Tomasz AMBROZIAK*, Roland JACHIMOWSKI*

ALGORYTM KLASTERYZACJI W ZASTOSOWANIU DO PROBLEMU TRASOWANIA POJAZDÓW

Streszczenie

W artykule scharakteryzowano problematykę klasteryzacji punktów obsługi dla problemu trasowania pojazdów. Przybliżono wybrane metody klasteryzacji opisywane w literaturze przedmiotu. Zaproponowano algorytm klasteryzacji punktów obsługi zmniejszający złożoność obliczeniową problemu trasowania pojazdów.

Słowa kluczowe: problem trasowania pojazdów, klasteryzacja, algorytm klasteryzacji

1. WPROWADZENIE

Problem trasowania pojazdów (*ang. Vehicle routing problem VRP*) jest rozwinięciem jednego z najstarszych problemów optymalizacyjnych na sieciach – problemu komiwojażera (*ang. Traveling Salesman Problem TSP*), który polega na odwiedzeniu dokładnie jeden raz każdej z wybranych miejscowości i powrocie do miejscowości, z której rozpoczęto podróż [7]. Znane są koszty przejazdu miedzy każdą parą miejscowości. Należy zaplanować komiwojażerowi drogę przejazdu w taki sposób, aby mógł on odwiedzić każdą miejscowość dokładnie jeden raz i całkowity koszt podróży był możliwie najmniejszy. W najprostszej formie problem trasowania pojazdów różni się od problemu komiwojażera dodatkowymi ograniczeniami związanymi z liczbą oraz ładownością pojazdów [1].

Problem trasowania pojazdów podobnie jak problem komiwojażera jest zagadnieniem *NP*-trudnym, tzn. nie istnieją dla tego zagadnienia algorytmy dokładne rozwiązujące ten problem w czasie zależącym wielomianowo od liczby analizowanych danych. Dowód *Np*-trudności problemu trasowania pojazdów znaleźć można w pracy [9]. Złożoność problemu trasowania pojazdów jest rzędu O(*n*-1!) przy czym *n* oznacza liczbę punktów, jakie należy odwiedzić. Czas trwania obliczeń rośnie wykładniczo względem liczby danych (w tym przypadku względem liczby odwiedzanych punktów, miast). Przykładowo, dla przypadku z 15 miastami liczba możliwych kombinacji trasy przejazdu wyniesie 87.178.291.200, a już dla 20 miast liczba ta wzrośnie prawie 1,4 miliona razy do wartości $1{,}21645 \cdot 10^{17}$. Dla komputera wykonującego milion operacji w ciągu sekundy czas trwania obliczeń dla 20 miast wyniesie prawie 4 tysiące lat.

W związku z dość długim czasem trwania obliczeń naukowcy skupiają się na poszukiwaniu algorytmów przybliżonych (heurystycznych), które dostarczałyby zadowalająco dobrych wyników w akceptowalnym czasie. Jednym ze sposobów osiągnięcia tego celu jest zmniejszenie złożoności rozpatrywanego przypadku problemu trasowania pojazdów. W literaturze cel ten osiągany jest poprzez stosowanie tzw. metod dwufazowych:

- najpierw klaster potem trasa (ang. *cluster first/route second*),
- najpierw trasa, potem klaster (*ang. route first/cluster second*).

Słowo „*klaster*" w tym przypadku odnosi się do podzbiorów wydzielonych ze zbioru obsługiwanych punktów. Każdy podzbiór obsługiwany jest przez pojedynczy pojazd. Tym

* Politechnika Warszawska, Wydział Transportu

samym przestrzeń rozwiązań maleje wraz ze wzrostem liczby klastrów. Jeżeli problem z 20 miastami podzielimy na 5 klastrów i przydzielimy do każdego klastra po jednym pojeździe, to każdy z przydzielonych pojazdów będzie miał do obsłużenia zaledwie 4 miasta. Dla takiej liczby miast w klastrze, liczba możliwych tras przejazdu wyniesie 24. Łączna długość trasy dla rozważanego przypadku będzie sumą długości tras w poszczególnych klastrach.

Taki podział zbioru obsługiwanych punktów na mniejsze podzbiory ma również swoje odzwierciedlenie w rzeczywistych problemach trasowania pojazdów. Sytuacja taka dotyczy najczęściej przedsiębiorstw zajmujących się dostawą towarów do klientów, których lokalizacja określona jest np. kodem pocztowym.

2. METODY KLASTERYZACJI W LITERATURZE

W literaturze dotyczącej rozwiązywania problemu trasowania pojazdów pojawiają się 3 zasadnicze grupy metod zmniejszania jego złożoności oparte na zasadzie *najpierw klaster potem trasa*:

- podstawowe metody klasteryzacji [6], [5], [3];
- metody wykorzystujące algorytm podziału i ograniczeń (*ang. branch & bound*) [4];
- metody wykorzystujące *petal algorithm* [8].

Podstawowe metody klasteryzacji dokonują prostego podziału punktów obsługi na klastry, a następnie realizują trasowanie pojazdów w tych klastrach. Wśród algorytmów realizujących podstawowe metody klasteryzacji wyróżnia się:

- Sweep Algorithm [6] – w którym klastry tworzone są poprzez obracanie promienia obsługi, którego środek zlokalizowany jest w bazie magazynowej. Następnie trasy pojazdów uzyskiwane są poprzez zrealizowanie problemu trasowania pojazdów w każdym klastrze.

- Fisher & Jaikumar Algorithm [5] – w którym przydział odbiorców do klastrów realizowany jest poprzez rozwiązanie problemu przydziału (*ang. general assignment problem*). Procedura tworzenia klastrów rozpoczyna się od wybrania kilku sztucznych punktów *„j”* w celu zainicjowania klastrów. Następnie do tych sztucznych punktów przydzielane są faktyczne punkty odbioru *„i”* na podstawie kosztu przydziału obliczanego z następującej zależności:

$$K_i = \min\{k_{0i} + k_{ij} + k_{j0};\ k_{0j} + k_{ji} + k_{i0}\} - (k_{0j} + k_{j0}) \tag{1}$$

gdzie:
0 – oznacza bazę dla pojazdów,
K_i – koszt dołączenia do obsługi i-tego punktu,
k_{ij} – koszt transportu pomiędzy i-tym a j-tym punktem.

Kolejny krok algorytmu tej metody zakłada rozwiązanie problemu przydziału, a w celu wyznaczenia tras rozwiązywany jest problem komiwojażera.

- Bramel & Simchi-Levi Algorithm [3] – w którym sztuczne punkty inicjujące klastry wyznaczane są poprzez rozwiązanie problem lokalizacji z ograniczeniami na pojemność lokalizowanych punktów (*ang. capacited location problem*) w celu zminimalizowania odległości faktycznych punktów odbioru od ich najbliższych sztucznych punktów. Po zlokalizowaniu punktów inicjujących (sztucznych), faktyczne punkty odbioru są stopniowo dołączane na podstawie najniższego kosztu ich dołączenia do punktów sztucznych. Po dołączeniu wszystkich punktów odbioru tworzone są klastry, w których rozwiązywany jest problem komiwojażera.

W przypadku metody wykorzystującej algorytm *branch&bound,* jej autorzy w przeciwieństwie do podstawowych metod klasteryzacji, nie skupiają się na tworzeniu klastrów odbiorców, lecz na konstruowaniu zbiorów dopuszczalnych tras pojazdów. Liczba poziomów w drzewie poszukiwań jest bezpośrednio determinowana przez liczbę tras pojazdów, które na każdym poziomie stanowią dopuszczalne rozwiązania problemu. Szczegółowo zasadę działania tego algorytmu można znaleźć w pracy [4].

Algorytm *Petal* zastosowany do problemu klasteryzacji jest rozwinięciem przedstawionego wcześniej algorytmu *Sweep*, jednakże podobnie jak algorytm wykorzystujący branch&bound tak i w tym przypadku zamiast klastrów odbiorców tworzone są zbiory tras. Końcowa selekcja tras dokonywana jest za pomocą *problem podziału zbioru*. Problem podziału polega w tym przypadku, na znalezieniu minimum funkcji:

$$\sum_{t \in T} k_t x_t \rightarrow \min \tag{2}$$

przy ograniczeniach postaci:

$$\sum_{t \in T} z_{it} x_t = 1; \quad i = 1, \ldots, n \tag{3}$$

gdzie:
T – zbiór tras,
x_t – zmienna binarna przyjmująca wartość 1, gdy t-ta trasa należy do rozwiązania,
z_{it} – parametr binarny przyjmujący wartość 1, gdy i-ty punkt obsługi wchodzi do t-tej trasy,
k_t – koszt t-tej trasy.

Szczegółowo zasada działania algorytmu omówiona została w pracy [8].

Metoda zmniejszania złożoności problemu trasowania pojazdów zgodnie z zasadą *najpierw trasa potem klaster* składa się w zasadzie z dwóch etapów. W pierwszym etapie tworzona jest jedna trasa dla wszystkich punktów obsługi, nieuwzględniająca żadnych ograniczeń. W drugim etapie trasa ta jest dzielona na krótsze trasy, przy uwzględnieniu nakładanych ograniczeń (brak jest ograniczeń dotyczących liczby pojazdów), dzięki czemu każda nowopowstała trasa dla pojazdu jest trasą dopuszczalną. W pracy [2] autor udowodnił, że drugi etap metody może być z powodzeniem rozwiązywany z wykorzystaniem zagadnienia *najkrótszej ścieżki* w acyklicznym grafie. W powyższej sytuacji złożoność problemu trasowania pojazdów zredukowana zostaje do poziomu $O(n^2)$. W literaturze istnieje wiele algorytmów rozwiązywania problemu wyznaczania najkrótszej ścieżki w grafie, przy czym najczęściej stosowany jest algorytm Dijkstry, którego opis zawiera większość pozycji literatury z zakresu badań operacyjnych.

3. ALGORYTM KLASTERYZACJI DLA PROBLEMU TRASOWANIA POJAZDÓW

Algorytm klasteryzacji odbiorców opracowano dla jedno-bazowego problemu trasowania pojazdów. Zakłada się, że zbiór pojazdów jest heterogeniczny, a każdy pojazd oprócz ładowności scharakteryzowany jest jednostkowym kosztem zużycia paliwa oraz maksymalnym czasem pracy w ciągu dnia. Czasy przejazdu pomiędzy wyróżnionymi punktami obsługi wyznaczane są na podstawie odległości miedzy nimi i przyjętej średniej prędkości jazdy. Celem prawidłowego zrozumienia działania algorytmu wyjaśnieniu wymagają następujące oznaczenia pojawiające się w treści zapisu algorytmu:
PO – (1, 2, …i, i'…, I) – uporządkowana lista punktów obsługi, gdzie i oznacza numer punktu obsługi. Bazę magazynową oznaczono numerem 0,
P – (1, 2, …p,…, P) – uporządkowana lista pojazdów, gdzie p oznacza numer pojazdu,
Q^p – ładowność p-tego pojazdu,

ZP^p – jednostkowe zużycie paliwa p-tego pojazdu,
t_i – czas obsługi i-tego punktu,
$t_{ii'}$ – czas transportu pomiędzy punktami i oraz i',
$d_{ii'}$ – odległość pomiędzy punktami i oraz i',
q_i – zapotrzebowanie na przewóz i-tego punktu obsługi,
q_k – zapotrzebowanie na przewóz k-tego klastra,
K – klaster.

Kolejne kroki algorytmu zapisano następująco:

Krok 1
Utwórz listę punktów obsługi *PO* i posortuj je rosnąco względem ich odległości od bazy.
Krok 2
Utwórz listę pojazdów *P* i posortuj pojazdy według malejącej wartości stosunku ładowności Q^p do jednostkowego zużycia paliwa ZP^p.
Krok 3
Określ maksymalną dozwoloną odległość D_{max} pomiędzy dowolną parą punktów obsługi.
Krok 4
Utwórz nową pustą listę *L*. Lista ta połączona będzie z nowotworzonym klastrem *k*. Pobierz największą wartość z listy *P* posortowanych pojazdów i przypisz ją do nowotworzonego klastra *K*. Następnie usuń ten pojazd z listy *P*.
Krok 5
Pobierz pierwszy węzeł *i* z listy *PO* i umieść na końcu listy *L*. Na podstawie parametrów tego punktu (zapotrzebowania na przewóz q_i) zainicjuj parametry nowotworzonego klastra. Tym samym tymczasowe zapotrzebowanie na przewóz nowotworzonego klastra q_k będzie równe zapotrzebowaniu na przewóz pierwszego punktu q_i dołączonego do klastra *K*.

$$q_k = q_i$$

Krok 6
Usuń pierwszy węzeł *i* z listy *PO* i zapisz tę listę jako *PO'*.
Krok 7
Pobierz pierwszy węzeł *i'* z listy *PO'* i sprawdź, czy aktualne zapotrzebowanie na przewóz q_k klastra (czyli wielkość zapotrzebowania na przewóz q_i punktu *i* wybieranego z listy *PO* w poprzednim kroku + zapotrzebowanie na przewóz *i'*-tego punktu $q_{i'}$) nie przekracza ładowności pojazdu do niego przydzielonego.

$$q_k = q_i + q_{i'} \le Q^p$$

Jeśli ładowność pojazdu została przekroczona, to usuń węzeł i' z listy PO' i powtórz **krok 7**.
Jeśli ładowność pojazdu nie jest przekroczona, to przejdź do **kroku 8.**
Krok 8
Sprawdź czy aktualny czas przebywania pojazdu w klastrze nie przekracza dopuszczalnego czasu pracy pojazdu T^p w ciągu dnia:

$$t_{0i} + t_i + t_{ii'} + t_{i'} + t_{i'0} \le T^p$$

Jeśli czas ten jest przekroczony, to należy usunąć punkt *i'* z listy *PO'* i powrócić do **kroku 7.**
Jeśli dopuszczalny czas pracy pojazdu nie został przekroczony, to przejdź do **kroku 9**.
Krok 9
Sprawdź czy odległość punktu *i'* do punktu *i* (zapisanego na liście *L* w **kroku 5**) nie przekracza dozwolonej maksymalnej odległości pomiędzy parą punktów:

$$d_{i'i} \le D_{max}$$

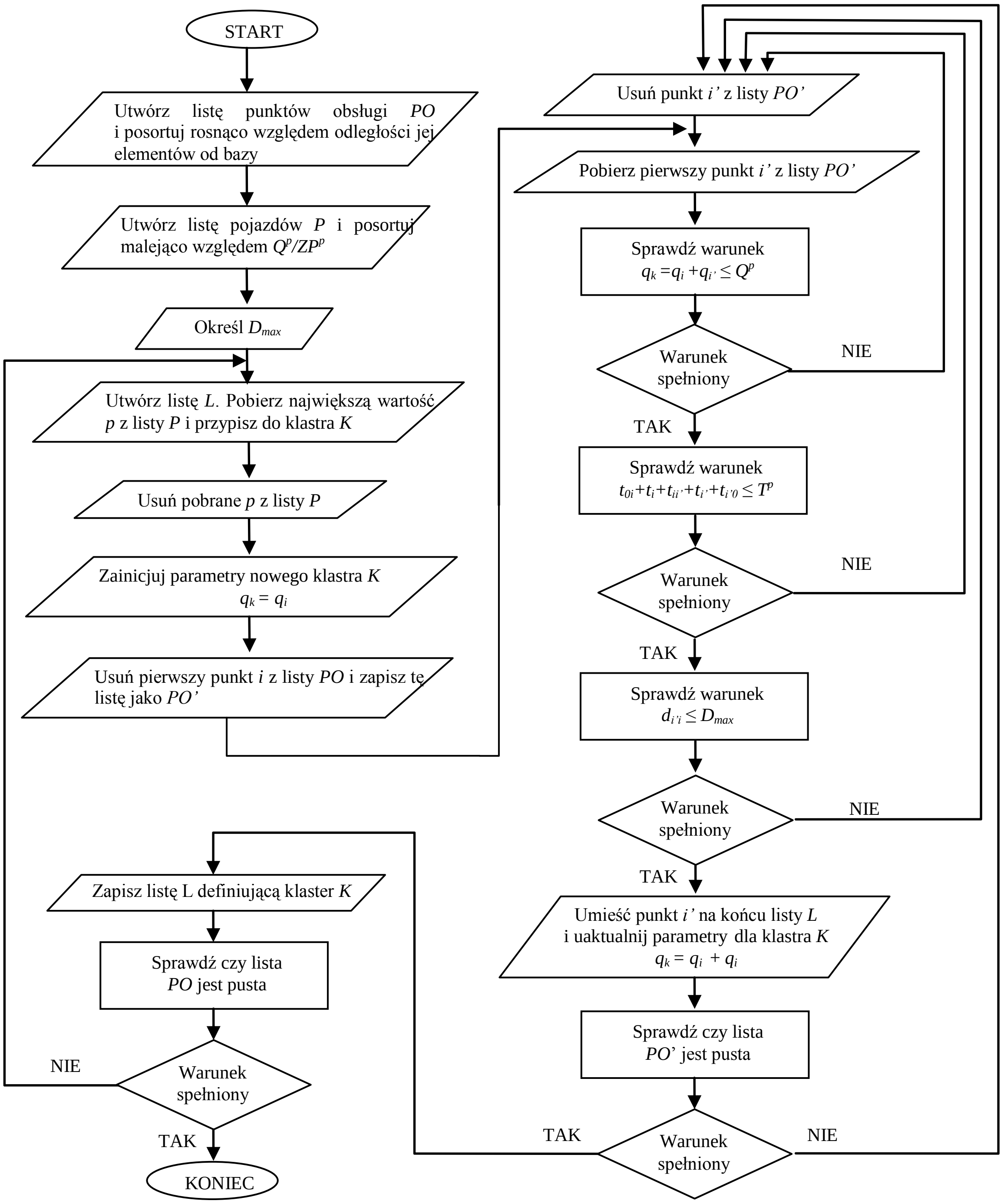
Jeśli odległość pomiędzy punktami przekracza D_{max}, to usuń punkt *i'* z listy *PO'* i powrócić do **kroku 7.**
Jeśli odległość pomiędzy punktami nie przekracza D_{max}, to przejdź do **kroku 10.**

Krok 10
Umieść punkt *i'* na końcu listy *L* i uaktualnij parametry dla klastra *K*.

$$q_k = q_i + q_i$$

Krok 11
Jeżeli lista *PO'* jest pusta, to zapisz listę *L* definiującą klaster *K*. W przeciwnym przypadku wróć do **kroku 7**.
Krok 12
Powtarzaj kroki od **4** do **11** do momentu, aż lista *PO* będzie pusta.

Schemat blokowy algorytmu przedstawiono na rys. 1.



Rys. 1. Schemat blokowy algorytmu klasteryzacji
Źródło: opracowanie własne.

4. PODSUMOWANIE

Przedstawiony w artykule algorytm klasteryzacji punktów obsługi można wykorzystać do rozwiązywania problemu trasowania pojazdów. Redukcja złożoności obliczeniowej za pomocą przedstawionego algorytmu pozwala na zastosowanie mniej skomplikowanych metod przeszukiwania przestrzeni rozwiązań (możliwych tras) rozpatrywanego problemu. Biorąc pod uwagę czas obliczeń oraz jakość uzyskiwanych rozwiązań, w przypadku dużych problemów trasowania, do wyznaczania kolejności obsługi punktów w poszczególnych klastrach można wykorzystać m.in. heurystyki konstrukcji lub ulepszania tras.

Adknowledge: Praca naukowa finansowana ze środków budżetowych na naukę w latach 2010-2012 jako projekt badawczy". Projekt N N509 601839 pt. Metodyka kształtowania sieci transportowo-logistycznej w wybranych obszarach.

LITERATURA

[1] Ambroziak T., Jachimowski R.: *Problematyka obsługi transportowej w jednoszczeblowym systemie dystrybucji*. Logistyka 4/2011: 17-24, (2011)
[2] Basley J.: *Route First-Cluster Second Methods for Vehicle Routing*. Omega, 11:403-408, (1983).
[3] Bramel J., Simchi-Levi D.: *A location based heuristic for general routing problems*. Operations Research, 43:649-660, (1995).
[4] Christofides N., Mingozzi A., Toth P.: *The vehicle routing problem*. In Combinatorial Optimization, Wiley UK, (1979).
[5] Fisher, M., Jaikumar R.: *A generalized assignment heuristic for vehicle routing*. Networks 11:109–124, (1981).
[6] Gillett, B., Miller L.: *A heuristic algorithm for the vehicle dispatch problem*. Operations Research 22:340–349, (1974).
[7] Jacyna M.: *Modelowanie i ocena systemów transportowych*, Oficyna Wydawnicza PW, Warszawa, (2009).
[8] Ryan D., Hjorring C., Glover F.: *Extensions of the petal method for vehicle routing.* Journal of Operational Research Society, 44:289-296, (1993).
[9] Savelsbergh, M.W.P. *Local search in routing problems with time windows*. Annals of Operations Research, 4:285-305, (1985).

CLUSTERING ALGORITHM FOR VEHICLE ROUTING PROBLEM

Abstract

The article characterizes the problem of service points clustering in vehicle routing problem. Selected clustering methods described in literature are reviewed. New heuristic service points clustering algorithm that reduces the computational complexity of the vehicle routing problem is presented.

Keywords: vehicle routing problem, clustering, clustering algorithm