Université de Technologie de Compiègne
TX - Étude expérimentale

Éxploitation de données pour algorithmes de graphes multimodaux

Jing Li & Henri Lefebvre

Printemps 2018

Table des matières

1 Formats de données standards

1.1 Formats NTFS et GTFS

Présentation de chaque fichier

1.2 Réduction des données

Ne garder que le centre de Paris

2 Modèles sans dépendances temporelles

2.1 Représentation

Time independant model principalement pour les réseaux routiers

2.2 Algorithme de Dijkstra

Parler des améliorations qui existent

2.3 Algorithme de Bellman-Ford

Algorithm 1 Algorithme de Bellman-Ford

```
1: procedure BELLMAN-FORD(G = (V, E, w), s)
                                                          ▷ Initialisation
2:     for all sommet v ∈ V do
3:         d[v] ← ∞
4:         π[v] ←NIL
5:     end for
6:     d[s] ← 0
                                                  ▷ Calcul du plus cours chemin
7:     for i de 1 à |V| − 1 do
8:         for all arc (u, v) ∈ E do
9:             if d[v] > d[u] + w(u, v) then
10:                d[v] ← d[u] + w(u, v)
11:                π[v] ← u
12:            end if
13:        end for
14:    end for

15: end procedure
```

Remarque : on peut aider bellman en triant les distances et on s'arrête dès qu'aucun changement n'a été fait lors d'une itération. Dans le pire des cas, cela ne change rien, mais sur nos données c'est assez pertinent...

3 Modèles avec dépendances temporelles

3.1 Représentation

3.1.1 *Condensed Model*

Utiliser le min pour chaque arc, ça donne une borne inférieure (donc pas de résultat exact) mais le graph est petit

3.1.2 *Time-expanded Model*

Les noeuds sont des evenements : $(S_1, S_2, \tau_1, \tau_2)$: Départ de S_1 à τ_1 pour une arrivée à S_2 en τ_2
Pour les transferts on ajoute : stop_1 = from_stop_id, stop_2 = to_stop_id, arrivée(stop_1), arrivée(stop_1) + min_transfer_time pour tous les départ en stop_1 possible (donc complexité mémoire un peu pourri...)

3.2 Algorithme de Bellman-Ford

Grâce au *DataStore* on stock en définitif pour bellman que n_s (= nombre de noeuds dans la composante connexe du stop de départ)

3.2.1 Bellman-Ford réparti

Comme $O(nm)$ et $m >> n$ on pourrait faire du bellman réparti dans la deuxième boucle en mode "map-reduce"
chaque noeud réparti traite un sous ensemble des arcs (map) puis on collecte les resultats (qui sont de la taille de la composante connexe du noeud de départ) et on calcul les min des meilleurs résultats trouvés par chacun des noeuds (reduce).