Enjeux éthiques d'un algorithme d'aide à la décision

Ou quand l'optimisation d'intérêts individuels entre en conflit avec des intérêts collectifs

Christine Solnon (Equipe Emeraude, CITI, INSA Lyon / Inria)

Travail en collaboration avec Odile Bellenguez, Nadia Brauner et Alexis Tsoukias

Algorithmes et Société, Grenoble

29 mai 2026

Pourquoi vous parler d'enjeux éthiques d'un algorithme d'aide à la décision ?

- Concevoir une nouvelle application informatique peut être facile et rapide
  ... et souvent même très plaisant !
- Mais les impacts sur la société peuvent être radicaux et rapides

Illustration : Waze

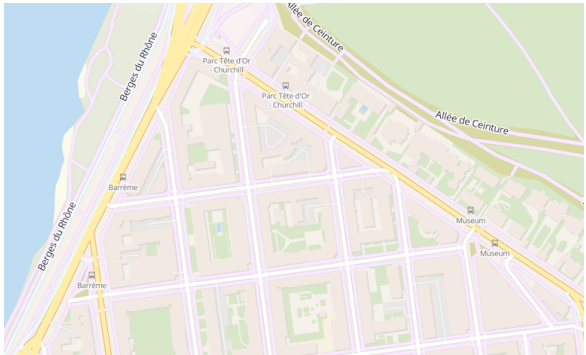
- Basé sur de très beaux algorithmes
- Devenu (relativement) incontournable
- Modifie notre façon de nous déplacer... et plein d'autres choses dont on n'a pas forcément conscience

Enjeux éthiques d'un algorithme d'aide à la décision

Ou quand l'optimisation d'intérêts individuels entre en conflit avec des intérêts collectifs

Comment passer d'un plan de ville à un graphe orienté et pondéré ?

- Associer un sommet à chaque intersection
- Associer un arc à chaque tronçon de route
  ⇝ Orientation de l'arc = Sens de circulation
- Associer un coût à chaque arc
  ⇝ Longueur (ou durée) du tronçon



Comment passer d'un plan de ville à un graphe orienté et pondéré ?

- Associer un sommet à chaque intersection
- Associer un arc à chaque tronçon de route
  ⇝ Orientation de l'arc = Sens de circulation
- Associer un coût à chaque arc
  ⇝ Longueur (ou durée) du tronçon

Comment passer d'un plan de ville à un graphe orienté et pondéré ?

- Associer un sommet à chaque intersection
- Associer un arc à chaque tronçon de route
  ⇝ Orientation de l'arc = Sens de circulation
- Associer un coût à chaque arc
  ⇝ Longueur (ou durée) du tronçon

Comment passer d'un plan de ville à un graphe orienté et pondéré ?

- Associer un sommet à chaque intersection
- Associer un arc à chaque tronçon de route
  ⇝ Orientation de l'arc = Sens de circulation
- Associer un coût à chaque arc
  ⇝ Longueur (ou durée) du tronçon

Comment passer d'un plan de ville à un graphe orienté et pondéré ?

- Associer un sommet à chaque intersection
- Associer un arc à chaque tronçon de route
  ⇝ Orientation de l'arc = Sens de circulation
- Associer un coût à chaque arc
  ⇝ Longueur (ou durée) du tronçon

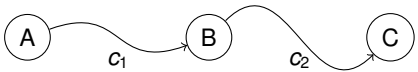
Meilleur itinéraire dans le plan = Plus court chemin dans le graphe

⇝ Séquence de sommets reliés deux à deux par des arcs et minimisant la somme des coûts des arcs

Programmation dynamique pour le calcul de plus courts chemins

Propriété d'optimalité des sous-chemins :

Tout préfixe d'un plus court chemin est un plus court chemin



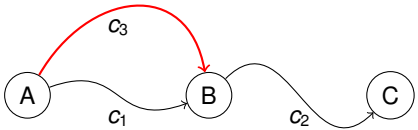
Programmation dynamique pour le calcul de plus courts chemins

Propriété d'optimalité des sous-chemins :

Tout préfixe d'un plus court chemin est un plus court chemin

$\rightsquigarrow$ Démonstration par l'absurde :

$$c_3 < c_1 \Rightarrow c_3 + c_2 < c_1 + c_2$$



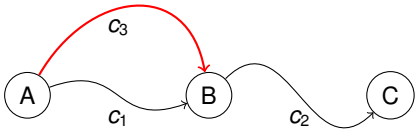
Programmation dynamique pour le calcul de plus courts chemins

Propriété d'optimalité des sous-chemins :

Tout préfixe d'un plus court chemin est un plus court chemin

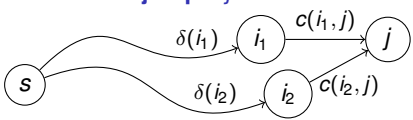
$\rightsquigarrow$ Démonstration par l'absurde :

$$c_3 < c_1 \Rightarrow c_3 + c_2 < c_1 + c_2$$



Equations de Bellman pour définir la longueur $\delta(j)$ du plus court chemin de s jusque j :

- Si $j = s$: $\delta(j) = 0$
- Sinon : $\delta(j) = min_{i \in pred(j)} \delta(i) + c(i,j)$



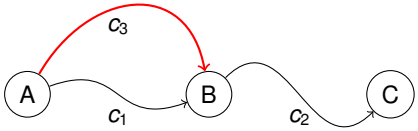
Programmation dynamique pour le calcul de plus courts chemins

Propriété d'optimalité des sous-chemins :

Tout préfixe d'un plus court chemin est un plus court chemin

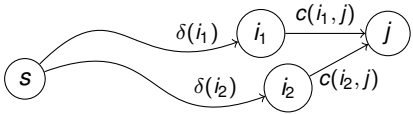
$\rightsquigarrow$ Démonstration par l'absurde :

$$c_3 < c_1 \Rightarrow c_3 + c_2 < c_1 + c_2$$



Equations de Bellman pour définir la longueur $\delta(j)$ du plus court chemin de s jusque j :

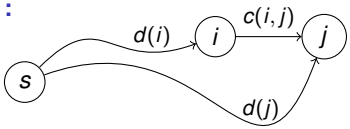
- Si $j = s$: $\delta(j) = 0$
- Sinon : $\delta(j) = min_{i \in pred(j)} \delta(i) + c(i,j)$



Principe commun aux algorithmes classiques de plus courts chemins :

Calculer δ en grignotant une borne supérieure d par relâchement d'arcs

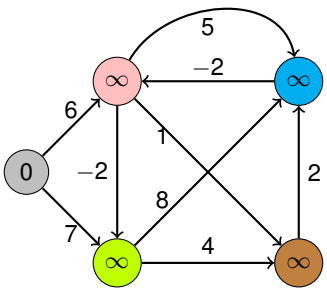
$\rightsquigarrow$ Relâcher l'arc (i,j) :
Si $d(j) > d(i) + c(ij)$ alors $d(j) \leftarrow d(i) + c(i,j)$



Un premier algorithme très simple (Bellman-Ford, 1955-1958)

Algorithme pour calculer la longueur de tous les plus courts chemins au départ de s :

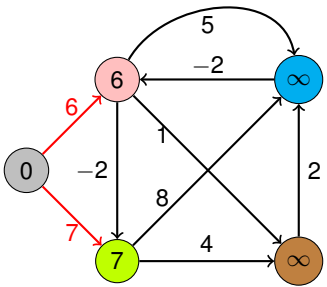
- Initialiser $d(i)$ à ∞ pour tout sommet $i \neq s$, et initialiser $d(s)$ à 0
- Relâcher tous les arcs jusqu'à convergence



Un premier algorithme très simple (Bellman-Ford, 1955-1958)

Algorithme pour calculer la longueur de tous les plus courts chemins au départ de s :

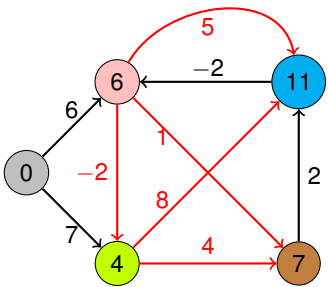
- Initialiser $d(i)$ à ∞ pour tout sommet $i \neq s$, et initialiser $d(s)$ à 0
- Relâcher tous les arcs jusqu'à convergence



Un premier algorithme très simple (Bellman-Ford, 1955-1958)

Algorithme pour calculer la longueur de tous les plus courts chemins au départ de s :

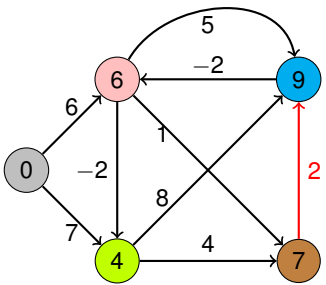
- Initialiser $d(i)$ à ∞ pour tout sommet $i \neq s$, et initialiser $d(s)$ à 0
- Relâcher tous les arcs jusqu'à convergence



Un premier algorithme très simple (Bellman-Ford, 1955-1958)

Algorithme pour calculer la longueur de tous les plus courts chemins au départ de s :

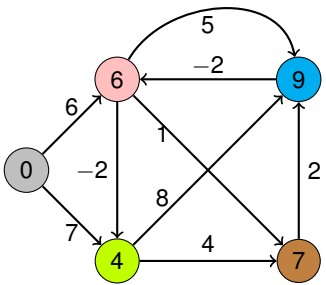
- Initialiser $d(i)$ à ∞ pour tout sommet $i \neq s$, et initialiser $d(s)$ à 0
- Relâcher tous les arcs jusqu'à convergence



Un premier algorithme très simple (Bellman-Ford, 1955-1958)

Algorithme pour calculer la longueur de tous les plus courts chemins au départ de s :

- Initialiser $d(i)$ à ∞ pour tout sommet $i \neq s$, et initialiser $d(s)$ à 0
- Relâcher tous les arcs jusqu'à convergence

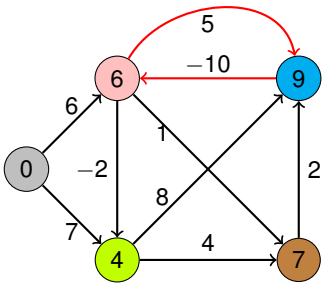


Est-on bien certain de converger un jour ?

Un premier algorithme très simple (Bellman-Ford, 1955-1958)

Algorithme pour calculer la longueur de tous les plus courts chemins au départ de s :

- Initialiser $d(i)$ à ∞ pour tout sommet $i \neq s$, et initialiser $d(s)$ à 0
- Relâcher tous les arcs jusqu'à convergence



Est-on bien certain de converger un jour ?

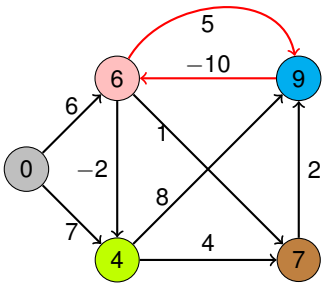
- Pas dans le cas où le graphe contient des circuits de coût total négatif !
  $\rightsquigarrow$ Supposons que le graphe n'a pas de circuit de coût total négatif

Au bout de combien d'itérations converge-t-on ?

Un premier algorithme très simple (Bellman-Ford, 1955-1958)

Algorithme pour calculer la longueur de tous les plus courts chemins au départ de s :

- Initialiser $d(i)$ à ∞ pour tout sommet $i \neq s$, et initialiser $d(s)$ à 0
- Relâcher tous les arcs jusqu'à convergence



Est-on bien certain de converger un jour ?

- Pas dans le cas où le graphe contient des circuits de coût total négatif !
  $\rightsquigarrow$ Supposons que le graphe n'a pas de circuit de coût total négatif

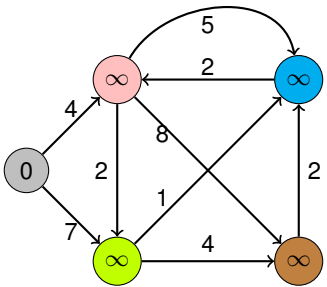
Au bout de combien d'itérations converge-t-on ?

Maximum $n - 1$ itérations (si n sommets et p arcs)
$\rightsquigarrow$ Complexité en $\mathcal{O}(n * p)$

Peut-on faire mieux que Bellman-Ford ?

Algorithme de (Dijkstra, 1959) dans le cas où tous les coûts sont positifs :

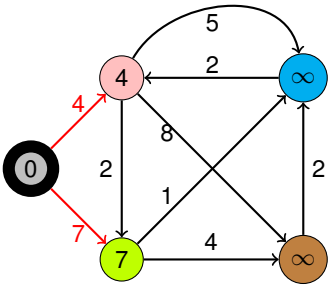
- Initialiser $d(i)$ à ∞ pour tout sommet $i \neq s$, et initialiser $d(s)$ à 0
- Tant qu'il existe un sommet non traité :
  - Relâcher les arcs au départ du sommet non traité ayant la plus petite valeur de d
  - Marquer ce sommet comme "traité"



Peut-on faire mieux que Bellman-Ford ?

Algorithme de (Dijkstra, 1959) dans le cas où tous les coûts sont positifs :

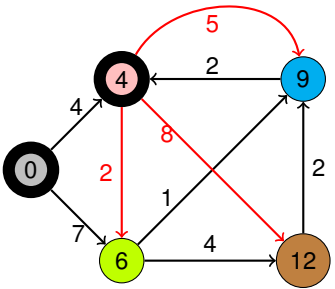
- Initialiser $d(i)$ à ∞ pour tout sommet $i \neq s$, et initialiser $d(s)$ à 0
- Tant qu'il existe un sommet non traité :
  - Relâcher les arcs au départ du sommet non traité ayant la plus petite valeur de d
  - Marquer ce sommet comme "traité"



Peut-on faire mieux que Bellman-Ford ?

Algorithme de (Dijkstra, 1959) dans le cas où tous les coûts sont positifs :

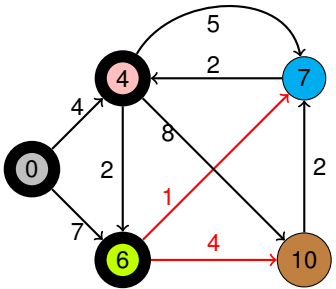
- Initialiser $d(i)$ à ∞ pour tout sommet $i \neq s$, et initialiser $d(s)$ à 0
- Tant qu'il existe un sommet non traité :
  - Relâcher les arcs au départ du sommet non traité ayant la plus petite valeur de d
  - Marquer ce sommet comme "traité"



Peut-on faire mieux que Bellman-Ford ?

Algorithme de (Dijkstra, 1959) dans le cas où tous les coûts sont positifs :

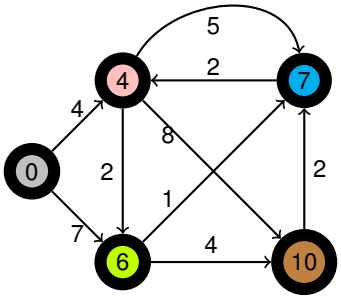
- Initialiser $d(i)$ à ∞ pour tout sommet $i \neq s$, et initialiser $d(s)$ à 0
- Tant qu'il existe un sommet non traité :
  - Relâcher les arcs au départ du sommet non traité ayant la plus petite valeur de d
  - Marquer ce sommet comme "traité"



Peut-on faire mieux que Bellman-Ford ?

Algorithme de (Dijkstra, 1959) dans le cas où tous les coûts sont positifs :

- Initialiser $d(i)$ à ∞ pour tout sommet $i \neq s$, et initialiser $d(s)$ à 0
- Tant qu'il existe un sommet non traité :
  - Relâcher les arcs au départ du sommet non traité ayant la plus petite valeur de d
  - Marquer ce sommet comme "traité"



Complexité en $\mathcal{O}(p + n \log n)$ s'il y a n sommets et p arcs

Peut-on faire mieux ?

- Dijkstra trouve tous les plus courts chemins au départ de s
  Mais il trie aussi les sommets, du plus proche de s au plus éloigné
  $\rightsquigarrow$ Impossible de faire mieux si on veut trier les sommets
- On peut faire mieux si on ne trie pas les sommets (Duan et al, 2025) !
- On peut aussi faire mieux si on connait la destination

Au delà de Dijkstra

Algorithme A* dans le cas où la destination f est connue (Hart, Nilsson, Raphael 1968) :

- Définir une heuristique h telle que $h(j)$ = borne inférieure du coût pour aller de j jusque f
- A chaque itération, relâcher les arcs partant du sommet j tel que $d(j) + h(j)$ soit minimal

$\rightsquigarrow$ h donne un cap à l'algorithme et permet généralement de réduire le nombre d'arcs relâchés

Améliorations possibles :

- Recherche bi-directionnelle : lancer A* en parallèle depuis s et depuis f
- Pré-calcul de distances entre *landmarks* pour améliorer l'heuristique h
- ...

Voir par exemple (Wagner & Willhalm, 2007)

Intégration des conditions de circulation

On ne roule pas à la même vitesse en fonction de l'heure de passage

La fonction de coût doit dépendre de l'heure de passage

$\rightsquigarrow$ $c(i, j, \mathbf{t})$ = durée pour aller de i à j **quand on part de i à l'instant t**

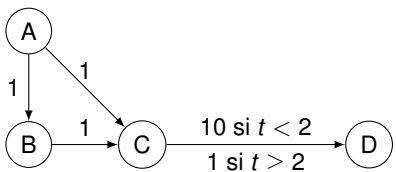
Peut-on utiliser A* (ou Dijkstra, ou Bellman-Ford) sachant que $c(i, j, t)$ est toujours positif ?

Pour répondre à cette question, commençons par étudier la propriété d'optimalité des sous-chemins...

Propriété d'optimalité des sous-chemins

Un sous-chemin d'un chemin optimal est-il toujours optimal ?

Réponse sur un exemple :



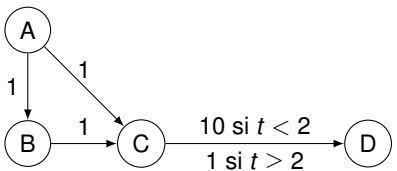
En partant à $t = 0$

- (A, B, C, D) minimise l'heure d'arrivée sur D
- Mais (A, B, C) ne minimise pas l'heure d'arrivée sur C
  $\rightsquigarrow$ Ce sous-chemin n'est pas optimal !

Propriété d'optimalité des sous-chemins

Un sous-chemin d'un chemin optimal est-il toujours optimal ?

Réponse sur un exemple :



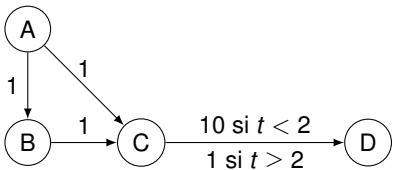
En partant à $t = 0$

- (A, B, C, D) minimise l'heure d'arrivée sur D
- Mais (A, B, C) ne minimise pas l'heure d'arrivée sur C
  $\rightsquigarrow$ Ce sous-chemin n'est pas optimal !

Propriété d'optimalité des sous-chemins

Un sous-chemin d'un chemin optimal est-il toujours optimal ?

Réponse sur un exemple :



En partant à $t = 0$

- (A, B, C, D) minimise l'heure d'arrivée sur D
- Mais (A, B, C) ne minimise pas l'heure d'arrivée sur C
  $\rightsquigarrow$ Ce sous-chemin n'est pas optimal !

Que peut-on faire ?

- Solution 1 : Définir un nouveau graphe satisfaisant la propriété d'optimalité des sous-chemins
- Solution 2 : Contraindre la fonction de coût pour restaurer la propriété d'optimalité des sous-chemins

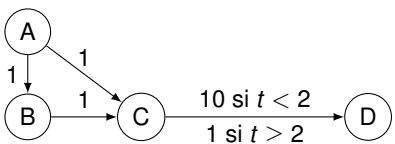
Solution 1 : Définition d'un graphe états-transitions

Graphe états-transitions pour calculer le meilleur *st*-chemin dans le graphe $G = (S, A)$ en partant à t_0 :

- États = couples $(i, t) \in S \times \mathbb{R}$
- Transition de (i, t_i) vers (j, t_j) si $(i, j) \in A$ et $t_j = t_i + c(i, j, t_i)$
- Coût de la transition $((i, t_i), (j, t_j)) = t_j - t_i = c(i, j, t_i)$
- Solution = Plus court chemin de (s, t_0) jusqu'à un état (i, t_i) tel que $i = f$

$\rightsquigarrow$ Construction dynamique du graphe : on crée les états au moment du relâchement des arcs

Exemple :



On a maintenant un problème "classique" de recherche de plus court chemin...

... mais quelle est la taille du graphe états-transitions par rapport au graphe initial ?

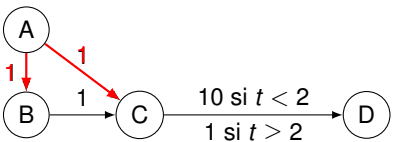
Solution 1 : Définition d'un graphe états-transitions

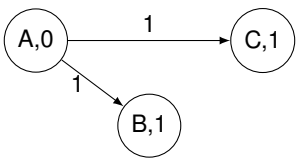
Graphe états-transitions pour calculer le meilleur *st*-chemin dans le graphe $G = (S, A)$ en partant à t_0 :

- États = couples $(i, t) \in S \times \mathbb{R}$
- Transition de (i, t_i) vers (j, t_j) si $(i, j) \in A$ et $t_j = t_i + c(i, j, t_i)$
- Coût de la transition $((i, t_i), (j, t_j)) = t_j - t_i = c(i, j, t_i)$
- Solution = Plus court chemin de (s, t_0) jusqu'à un état (i, t_i) tel que $i = f$

$\rightsquigarrow$ Construction dynamique du graphe : on crée les états au moment du relâchement des arcs

Exemple :





On a maintenant un problème "classique" de recherche de plus court chemin...

... mais quelle est la taille du graphe états-transitions par rapport au graphe initial ?

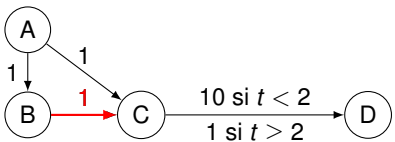
Solution 1 : Définition d'un graphe états-transitions

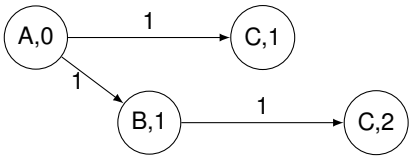
Graphe états-transitions pour calculer le meilleur *st*-chemin dans le graphe $G = (S, A)$ en partant à t_0 :

- États = couples $(i, t) \in S \times \mathbb{R}$
- Transition de (i, t_i) vers (j, t_j) si $(i, j) \in A$ et $t_j = t_i + c(i, j, t_i)$
- Coût de la transition $((i, t_i), (j, t_j)) = t_j - t_i = c(i, j, t_i)$
- Solution = Plus court chemin de (s, t_0) jusqu'à un état (i, t_i) tel que $i = f$

$\rightsquigarrow$ Construction dynamique du graphe : on crée les états au moment du relâchement des arcs

Exemple :





On a maintenant un problème "classique" de recherche de plus court chemin...

... mais quelle est la taille du graphe états-transitions par rapport au graphe initial ?

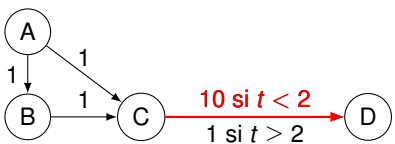
Solution 1 : Définition d'un graphe états-transitions

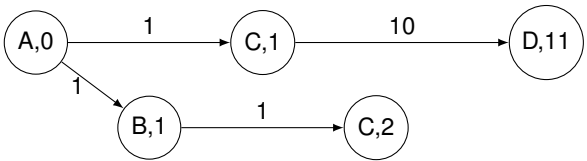
Graphe états-transitions pour calculer le meilleur *st*-chemin dans le graphe $G = (S, A)$ en partant à t_0 :

- États = couples $(i, t) \in S \times \mathbb{R}$
- Transition de (i, t_i) vers (j, t_j) si $(i, j) \in A$ et $t_j = t_i + c(i, j, t_i)$
- Coût de la transition $((i, t_i), (j, t_j)) = t_j - t_i = c(i, j, t_i)$
- Solution = Plus court chemin de (s, t_0) jusqu'à un état (i, t_i) tel que $i = f$

$\rightsquigarrow$ Construction dynamique du graphe : on crée les états au moment du relâchement des arcs

Exemple :





On a maintenant un problème "classique" de recherche de plus court chemin...

... mais quelle est la taille du graphe états-transitions par rapport au graphe initial ?

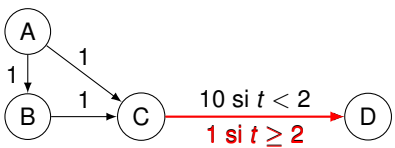
Solution 1 : Définition d'un graphe états-transitions

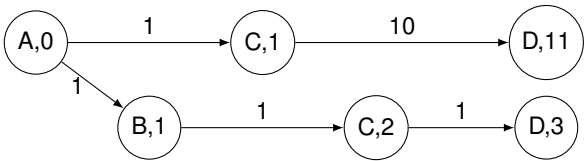
Graphe états-transitions pour calculer le meilleur *st*-chemin dans le graphe $G = (S, A)$ en partant à t_0 :

- États = couples $(i, t) \in S \times \mathbb{R}$
- Transition de (i, t_i) vers (j, t_j) si $(i, j) \in A$ et $t_j = t_i + c(i, j, t_i)$
- Coût de la transition $((i, t_i), (j, t_j)) = t_j - t_i = c(i, j, t_i)$
- Solution = Plus court chemin de (s, t_0) jusqu'à un état (i, t_i) tel que $i = f$

$\rightsquigarrow$ Construction dynamique du graphe : on crée les états au moment du relâchement des arcs

Exemple :





On a maintenant un problème "classique" de recherche de plus court chemin...

... mais quelle est la taille du graphe états-transitions par rapport au graphe initial ?

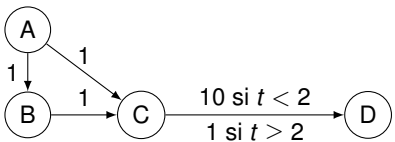
Solution 1 : Définition d'un graphe états-transitions

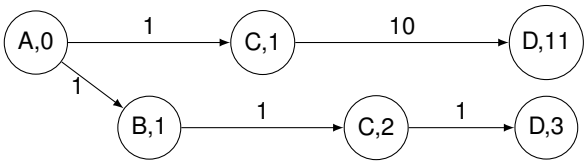
Graphe états-transitions pour calculer le meilleur *st*-chemin dans le graphe $G = (S, A)$ en partant à t_0 :

- États = couples $(i, t) \in S \times \mathbb{R}$
- Transition de (i, t_i) vers (j, t_j) si $(i, j) \in A$ et $t_j = t_i + c(i, j, t_i)$
- Coût de la transition $((i, t_i), (j, t_j)) = t_j - t_i = c(i, j, t_i)$
- Solution = Plus court chemin de (s, t_0) jusqu'à un état (i, t_i) tel que $i = f$

$\rightsquigarrow$ Construction dynamique du graphe : on crée les états au moment du relâchement des arcs

Exemple :

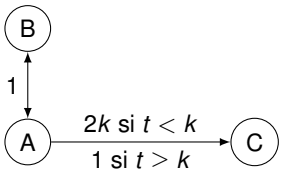




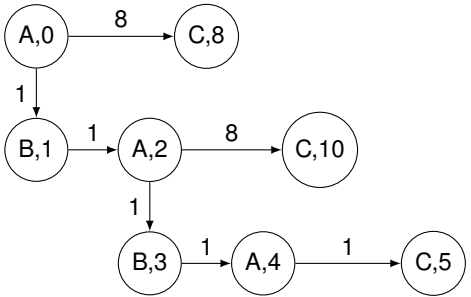
On a maintenant un problème "classique" de recherche de plus court chemin...

... mais quelle est la taille du graphe états-transitions par rapport au graphe initial ?

Cas où le graphe états-transitions a une taille arbitrairement grande



- Si $k = 4$, on a 8 états



- De façon générale, on a $\mathcal{O}(k)$ états
- Et si les coûts sont dans $\mathbb{R}$, on peut avoir une infinité d'états...

Le problème de recherche de plus courts chemins dépendant du temps est NP-difficile !

Voir (Zeitz, 2023)

Solution 2 : Restaurer la propriété d'optimalité des sous-chemins

Propriété FIFO pour une fonction de coût c :

$\forall t_1, t_2, t_1 < t_2 \Rightarrow t_1 + c(i, j, t_1) < t_2 + c(i, j, t_2)$
$\rightsquigarrow$ On ne peut pas arriver plus tôt sur j en partant plus tard de i

La propriété d'optimalité des sous-chemins est-elle vérifiée dans ce cas ?

Solution 2 : Restaurer la propriété d'optimalité des sous-chemins

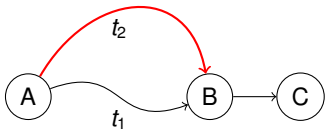
Propriété FIFO pour une fonction de coût c :

$\forall t_1, t_2, t_1 < t_2 \Rightarrow t_1 + c(i, j, t_1) < t_2 + c(i, j, t_2)$
$\rightsquigarrow$ On ne peut pas arriver plus tôt sur j en partant plus tard de i

La propriété d'optimalité des sous-chemins est-elle vérifiée dans ce cas ?

Oui, car $t_1 < t_2 \Rightarrow t_1 + c(B, C, t_1) < t_2 + c(B, C, t_2)$

- Les équations de Bellman sont de nouveau valides
- On peut résoudre le problème en relâchant les arcs :
  Si $d(j) > d(i) + c(i, j, \mathbf{d(i)})$ alors $d(j) \leftarrow d(i) + c(i, j, \mathbf{d(i)})$
  $\rightsquigarrow$ Dijkstra et A* peuvent être utilisés !
- La recherche bidirectionnelle peut aussi être adaptée
  (voir Nannicini et al, 2010)



Solution 2 : Restaurer la propriété d'optimalité des sous-chemins

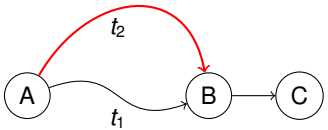
Propriété FIFO pour une fonction de coût c :

$\forall t_1, t_2, t_1 < t_2 \Rightarrow t_1 + c(i, j, t_1) < t_2 + c(i, j, t_2)$
$\rightsquigarrow$ On ne peut pas arriver plus tôt sur j en partant plus tard de i

La propriété d'optimalité des sous-chemins est-elle vérifiée dans ce cas ?

Oui, car $t_1 < t_2 \Rightarrow t_1 + c(B, C, t_1) < t_2 + c(B, C, t_2)$

- Les équations de Bellman sont de nouveau valides
- On peut résoudre le problème en relâchant les arcs :
  Si $d(j) > d(i) + c(i, j, \mathbf{d(i)})$ alors $d(j) \leftarrow d(i) + c(i, j, \mathbf{d(i)})$
  $\rightsquigarrow$ Dijkstra et A* peuvent être utilisés !
- La recherche bidirectionnelle peut aussi être adaptée
  (voir Nannicini et al, 2010)



Et en pratique, les fonctions de coût associées à des tronçons de route sont-elles FIFO ?

- Pas nécessairement si la fonction est constante par morceaux...
- Mais on a un algorithme efficace pour la rendre FIFO dans ce cas !

En quoi est-ce beau ?

Un problème concret que tout le monde connait, mais aussi l'occasion de :

- Être confronté à un cas où la propriété d'optimalité des sous-chemins n'est pas vérifiée
- Reformuler un problème sous la forme de la recherche d'un chemin dans un graphe états-transition
- Constater que la frontière entre la classe $\mathcal{P}$ et celle des problèmes $\mathcal{NP}$-difficiles est étroite

En quoi est-ce beau ?

Un problème concret que tout le monde connait, mais aussi l'occasion de :

- Être confronté à un cas où la propriété d'optimalité des sous-chemins n'est pas vérifiée
- Reformuler un problème sous la forme de la recherche d'un chemin dans un graphe états-transition
- Constater que la frontière entre la classe $\mathcal{P}$ et celle des problèmes $\mathcal{NP}$-difficiles est étroite

Autres discussions techniques possibles :

- Comment définir la fonction de coût c ?
  $\rightsquigarrow$ Quelles données utiliser ? Quel modèle apprendre ?
- Comment adapter dynamiquement les chemins aux conditions de trafic quand elles sont différentes des prévisions initiales ?
- ...

Qu'est-ce qu'un bon algorithme ?

La réponse dépend de à qui l'on pose la question !

- L'informaticien qui conçoit et développe l'application
- L'entreprise qui distribue l'application et récolte les données
- L'automobiliste qui est guidé par l'application
- Les autres usagers sur l'itinéraire ou les riverains des routes empruntées
- Les collectivités locales qui définissent les politiques publiques de déplacement
- Les défenseurs de l'environnement

⇝ **Identifier les différentes parties prenantes, et leurs intérêts**

La réponse dépend aussi de comment on définit ce qui est bon

- Quelles sont les conséquences possibles pour chaque partie prenante ?
- Quelles sont les valeurs et principes qui guident la décision ?
  ⇝ Optimiser ses déplacements *versus* Prendre son temps

⇝ **Une application informatique n'est jamais neutre (Ellul, 1977)**

Qu'est-ce qu'un bon algorithme ?

La réponse dépend de à qui l'on pose la question !

- L'informaticien qui conçoit et développe l'application
- L'entreprise qui distribue l'application et récolte les données
- L'automobiliste qui est guidé par l'application
- Les autres usagers sur l'itinéraire ou les riverains des routes empruntées
- Les collectivités locales qui définissent les politiques publiques de déplacement
- Les défenseurs de l'environnement

⇝ **Identifier les différentes parties prenantes, et leurs intérêts**

La réponse dépend aussi de comment on définit ce qui est bon

- Quelles sont les conséquences possibles pour chaque partie prenante ?
- Quelles sont les valeurs et principes qui guident la décision ?
  ⇝ Optimiser ses déplacements *versus* Prendre son temps

⇝ **Une application informatique n'est jamais neutre (Ellul, 1977)**

Questionner les besoins et les alternatives possibles

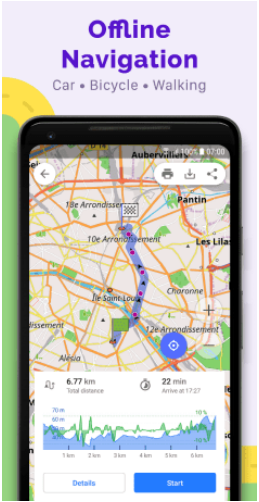
⇝ Besoin 1 : Rechercher un itinéraire

Alternatives possibles :

- Carte papier + cerveau humain
- Carte numérique + algorithme (sans réseau)
  ⇝ Voir par exemple OpenStreetMap
- GPS + carte numérique + algorithme (sans réseau)
  ⇝ Voir par exemple OsmAnd

Ces différentes alternatives sont-elles conviviales au sens de (Illich, 1973)

- Élargissement du rayon d'action personnel ?
- Dégradation de l'autonomie personnelle ?
- Créations de relations de pouvoir ?



Questionner les besoins et les alternatives possibles

⇝ Besoin 2 : Rechercher l'itinéraire le plus rapide en fonction des conditions de circulation

Alternatives possibles pour accéder aux conditions de circulation :

- Services publics (Exemple : plateforme data de la métropole de Lyon)

‹ Retour

Etat du trafic de la Métropole de Lyon - disponibilités temps réel

Métropole de Lyon

Informations — Licence Mobilités

Données — 3037 lignes

Télécharger — XML, PDF, SHAPE-ZIP, KML ...

API — WMS, V

DESCRIPTION

L'état du trafic temps réel est un objet linéaire qui représente la densité de trafic mesurée en temps réel (vert = fluide, orange = dense, rouge = chargé, noir = route coupée, gris = donnée indisponible). L'état du trafic est rafraîchi toutes les minutes.

Ces données sont issues d'informations remontées par des capteurs de mesures de trafic et sont consolidées dans le système Criter (régulation automatisé du trafic routier) de la Direction de la Voirie du Grand Lyon.

Questionner les besoins et les outils possibles

Besoin 3 : Être informé des radars et contrôles de police

Alternatives possibles :

- Appels de phare

Est-ce bien de signaler les radars et contrôles de police ?

On peut imaginer des arguments en faveur du oui

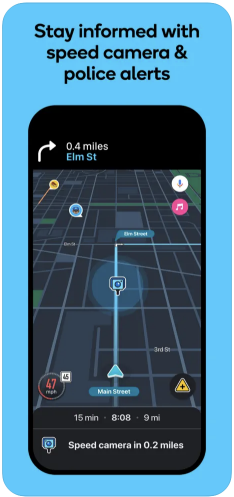
- Incitation à la prudence : Alertes considérées utiles par la sécurité routière
- Liberté de communication garantie par la Constitution

Mais aussi des arguments en faveur du non

- Distraction des automobilistes
- Nuisance aux opérations des forces de l'ordre
  ⇝ Les automobilistes peuvent être des malfaiteurs ou des terroristes...

Décret de 2021 :

Les signalements de contrôles de police doivent pouvoir être bloqués quand l'état en fait la demande



Questionner les devoirs et interdits

Peut-on imaginer d'autres cas d'atteinte à la sécurité individuelle ou collective ?

- Itinéraires dangereux du fait de données obsolètes ?
- Délestages créant des embouteillages gênant l'accès aux urgences ?
- ...

Qui est responsable : Le développeur ? L'utilisateur ? L'entreprise ? ...

- Peut-on reproduire la situation ayant mené à la décision ?
- Peut-on expliquer la décision ?

⇝ Les procès fictifs de l'IA montrent qu'il est complexe de répondre !

Questionner les contradictions entre intérêts individuels et intérêt collectif

Algorithmes et régulation des territoires
(D. Cardon et M. Crépel, 2019)

En termes de politique publique, il est frappant de constater que les enjeux tenant aux calculs de la ville font apparaître un changement de paradigme qui voit la régulation de la ville se déplacer d'une logique de choix collectifs orientant les usages de la ville à une optimisation utilitariste de la satisfaction des usagers des plateformes.

Questionner le modèle économique

Principale source de revenu : Publicités géolocalisées et personnalisées

- Nombreuses données collectées (requêtes, traces, signalements, ...) pour cibler les publicités
- Comment vérifier l'absence de biais ou de manipulations ?
  ⇝ Équipe Projet Inria Regalia
- Est-ce moralement acceptable qu'une entreprise en sache autant sur nous ?
  ⇝ Accepterait-on cela de l'état ? de nos proches ?

Création d'une communauté

Le signalement d'événements donne des points et des récompenses (avatars)
⇝ Gamification de l'application pour créer de l'engagement

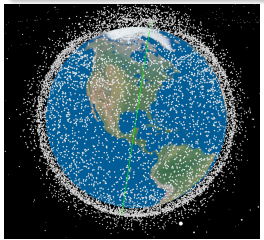
Création d'une situation de monopole

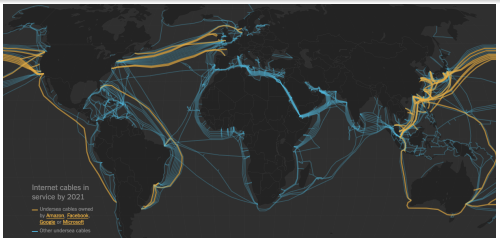
Plus la plateforme est utilisée, meilleur est le service
Waze is a live map that harnesses the local knowledge of tens of millions of drivers around the world

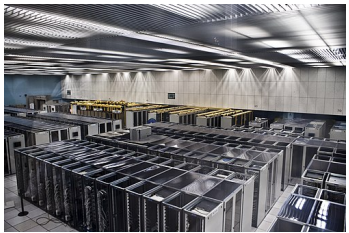
Questionner les impacts environnementaux via la taxonomie de (Horner et al, 2016)

Type	Scope	Effect
1st order	Direct	Manufacturing impact
		Use impact
		End of life impact

- Manufacturing of GPS, smartphones, antennas, servers, ...
- Use of GPS, smartphones, antennas, servers, ...
- End of life of GPS, smartphones, antennas, servers, ...







Questionner les impacts environnementaux via la taxonomie de (Horner et al, 2016)

Type	Scope	Effect
1st order	Direct	Manufacturing impact
		Use impact
		End of life impact
2nd order	Indirect: Unique service	Optimisation
		Substitution

- Optimisation: Travel times and costs are decreased thanks to the routing system
- Substitution: Replacement of paper-based maps

Questionner les impacts environnementaux via la taxonomie de (Horner et al, 2016)

Type	Scope	Effect
1st order	Direct	Manufacturing impact
		Use impact
		End of life impact
2nd order	Indirect: Unique service	Optimisation
		Substitution
3rd order		Direct rebound

- The number of travels increases because travel times and costs have decreased

Questionner les impacts environnementaux via la taxonomie de (Horner et al, 2016)

Type	Scope	Effect
1st order	Direct	Manufacturing impact
		Use impact
		End of life impact
2nd order	Indirect: Unique service	Optimisation
		Substitution
3rd order		Direct rebound
	Indirect: Complementary services	Indirect rebound

- Saved time and costs are re-invested in other activities that generate new impacts

Questionner les impacts environnementaux via la taxonomie de (Horner et al, 2016)

Type	Scope	Effect
1st order	Direct	Manufacturing impact
		Use impact
		End of life impact
2nd order	Indirect: Unique service	Optimisation
		Substitution
3rd order		Direct rebound
	Indirect: Complementary services	Indirect rebound
	Indirect: Economy	Structural changes

- The system enables autonomous vehicles and causes growth of intelligent transportation system manufacturing

Questionner les impacts environnementaux via la taxonomie de (Horner et al, 2016)

Type	Scope	Effect
1st order	Direct	Manufacturing impact
		Use impact
		End of life impact
2nd order	Indirect: Unique service	Optimisation
		Substitution
3rd order		Direct rebound
	Indirect: Complementary services	Indirect rebound
	Indirect: Economy	Structural changes
	Indirect: Society	Systemic changes

- Cities modify traffic plans to increase travel times of routes that cross residential districts

Enjeux éthiques d'un algorithme d'aide à la décision

Ou quand l'optimisation d'intérêts individuels entre en conflit avec des intérêts collectifs

Le mot de la fin par Bernard Stiegler...

Le numérique est un pharmakon, un poison et un remède (Stiegler, 2015)

Le numérique est un pharmakon ayant d'extraordinaires potentialités de libération et d'ouverture, mais à condition seulement de définir des règles de conception et de fonctionnement des plateformes.

Le numérique est un pharmakon qui, tel un médicament, suppose une thérapeutique, laquelle ne peut pas être déléguée au fabriquant du « médicament ». Il en va ainsi avec les narcotrafiquants, et rien n'est plus destructeur. La thérapeutique est l'affaire de la politique, c'est-à-dire de tous les citoyens. Et ce sont les établissements d'enseignement et de recherche qui doivent le permettre – par la généralisation de ce que nous appelons la recherche contributive.

La technologie numérique est une forme de l'écriture, une écriture qui se produit à la vitesse de la lumière, par le truchement de machines auxquelles on délègue des processus de lecture et d'écriture organisés et contrôlés par un secteur industriel planétaire, fondé sur des entreprises de taille mondiale qui existent depuis très peu de temps. L'écriture et la lecture numériques constituent le nouveau milieu des savoirs : l'astrophysique, la nano-physique, la biologie, la géographie, l'histoire, les mathématiques, la linguistique, les sciences du sport même. Ainsi, tout, absolument tout, est en train de devenir numérique. On assiste à une mutation totale du savoir, qui concerne aussi les savoir-faire et les savoir-vivre. C'est la vie quotidienne qui s'en trouve d'abord bouleversée, dans toutes les dimensions de l'existence.