**Travelling salesman problem with a drone**

In rezolvarea acestei probleme am decis sa alegem articolul “*Optimization Approaches for the Traveling Salesman Problem with Drone*” - *Niels Agatz, Paul Bouman, Marie Schmidt.* In acest articol sunt descrisi doi algoritmi de rezolvare a problemei mentionate prin programare dinamica sau folosing metoda greedy.

1. **Greedy**

Ideea de bază este simpla: avand o problema de optimizare, de calcul al unui cost minim sau maxim, se va alege la fiecare pas decizia cea mai favorabila, fara a evalua global eficienta solutiei.

Descrierea solutiei:

* Se construieste un drum pentru a se obtine ordinea in care sunt parcurse nodurile de catre camion fara a fi luata in calcul drona (R = r0, r1...rn); solutia se obtine prin impartirea drumului in doua secvente ce reprezinta nodurile parcurse de camion si de drona.
* Fiecare nod are asociata una dintre etichetele:
  + "truck" - nod vizitat de camion
  + "drone" - nod vizitat de drona
  + "combined" - nod vizitat de camion si drona
  + "simple" - nod nevizitat inca
* Operatia MakeFly:
  + poate fi realizata pentru nodurile care au predecesor si succesor
  + nodul ri(curent) primeste eticheta "drone"
  + nodurile ri-1 si ri+1 primesc eticheta "combined"
* Operatia PushLeft:
  + poate fi realizata daca nodul parcurs anterior are eticheta "combined"
  + nodul curent primeste eticheta "combined"
  + nodul ri-1 primeste eticheta "truck"
* Operatia PushRight:
  + poate fi realizata daca nodul parcurs ulterior are eticheta "combined"
  + nodul curent primeste eticheta "combined"
  + nodul ri+1 primeste eticheta "truck"
* la fiecare pas, pentru toate operatiile care pot fi efectuate (dupa ce au fost verificate constrangerile), sunt calculate costurile, in functie de distantele care trebuie parcurse de drona si de camion si este efectuata operatia cu costul cel mai mic

1. **Programare dinamica**

Programare dinamica presupune rezolvarea unei probleme prin descompunerea ei în subprobleme si rezolvarea acestora. Aceasta tehnica determina ”valoarea” solutiei pentru fiecare din subprobleme. Mergand de la subprobleme mici la subprobleme din ce in ce mai mari ajungem la solutia optima, la nivelul intregii probleme.

Descrierea solutiei:

* Se construieste un drum in care camionul viziteaza toate nodurile (clientii) = R1. Pornim de la acest drum si incercam sa alocam o parte din nodurile camionului dronei, construind un drum D pentru drona si un drum R pentru camion.
* Notam:
  + R1 = (r0,r1...rn+1), cu r0 = rn+1 = v0 (depou)
  + Operatie = (i,j,k), unde ri = nod start, rj = nod destinatie, rk = nod drona, i < k < j ( k = -1 daca operatia nu include drona)
* Pasi:
  + Se calculeaza costul (timpul) necesar unei operatii T(i,j,k) prin formula:

T(i, j, k) = max{cd (ri , rk) + cd (rk + rj ), SUM(l=i,k-2) c(rl , rl+1) + SUM(l=k+1,j−1) c(rl , rl+1)},

unde cd (ri , rk) + cd (rk + rj ) = costul dronei pentru drumul de la i la k + costul drumului de la k la j,

SUM(l=i,k-2) c(rl , rl+1) + SUM(l=k+1,j−1) c(rl , rl+1) = costul camionului de la a ajungi din i la j adunand toate drumurile intermediare;

Acolo unde drona nu este folosita putem nota

T(i, i + 1, −1) = c(ri , ri+1)

* + Pentru fiecare secventa consecutiva (ri,ri+1…rj) se calculeaza solutia optima care acopera subsecventa:

T(i, j) = min(k=i+1, j−1) T(i, j, k)

Solutia poate fi retinuta prin calcularea unei matrici M = (ri, rj) = costul minim gasit.

**Structuri de date folosite**:

* Operation - structura de date ce contine informatii referitoare la nodul de start (\_start), nodul destinatie (\_end), ultima distanta parcursa de camion (\_lastDrive) sau de camion (\_lastFly) etc. si operatii pe acestea, cum ar fi calcularea costului unui drum de la i la j cu camionul sau cu drona.
* Solution - structura de date ce contine o lista de operatii, o multime de locatii vizitate de drona si camion, costul total al drumului, date referitoare la costul maxim al dronei/ camionului, a timpului maxim de asteptare etc.
* Vec2D - structura de date ce contine coordonatele locatiilor clientilor si operatii cu acestea.

**Rezultate:**

Pentru input-ul:

*/\*The speed of the Truck\*/*

*1.0*

*/\*The speed of the Drone\*/*

*0.5*

*/\*Number of Nodes\*/*

*9*

*/\*The Depot\*/*

*0.15 0.04 depot*

*/\*The Locations (x\_coor y\_coor name)\*/*

*71.0 82.0 loc1*

*88.0 66.0 loc2*

*16.0 67.0 loc3*

*58.0 82.0 loc4*

*61.0 0.0 loc5*

*3.0 80.0 loc6*

*73.0 91.0 loc7*

*21.0 1.0 loc8*

Am obtinut rezultatele:

* **Greedy**

*StartNode Combined Drone TruckPath DronePath*

*0 5 8 [depot, loc5] [depot, loc8, loc5]*

*5 2 3 [loc5, loc2] [loc5, loc3, loc2]*

*2 1 7 [loc2, loc1] [loc2, loc7, loc1]*

*1 4 -1 [loc1, loc4] []*

*4 0 6 [loc4, depot] [loc4, loc6, depot]*

*Total cost : 273.8564933207237*

* **Dynamic Programming**

*StartNode Combined Drone TruckPath DronePath*

*0 8 -1 [depot, loc8] []*

*8 3 5 [loc8, loc3] [loc8, loc5, loc3]*

*3 1 6 [loc3, loc1] [loc3, loc6, loc1]*

*1 4 7 [loc1, loc4] [loc1, loc7, loc4]*

*4 0 2 [loc4, depot] [loc4, loc2, depot]*

*Total cost : 257.72907941820273*