

Problem dystrybucji towarów z najwcześniejszymi i najpóźniejszymi terminami dostaw

Dawid Ryznar, Krzysztof Zielonka

24 październik 2012

Opis problemu (orginalny)

Jako firma transportowa mamy dostarczyć towary do miast. Każde miasto ma ustaloną karę za spóźnienie lub przybycie zawczasie. Przemieszczenia się między miastami trwa pewną liczbę czasu. Należy znaleźć ciąg miast, dla którego ciężarówka odwiedza każde miasto i minimalizujący sumę kar jaką trzeba zapłacić za zbyt wczesne lub zbyt późne przybycie.

Nasze założenia

- Ciężarówka ma nieskończoną pojemność. Jest w stanie zabrać towary dla wszystkich miast.
- Dla każdego miasta mamy funkcje kary od czasu .
Ogólniejszy wariant kar.
- Czas jest w postaci liczby naturalnej np liczba sekund
- Znamy pewne górne ograniczenie czasowe, które określa maksymalny czas przejazdu.
- Rozładunek i załadunek nie wymaga czasu.
- W każdym mieście możemy czekać dowolną liczbę czasu zanim wyładujemy towar.
- Znamy wierzchołek startowy (magazyn).

Graf miast

W grafie miast, wierzchołki interpretujemy jako miasta, a wagi na krawędziach jako czasy potrzebne na przedostanie się z jednego miasta do drugiego. Czas na krawędzi $\{u, v\}$ jest najszybszym czasem potrzebnym na przedostanie się z miasta u do v .

- Graf miast spełnia nierówność trójkąta

$$a < b + c \wedge b < a + c \wedge c < a + b \quad (1)$$

$$\forall_{a,b,c \in V} t(\{a, c\}) < t(\{a, b\}) + t(\{b, c\}) \quad (2)$$

- Graf miast jest grafem pełnym

Jeżeli graf niespełnia jednego z powyższych założeń to możemy go do takiego przekonwertować wywołując dla każdego wierzchołka *BFS*.

Instancja problemu

Miastom przyporządkowujemy kolejno numery $2, \dots, n$. Dla uproszczenia będziemy zakładać, że magazyn zawsze ma numer 1. Instancją problemu jest para:

$$P = \langle T, p \rangle \quad (3)$$

Macierz czasu przjazdów

$$T = [t_{ij}]_{n \times n} \quad (4)$$

Kwadratowa macierz gdzie element t_{ij} to czas potrzebny na przejazd najszybszą drogą z miasta i do j .

Funkcja kary

$$p : N \rightarrow N \rightarrow R \quad (5)$$

Funkcja kary, przyjmująca kolejno numer miasta, czas rozładunku i zwracająca karę w postaci liczby rzeczywistej.

Rozwiązania dopuszczalne

Rozwiązaniem dopuszczalnym (spełniającym warunki zadania) jest permutacja liczb $1, \dots, n$.

Uzasadnienie

- W rozwiązaniu muszą znaleźć się wszystkie miasta. (definicja problemu)
- W rozwiązaniu miasta nie mogą się powtarzać. (nierówność trójkąta + założenie o nieskończonej ładowności ciężarówki + możliwość czekania w dowolnym mieście)
- W rozwiązaniu nie uwzględniamy magazynu. (to jest punkt startowy i końcowy + założenie o nieskończonej ładowności ciężarówki)

Funkcja celu

$$F : N^n \rightarrow R \quad (6)$$

$$F(v_1 \cdots v_n) = C(v_1, \cdots V_n, 0) \quad (7)$$

Funkcja pomocnicza

$$C : N^m \times N \rightarrow R \quad \text{gdzie } m > 0 \quad (8)$$

$$C(v, t) = \begin{cases} p(v, t) & \text{gd } t < t_{\max} \\ +inf & \text{wpp} \end{cases} \quad (9)$$

$$C(v_1, \cdots, v_n, t) = \min_{t \leq t_c \leq t_{\max}} \{C(v_2, \cdots, v_n, t_c + t_{1,2}) + p(v_1, t_c)\} \quad (10)$$

Złożoność problemu

Złożoność problemu

Problem dystrybucji towarów jest NP trudny. Udowodnimy to konstruując wielomianową redukcję problemu komiwojażera do problemu dystrybucji towarów.

Problem komiwojażera

Mamy dany pełny ważony graf G . Rozwiązaniem problemu jest minimalny cykl Hamiltona na tym grafie.

Dowód

Data: G - graf pełny wazony, w - funkcja wagowa

Result: trojka $\langle T, p, t_{max} \rangle$

f – funkcja przyporządkowująca wierzchołkom grafu kolejne liczby naturalne

g – funkcja przyporządkowująca krawędziom kolejne liczby naturalne

$$p(v, t) = \lambda(v, t) \rightarrow \sum_{\{u, v\} \in E} (2^{g(\{u, v\})} \& t) \cdot w(\{u, v\})$$

forall $v, u \in V$ **do**

$$T[f(v), f(u)] = 2^{g(\{v, u\})}$$

end

$$t_{max} = \sum_{i=1}^{|V|} w(f(v_{i-1}), f(v_i))$$

return $\langle T, p, t_{max} \rangle$

Algorytm konstrukcyjny

Opis algorytmu konstrukcyjnego

Algorytm konstrukcyjny oparliśmy na podstawie algorytmu Local Search. Sąsiadów wybieramy wykorzystując wszystkie możliwe transpozycje.

Local Search

Data: x – initial node

Result: *best_neighbour* – the best local node

current = none

best_neighbor = x

repeat

current = *best_neighbour* *neighbours* = find all neighbours of

current *best_neighbour* = select best from

neighbours $\cup \{current\}$

until $F(best_neighbour) == F(current)$;

return *best_neighbour*

Neighbors

Data: x – current node

Result: neighbours – set of x 's neighbours

neighbours =

forall $1 \leq i, j \leq n \wedge i < j$ **do**

$\text{neighbours} \cup = x_1 \cdots x_{i-1}, x_j, x_{i+1} \cdots x_{j-1}, x_i, x_{j+1} \cdots x_n$

end

return neighbours

Algorytm konstrukcyjny dla problemu dystrybucji towarów

Data: T, p, t_{max}, x – initial node

Result: $best_neighbor$ – best local neighbour

$current = none$

$best_neighbor = x$

repeat

$current = best_neighbor$

$neighbours = Neighbours(x) \cup \{current\}$ $best_neighbour =$

 select best from $neighbours$

until $F(best_neighbour) == F(current)$;

return $best_neighbor$