Algorithme Greedy (sans puis avec pré-traitements)

Octobre 2023

1 Greedy: comment faire mieux

1.1 Notations

- $G(X, A, \bar{A})$: le graphe initial, X l'ensemble des noeuds, A l'ensemble des arcs; $\bar{A}$, l'ensemble des arcs améliorés.
- $dist_{i,j}, dang_{i,j}$ (resp. $\bar{dist}_{i,j}, \bar{dang}_{i,j}$) la distance et le danger de $(i,j) \in A$ (resp. la distance et le danger de $(i,j) \in \bar{A}$). Par extension $dist_z, dang_z$ ou z est un chemin sont la distance et le danger de ce chemin, i.e., la somme des distances des arcs de z et la somme des dangers des arcs de z
- $G(X, A)_{(i,j)}$: le graphe initial dans lequel seul l'arc (i,j) est amélioré
- T ensemble de traces, $card(T) = s$; $start^k$ le point départ de la trace k; end^k, son point d'arrivée, α_k : la préférence associée à la trace, et E^k, l'effort de la trace $k = 1, \ldots, s$.
- $dist^k$, et $dang^k$ représentent respectivement la distance et le danger de la trace k; $E^k = \alpha_k \cdot dist^k + (1 - \alpha^k) \cdot dang^k$, est l'effort de la trace k. Similairement, l'effort associé à un arc pour la trace k est $E^k(i,j) = \alpha_k \cdot dist^k_{i,j} + (1 - \alpha_k) \cdot dang^k_{i,j}$
- $Visi^k$: la visibilité de la trace k, i.e., ensemble d'arcs susceptibles d'être empruntés par l'usager k ; $G(A, V) \bigcap Visi^k$ est le graphe initiale restreint aux arcs qui sont dans la visibilité de la trace k
- $T_{i,j}$, l'ensemble des traces dont la visibilité contient $(i,j) \in A$
- $Couv_{i,j} = \bigcup_{k \in T_{i,j}} Visi^k$: pour chaque arc (i,j), l'ensemble des arcs appartenant à au moins une trace dont la visibilité contient (i,j)
- $\psi^k_{i,j}$, l'amélioration de la trace k lorsque l'arc (i,j) est amélioré ; $\sum_{k \in T_{(i,j)}} \psi^k_{i,j}$ est le score de l'arc (i,j).

1.2 Greedy Basic

Principe

1. On calcule $\mathcal{C}$ l'ensemble des arcs appartenant à au moins une couverture d'une trace.
$$\mathcal{C} = \bigcup Couv_{i,j}$$

2. Pour chaque arc de $\mathcal{C}$ trié dans l'ordre de $\sum_{k \in T_{i,j}} \psi max_{i,j}^k$ on mesure l'impact de son amélioration sur chacune des traces
$$\forall (i,j) \in \mathcal{C} \text{ on calcule } \sum_{k \in T_{i,j}} \psi_{i,j}^k.$$
$\psi_{i,j}^k$ est obtenu en recalculant le chemin de meilleur effort sur $G(X,A)_{(i,j)}$ pour $start^k$ à end^k, avec la préférence α^k, et en le soustrayant à l'effort de la trace

3. S'il y a modification de l'effort pour une trace, on stocke le nouvel effort pour cette trace.

4. $(i,j)^{max}$ est l'arc qui donne la meilleure amélioration parmi les arcs examinés, dont le coût reste dans le budget. C'est l'arc retenu pour être amélioré
$$(i,j)^{max} = argmax_{(i,j)} \sum_{k \in T_{i,j}} \psi_{i,j}^k$$

1.2.1 Algorithme

Algorithm 1 GREEDY simple

$Stop \leftarrow False$
while $Stop == False$ **do**
  for chaque trace k **do**
    for $(i,j) \in Visi^k$ **do**
      `Apply Dijkstra-Forward` en partant de $start^k$
      mettre à jour $\psi_{i,j}^k$
    end for
  end for
  $(i,j)^{max} = argmax_{(i,j)} \sum_{k \in T_{i,j}} \psi_{i,j}^k$ s.t. $dist_{i,j} < Budget$
  if $(i,j)^{max} ==$ `NONE` **then**
    $Stop \leftarrow True$
  end if
  $G \leftarrow G(X,A)_{(i,j)^{max}}$
  budget ← budget - $dist_{i,j^{max}}$
end while

1.2.2 Variante

1. Soit $(i,j)^{max}$ l'arc qui a été choisi pour être amélioré. On ne met à jour que les scores des arcs impactés

$$\forall (i,j) \in Couv_{(i,j)^{max}} \text{ on calcule } \sum_{k \in T_{i,j}} \psi_{i,j}^k . \texttt{(Apply Dijkstra)}$$

   Si il y modification de l'effort pour une trace, on stocke le nouvel effort pour cette trace.

2. On peut, à chaque itération, décider d'améliorer un certain nombre d'arcs correspondant à $\zeta\%$ du budget. (voir 100% du budget)

3. Pour un ensemble d'arcs dont le score est connu, on peut chercher à établir des chemins de poids max et de longueur bornée dans cet ensemble ... (pas très clair sur comment on peut faire)

4. En vue de réduire le nombre de d'arcs à examiner à chaque itération, on peut se limiter aux arcs $(i,j) \in Couv_{(i,j)^{max}}$ adjacents à un aménagement existant.

1.3 Greedy avec pré-traitement

1.3.1 Principe

- Pour chaque trace k de T on calcul un Dijkstra-Forward en partant de $start^k$ et un Dijkstra-Backward à partir de end^k, avec la fonction de score $E^k = \alpha_k \cdot dist^k + (1-\alpha^k) \cdot dang^k$, sur $G(A,V) \bigcap Visi^k$

- Pour chaque arc $(i,j) \in Visi^k$ on stocke l'effort du meilleur chemin de $start^k$ à i avec la préférence α^k (noté $E^k(start^k, i)$) et de j à end^k (noté $E^k(j, End^k)$) avec la préférence α^k

- si l'arc (i,j) est amélioré, l'effort de la trace k sur $G(X,A)_{(i,j)}$ est $E^k_{(i,j)}$:

$$E^k_{(i,j)} = E^k(start^k, i) + E^k(i,j) + E^k(j, End^k)$$

- $\psi^k_{i,j}$, l'amélioration de la trace k lorsque l'arc (i,j) est amélioré, avec E^k l'effort de la trace k

$$\psi^k_{i,j} = Max(0, E^k - E^k_{(i,j)})$$

- $(i,j)^{max}$ est l'arc qui donne la meilleure amélioration parmi les arcs examinés, dont le coût reste dans le budget. C'est l'arc retenu pour être amélioré

$$(i,j)^{max} = argmax_{(i,j)} \sum_{k \in T_{i,j}} \psi^k_{i,j} \text{ s.t. } dist_{i,j} < Budget$$

1.3.2 Algorithme

Algorithm 2 GREEDY avec preprocessing

$Stop \leftarrow False$
for trace $k \in T$ **do**
  $impact^k \leftarrow True$
end for
while $Stop == False$ **do**
  for chaque trace k tq $impact^k == True$ **do**
    $impact^k \leftarrow False$
    `Apply Dijkstra-Forward` en partant de $start^k$
    `Apply Dijkstra-Backward` à partir de end^k
    for $(i,j) \in Visi^k$ **do**
      Mettre à jour : $E^k(start^k, i)$ et $E^k(j, End^k))$
      $\psi^k_{i,j} = Max(0, E^k - (E^k(start^k, i) + E^k(i,j) + E^k(j, End^k)))$
    end for
  end for
  $(i,j)^{max} = argmax_{(i,j)} \sum_{k \in T_{i,j}} \psi^k_{i,j}$ s.t. $dist_{i,j} < Budget$
  if $(i,j)^{max} ==$ `NONE` **then**
    $Stop \leftarrow True$
  else
    for trace $k \in T_{i,j^{(max)}}$ **do**
      $impact^k \leftarrow True$
    end for
    $G \leftarrow G(X,A)_{(i,j)^{max}}$
    budget $\leftarrow$ budget - $dist_{i,j^{max}}$
  end if
end while