Problem dystrybucji towarów z najwcześniejszymi i najpóźniejszymi terminami dostaw

Dawid Ryznar, Krzystof Zielonka

24 październik 2012

Opis problemu (orginalny)

Jako firma transportowa mamy dostarczyć towary do miast. Każde miasto ma ustaloną kare za spóźnienie lub przybycie zawcześnie. Przemieszczenia się między miastami trwa pewną liczbę czasu. Należy znalęść 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 ogranicznie czasowe, które określa maksymalny czas przejazdu.
- Rozładunek i załadunek nie wymaga czasu.
- W kązdym miesic możemy czekać dowolną liczbe czasu znim wyładujemy towar.
- Znamy wierzchołek startowy (magazyn).

Graf miast

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

- Graf miast spełnia nierówność trójkąta

$$a < b + c \;\wedge\; b < a + c \;\wedge\; c < a + b \tag{1}$$

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

- Graf miast jest grafem pełnym

Jeżeli graf niespłnia jednego z powyższych założen to możemy go do takiego przekonwertować wywołujac dla każdego wierzchołka *BFS*.

Instancja problemu

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

$$P = \langle T, p \rangle \tag{3}$$

Macierz czasu przajazdów

$$T = [t_{ij}]_{n \times n} \tag{4}$$

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

Funkcja kary

$$p : N \to N \to R \tag{5}$$

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

Rozwiązania dopuszczalne

Rozwiązaniem dopuszczalnym (sepłniającym warunki zadania) jest permutacja liczb $1, \cdots, n$.

Uzasadnienie

- W rozwiąznu musza znaleść sie wszystkie miasta. (definicja problemu)
- W rozwiązaniu miasta nie mogą sie powtarzać. (nierówność trójkąta + założenie o nieskończonej ładowności ciężarówki + możliwość czekania w dowolnym mieście)
- W rozwiazniu nie uwzgledniamy magazynu. (to jest punkt startowy i końcowy + założenie o nieskończonej ładowności ciężarówki)

Funkcja celu

$$F : N^n \to R \tag{6}$$

$$F(v_1 \cdots v_n) = C(v_1, \cdots V_n, 0) \tag{7}$$

Funkcja pomocnicza

$$C : N^m \times N \to R \quad \text{gdzie } m > 0 \tag{8}$$

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

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

Złożonośc problemu

Złożnosc problemu

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

Problem komiwojażera

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

Złożonośc problemu

Dowód

Data: G -graf pełny wazony, w - funkcja wagowa
Result: trojka $\langle T, p, t_{max} \rangle$
f – funkcja przyporządkowwujaca wierzchołkom grafu kolejne liczby naturalne
$p(v,t) = \lambda(v,t) \to t$
forall $v, u \in V$ **do**
    $T[f(v), f(u)] = w(\{v,u\})$
end
$t_{max} = \sum_{i=2}^{|V|} w(f(v_{i-1}), f(v_i))$
return $\langle T, p, t_{max} \rangle$

Algorytm konstrukcyjny

Opis algorytmu konstrukcyjnego

Algorytm konstrukcyjny oparliśmy na podstawie algorytmu Local Search. Ssąsiądów 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**
    $neighbours \cup = x_1 \cdots x_{i-1}, x_j, x_{i+1} \cdots x_{j-1}, x_i, x_{j+1} \cdots x_n$
end
return neighbours

Algorytm konstrykcyjny 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 $best\_neighbor$