 z jedną tylko ważną zmianą. Na samym początku dla zbioru miast uruchamiamy algorytm heurystyczny i dopiero zbiór tras przez niego przygotowany przetwarzamy algorytmem NT-3.

4. Wyniki testowania algorytmów

Wszystkie algorytmy zostały zaimplementowane w języku C i były testowane za pomocą tej samej grupy zbiorów danych testowych na komputerze Pentium-133. W grupie 15 zbiorów danych liczba miast zawierała się w przedziale od 5 do 30. Dla każdego zbioru danych testowych uruchamiano każdy algorytm 1000 razy (za każdym razem losując kolejność miast w zbiorze wejściowym) i notowano koszty najlepszego, najgorszego rozwiązania oraz średnią kosztów wszystkich rozwiązań.

Dla porównania jakości rozwiązań znaleziono rozwiązania optymalne dla 14 z 15 badanych zbiorów danych testowych (dla wszystkich o rozmiarze nie większym niż 20) za pomocą algorytmu Branch & Bound.

Jako miarę jakości algorytmu przyjęto różnicę procentową pomiędzy średnim kosztem rozwiązań znajdowanych przez algorytm a kosztem najlepszego znalezionego rozwiązania. Dla każdego algorytmu obliczono średnią arytmetyczną różnic dla wszystkich zbiorów. Otrzymane wyniki przedstawia tabela 1.

Tabela 1

Wyniki działania algorytmów dla $n \in (5,30)$

Algorytm	Średnia różnica [%]
NT	1,37
NT-3	0,29
3-OPTU	0,09
HNT-3	0,15

Jak wynika z powyższej tabeli, dla badanych zbiorów danych wszystkie algorytmy dają wyniki dość dobre (każdy z nich przynajmniej raz znalazł rozwiązanie optymalne). Okazuje się jednak, że jakość wyników otrzymywanych za pomocą algorytmu NT szybko spada w miarę zwiększania się liczby miast. Do dalszego bardziej wnikliwego testowania wybrano więc algorytmy NT-3, 3-OPTU i HNT-3.

Kolejnym etapem było przetestowanie zachowania się algorytmów dla większych problemów. W tym celu użyto po 4 różne zbiory danych o rozmiarach 30, 40, 50 i 60 miast. Wyniki działania algorytmów dla tych zbiorów przedstawia tabela 2. Kolumny „Różnica" zawierają średnią arytmetyczną różnic (obliczanych tak samo, jak w poprzednim teście) dla zbiorów danych o tych samych rozmiarach.

Tabela 2

Wyniki działania algorytmów NT-3, 3-OPTU i HNT-3 dla $n \in \{30,40,50,60\}$

Liczba miast	Różnica [%] NT-3	Różnica [%] 3-OPTU	Różnica [%] HNT-3
30	0,50	0,75	0,54
40	0,83	0,80	1,09
50	0,79	0,58	0,91
60	0,87	0,68	0,91
Średnia	0,75	0,70	0,86

5. Wnioski końcowe

Dla małych zbiorów danych ($n < 30$) wszystkie algorytmy przedstawione w tej pracy dają wyniki dość dobre. Jednak w miarę wzrostu rozmiaru zadania okazuje się, że algorytm NT jest wyraźnie gorszy od pozostałych. Dość nietypowo zachowuje się również algorytm HNT-3. Wydawało się, że zastosowanie heurystyki do przygotowania wstępnego rozwiązania poprawi jakość algorytmu NT-3. Okazało się jednak, że otrzymywane wyniki są nieco gorsze, a czasami wręcz algorytm HNT-3 nie jest w stanie znaleźć rozwiązań, do których bez problemu dochodzą algorytmy NT-3 i 3-OPTU.

Zdecydowanie najlepsze wydają się więc algorytmy NT-3 i 3-OPTU. Trzeba jednak pamiętać, że algorytm 3-OPTU może dla pewnych zbiorów danych dawać rozwiązania dużo gorsze, gdyż zakłada on z góry pewną maksymalną liczbę tras, z których składa się rozwiązanie. Z testów przeprowadzonych dla kilkudziesięciu zbiorów danych wynika jednak, że takie sytuacje są dość rzadkie (dla badanych zbiorów najlepsze rozwiązania otrzymywane przez 3-OPTU i NT-3 były identyczne).

Istotnym faktem jest też to, że zarówno algorytm NT-3, jak i 3-OPTU dla danych z tabeli 2 znajdował najlepsze rozwiązania z prawdopodobieństwem ponad 30% (NT-3 - 32%, a 3-OPTU - 35%), natomiast prawdopodobieństwo to dla algorytmu HNT-3 wyniosło jedynie 23%.

Warto na koniec zauważyć, że wszystkie przedstawione tu algorytmy można stosować zarówno dla problemów planarnych, jak i nieplanarnych, gdyż korzysta się w nich jedynie z odległości między miastami.

LITERATURA

[1] Sysło M. M., Deo N., Kowalik J. S.: Algorytmy optymalizacji dyskretnej. PWN, Warszawa 1995, s. 303-309.

[2] Papadimitriou C. H., Steiglitz K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Englewood Cliffs, NJ, 1982, rozdz. 19.

[3] Reeves C. R., (Ed.): Modern Heuristic Techniques for Combinatorial Problems, Prentice-Hall, Englewood Cliffs, NJ, 1983.

Recenzent: Dr inż. Mariusz Boryczka

Wpłynęło do Redakcji 3 marca 1997 r.

Abstract

The Set Partitioning Problem (SPP) is a difficult combinatorial optimization problem which belongs to the NP-complete problems. The special case of the Set Partitioning Problem is a Delivery Problem (DP). In this paper an adaptation of the k-optimal algorithms to solve the DP is presented.

We discuss four modifications of the k-optimal algorithm. Each of them was tested on a group of 15 sets of different sizes ranging from 5 to 30 cities. All the algorithms were run 1000 times for each set and the costs of the best solution and the average of the costs of all solutions were found. The difference between the optimal cost (if known, or the best known otherwise) and the average cost for each set was taken as the basis for comparison of qualities of the algorithms. For each algorithm the average of differences for all sets was computed. The obtained results are presented in table 1.

The results for the NT algorithm were not satisfactory, therefore only the three remaining algorithms were taken to the future tests on the sets of sizes 30, 40, 50 and 60 cities. The results of these tests are presented in table 2. The columns denoted by „Różnica" include the average of the differences (computed as in the previous tests) for the sets of the same size.

The tests show that the NT-3 and 3-OPTU algorithms are the best. The differences between the optimal solution and the solutions obtained by both algorithms are very small. Both algorithms find the best known solutions with the probability greater then 30%.