Le guide du routard temporel

Quentin Bramas, Jean-Romain Luttringer, Sébastien Tixeuil

🥾 *Le Guide du Routard Temporel* 🥾

Quentin Bramas[1] et Jean-Romain Luttringer[1] et Sébastien Tixeuil[2]

[1] *Université de Strasbourg, ICube, CNRS* [2] *Sorbonne Université, CNRS, LIP6, IUF*

Dans ce guide, nous considérons des réseaux de transports modernes permettant de voyager dans le temps. Cependant, l'utilisation de dispositifs VIT (Voyage Inversé dans le Temps) induit un coût et peut être sujette à certaines contraintes. Nous modélisons ces réseaux de transports à l'aide de graphes dynamiques, dans lesquels l'agent peut remonter dans le temps avant de continuer sa route. Afin d'aider les routards temporels à naviguer le continuum espace-temps à moindre coût, nous proposons des algorithmes exacts permettant de calculer des itinéraires minimisant le délai de trajet (entre le moment de départ et celui de l'arrivée) tout en minimisant le coût d'utilisation des dispositifs VIT. Nous analysons également l'impact des différentes politiques de facturation sur ces calculs, et identifions les propriétés qu'elles doivent respecter pour permettre aux utilisateurs de planifier correctement leur voyage spatio-temporel.

Mots-clefs : Calcul de chemins, Graphes dynamiques, Voyage dans le temps

1 Introduction

La découverte du voyage dans le temps [†] a révolutionné le monde des transports. Les dispositifs VIT (Voyage Inversé dans le Temps) permettent en effet de revenir vers le passé et diminuent d'autant le délai occasionné par un trajet. Cependant, leur utilisation est coûteuse et les politiques de facturation obscures fixées par les opérateurs induisent une confusion certaine chez les utilisateurs, obligeant ces derniers à faire appel à des taxis temporels non-licenciés. Dans ce guide, nous montrons qu'il est possible de planifier des voyages spatio-temporels minimisant le délai pour un budget raisonnable. Nous modélisons ces réseaux de transport à l'aide de graphes dynamiques, dans lequel les nœuds représentent les dispositifs VIT, et les arêtes la possibilité de se déplacer d'un nœud à un autre (selon les aléas des transports traditionnels). Cependant, nous introduisons dans ce modèle la possibilité pour un agent de revenir à un instant précédent (pour un coût donné) avant de continuer sa route. Les dispositifs VIT sont considérés comme étant toujours accessibles.

Nous étudions la planification d'itinéraires entre un point A et un point B minimisant la différence entre l'instant de départ et l'instant d'arrivée, tout en tenant compte de *(i)* une contrainte sur le coût induit par les retours dans le temps, et *(ii)* une limite sur l'ampleur du retour temporel autorisé par le dispositif.

Nous commençons par étudier l'impact des politiques de facturation des sauts temporels sur la planification de tels trajets. Nous montrons que, si les horaires des transports classiques (spatiaux) sont prédictibles (*i.e.*, le graphe dynamique est connu), des conditions relâchées sur la politique de facturation sont suffisantes pour concevoir des algorithmes polynomiaux optimaux (vis à vis du délai et du coût), malgré une contrainte sur le coût pour l'agent ou sur l'ampleur maximale du retour temporel.

Le détail des résultats et preuves peut être trouvé dans la version longue de cet article [BLT23].

2 Modèle

Nous représentons le réseau sous la forme d'un graphe dynamique tel que défini par Ferreira [Fer02], *i.e.*, sous la forme d'une séquence de graphes statiques. Un graphe G est un couple $(V, (E_t)_{t\in\mathbb{N}})$, où V représente l'ensemble fini des nœuds du graphe, $\mathbb{N}$ est l'ensemble des instants temporels, et E_t représente l'ensemble des arêtes présentes à l'instant $t \in \mathbb{N}$. On note $(\{u, v\}, t)$ une arête qui existe à l'instant t si $\{u, v\} \in E_t$. Nous

†. Pour éviter les paradoxes temporels, nous ne pouvons donner de références postérieures à 2024 justifiant notre propos. Notons cependant que cette découverte a été dans un premier temps utilisée par les chercheurs pour soumettre dans les temps aux conférences sans extension de date limite

supposons qu'à chaque instant temporel t, un agent peut traverser un nombre arbitraire d'arêtes consécutives de E_t. Cependant, le graphe n'est pas nécessairement connecté à chaque instant temporel. Il peut donc être nécessaire de faire un voyage vers le futur (*i.e.*, attendre) ou le passé.

Définition 1 *Un voyage* spatio-temporel *de longueur k est une séquence $((u_0, t_0), (u_1, t_1), \ldots, (u_k, t_k))$ telle que*

- *$\forall i \in \{0, \ldots k\}$, $u_i \in V$ est un nœud et $t_i \in \mathbb{N}$ est un instant temporel,*
- *$\forall i \in \{0, \ldots k-1\}$, si $u_i \neq u_{i+1}$, alors $t_i = t_{i+1}$ et $\{u_i, u_{i+1}\} \in E_{t_i}$* i.e.*, il existe une arête entre u_i et u_{i+1} au temps t_i.*

Dans la suite, $\mathfrak{f}$ désigne une fonction de $\mathbb{N}$ dans $\mathbb{R}^+$ représentant le coût d'un retour dans le passé. Par convention, on pose $\forall x < 0$, $\mathfrak{f}(x) = 0$. Ainsi, pour un voyage $J = ((u_0, t_0), \ldots, (u_k, t_k))$, le coût $cost(J)$ est la somme des coûts des retours dans le passé de J : $cost(J) = \sum_{i=0}^{k-1} \mathfrak{f}(t_i - t_{i+1})$.

Définitions des problèmes. Dans un graphe dynamique G avec un nœud source s et un nœud destination d, nous considérons deux problèmes. Le premier, appelé C-contraint, consiste à trouver, parmi tous les voyages de s vers d dont le coût est inférieur à la contrainte C, un voyage qui arrive le plus tôt possible, et dont le coût est le plus faible (dans un second temps). Le second problème, appelé H-hist-contraint, consiste à trouver un voyage qui respecte la propriété suivante : après avoir atteint une date t, le voyage ne revient jamais dans le passé à une date antérieure à $t - H$, comme illustré dans la Figure 2. Parmi tous les voyages de s vers d respectant cette propriété, on cherche un voyage qui arrive le plus tôt possible et dont le coût est le plus faible.

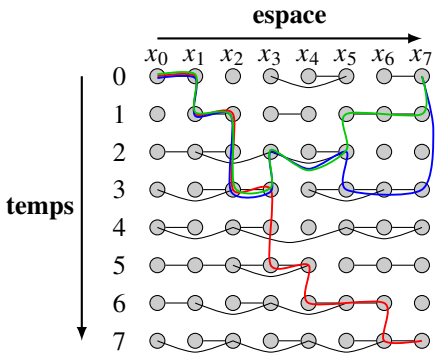

FIGURE 1 : Représentation possible d'un graphe évolutif. Les voyages possibles de x_0 à x_7 sont indiqués en rouge, vert et bleu. Notez que les voyages bleus et verts nécessitent d'envoyer un agent vers le passé (à un instant temporel précédent).

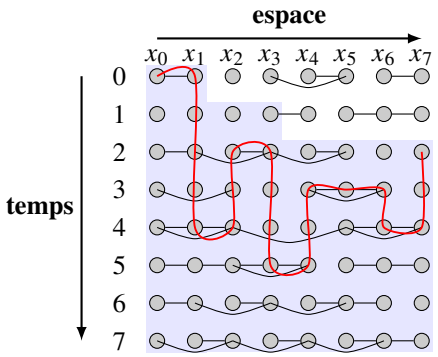

FIGURE 2 : Exemple de chemin 3-hist-contraint. La zone bleue indique les nœuds temporels atteignables par le voyageur. Après avoir atteint le nœud x_1 au temps 4, il n'est plus possible de revenir au temps 0, même après être revenu au temps 2, car la limite est absolue.

Conditions sur la fonction de coût On définit deux ensembles de fonctions de coût. L'ensemble $\mathcal{FO}$ des fonctions *optimisables* qui sont positives et qui atteignent leur minimum dans tous les intervalles $[C, \infty)$, $\forall C \geq 0$, et l'ensemble $\mathcal{FS}$ des fonctions *sympas*, qui sont optimisables, croissantes et sous-additives. Il s'avère que la fonction de coût doit être optimisable pour permettre l'existence d'une solution.

Théorème 1 *Si la fonction de coût n'est pas dans $\mathcal{FO}$, alors il existe des graphes dynamiques connectés dans lesquels les problèmes considérés n'ont pas de solution.*

En effet, si le coût peut être négatif, alors tout voyage peut être remplacé par le même voyage augmenté d'une attente et d'un retour vers le passé, qui a donc un coût inférieur. De manière similaire, s'il existe un C tel que la fonction de coût n'atteint pas son minimum sur $[C, \infty]$, il est toujours possible de trouver un coût inférieur en attendant plus longtemps afin d'augmenter l'ampleur du retour temporel et d'en diminuer le coût. Par ailleurs, si la fonction n'est pas sympa, il est intuitif de voir que l'on peut se ramener à une fonction sympa. Par exemple, s'il est moins coûteux de remonter deux instants dans le passé qu'un seul instant, alors on peut remplacer chaque retour dans le passé d'un instant par une attente suivi d'un retour de deux instants dans le passé. De manière générale, on obtient le théorème suivant.

Théorème 2 *Si un algorithme résout un problème pour toute fonction dans $\mathcal{FS}$, alors il existe un algorithme qui résout le même problème pour toute fonction dans $\mathcal{FO}$.*

On peut ainsi supposer que la fonction de coût est sympa lors de la conception de nos algorithmes : on peut alors automatiquement générer à partir du premier algorithme un nouvel algorithme qui produit un voyage optimal avec une fonction optimisable.

3 Algorithmes

Difficulté et intuition Nous utilisons les principes généraux des algorithmes de plus courts chemins habituels, mais nous conservons, pour chaque nœud, des informations de coût associées à différents instants. Cependant, nous ne pouvons pas considérer le délai comme une métrique standard, sous peine d'obtenir un algorithme exponentiel. De plus, nous ne pouvons pas simplement prendre le graphe statique construit à partir du graphe dynamique en faisant l'union de tous les instants et en reliant les nœuds à différents instants (dont le poids de l'arête orienté vers le passé serait le coût), ce graphe étant potentiellement infini. Notre technique est donc d'utiliser une structure de donnée pour stocker le meilleur coût connu pour atteindre un nœud donné à un instant donné, en s'assurant de ne pas étendre les voyages vers tous les nœuds accessibles, mais uniquement ceux pouvant être optimaux (qui sont en nombre fini).

Algorithme pour le problème C-contraint. La résolution de ce problème réside dans le maintien de deux structures : `cout`, où $\texttt{cout}(u,t)$ est le coût du meilleur voyage C-contraint vers u qui arrive au temps t ; et `Vmin`, où $\texttt{Vmin}(u) = (c_{\min}, t_{c_{\min}})$ signifie que le voyage de coût minimum connu vers u a un coût $c_{\min}$ et qu'un voyage d'un tel coût arrive au mieux au temps $t_{c_{\min}}$. Par défaut `cout` ne contient que $\texttt{cout}(s,0)$ pour le nœud source s. Si une clé lue n'existe pas, elle est initialisée à l'infini. De même $\texttt{Vmin}(s) = (0,0)$.

Notre algorithme consiste en une boucle qui étend les voyages existants d'une arête spatiale, d'une attente ou d'un retour vers le passé, dans l'ordre croissant de leur coût. L'algorithme 1 montre le pseudo-code pour le maintien de la structure `cout` uniquement, par manque de place. De manière similaire à l'algorithme de Dijkstra, la propriété qu'un sous-voyage d'un voyage optimal est optimal garantit que si une solution est trouvée vers la destination, elle sera correcte. Cependant, lors de l'étape d'extension, un voyage n'est pas étendu de toutes les manières possibles (car il y en a une infinité). Un voyage terminant sur le nœud u au temps t est étendu vers tous ses voisins en utilisant (a) toutes les arêtes temporelles présentes avant le (ou au) temps t (ce nombre est fini) ; et (b) une seule arête future (la plus proche dans le futur si elle existe) si le voyage obtenu fait mieux que le voyage de coût minimum stocké dans `Vmin`. Après extension vers un nœud v au temps t', si le coût du voyage est de coût inférieur à $\texttt{cout}(v,t')$, la structure est mise à jour. De plus, la structure `Vmin` est mise à jour à chaque fois qu'un voyage de coût inférieur est trouvé.

Ainsi, lorsqu'il n'existe plus de voyage à étendre, le coût du voyage optimal vers la destination est le coût (non-infini) correspondant à la clé (d,t), pour t minimum. Une induction à rebours permet de retrouver le voyage complet de la source s vers la destination d.

Théorème 3 *Étant donné un graphe dynamique G et une contrainte C, notre algorithme termine toujours en temps fini, et trouve, si elle existe, une solution au problème du voyage C-contraint.*

Pour calculer la complexité de cet algorithme, on s'intéresse au nombre de fois qu'un voyage est étendu. Comme dans le cas d'un algorithme de plus court chemin, un voyage arrivant à un nœud u au temps t n'est étendu qu'une seule fois. Comme notre algorithme termine toujours, le nombre d'extensions est fini et une extension (u,t) se fait toujours à partir d'un nœud qui possède une arête temporelle au temps t. On définit donc $\mathcal{E}$, l'ensemble des arêtes temporelles associées aux extensions. Ce sous-ensemble de E est toujours fini, même si E est infini. Le nombre d'arêtes par lesquels un chemin est étendu est aussi inclus dans $\mathcal{E}$, donc la complexité totale de notre algorithme est en $O(|\mathcal{E}|^2)$.

Algorithme pour le problème H-hist-contraint. Pour ce problème, la méthode d'extension des voyages est plus proche de l'algorithme Bellman-Ford, puisqu'il est plus difficile d'ordonner les meilleurs voyages connus. Nous utilisons une structure `Hcout` dont les clés sont des tuples $(u, t-h, t)$ et la valeur associée à une telle clé représente le coût minimal d'un voyage de s vers u arrivant au temps $t-h$ et dont le temps maximum atteint dans le futur est t.

Algorithme 1 : C-contraint optimal

$Restant \leftarrow \{(s, 0)\}$
tant que $Restant \neq \emptyset$ **faire**
  $(u, t) \leftarrow \text{argmin}_{(u,t) \in Restant}(\texttt{cout}(u, t))$
  pour *v voisin de u* **faire**
    pour *les arêtes* $\{u, v\} \in E_{t'}$ *d'intérêt* **faire**
      si $cout(u, t) + f(t - t') < cout(v, t')$ **alors**
        $\texttt{cout}(v, t') \leftarrow \texttt{cout}(u, t) + f(t - t')$
        $Restant \leftarrow Restant \cup \{(v, t')\}$

Algorithme 2 : H-hist-contraint optimal

pour $t = 0, \ldots, t_{\max}$ **faire**
  pour $0 \leq h' \leq h \leq H$ **faire**
    $\texttt{Hcout}(u, t - h, t) \leftarrow \texttt{Hcout}(u, t - h', t - 1)$;
  répéter n **fois**
    pour $\{u, v\} \in E'_t, t' \in [t - H, t]$ **faire**
      pour $h = H, H - 1, \ldots, 0$ **faire**
        mettre à jour $\texttt{Hcout}(u, t - h, t)$ en fonction de $\texttt{Hcout}(v, t', t)$

Comme illustré dans le pseudo-code de l'Algorithme 2, nous itérons sur chaque instant t entre 0 et une borne $t_{\max}$ qu'une solution peut atteindre (pouvant être calculé en $O(n)$ par un algorithme simple). Pour chaque t, le but est de calculer toutes les valeurs de $\texttt{Hcout}(u, t - h, t)$ pour tout $h \in [0, H]$. Le calcul de ces valeurs dépend des valeurs des itérations précédentes et du graphe dynamique aux instants $[t - H, t]$.

Pour un t donné, l'algorithme répète n fois l'extension de tous les voyages connus le long des arêtes temporelles de l'intervalle $[t - H, t]$. Un voyage se terminant sur le nœud u au temps $t - h$ ne dépassant jamais le temps t est étendu vers tous ses voisins en utilisant toutes les arêtes temporelles connectées à u et présentent entre les temps $t - H$ et t. Dès qu'un voyage est trouvé arrivant au nœud d au temps $t_{\min}$, on sait que $t_{\min}$ est le temps d'arrivée de la solution. Il faut encore effectuer au plus H itérations pour s'assurer de trouver le coût minimal. Si la boucle principale ($t = t_{\max}$) se termine sans avoir trouvé la valeur de $t_{\min}$, c'est qu'il n'y a aucune solution au problème.

Les arêtes temporelles apparaissant après une borne $t_{\max}$ sont ignorées par notre algorithme, ainsi, une itération pour un t donné a une complexité en $O(nm_{\max}H)$, où $m_{\max}$ est le nombre d'arêtes temporelles apparaissant avant t_{max}. Le nombre d'itérations globales est $\min(t_{\max}, t_{\min}+H)$. Ainsi la complexité globale de notre algorithme est $O(nm_{\max}H \min(t_{\max}, t_{\min} + H))$. Il est intéressant de noter que si $t_{\min}$ est plus grand que $m_{\max}$, cela signifie qu'il y a des instants où aucune arête n'est présente, et notre algorithme peut être optimisé pour ignorer ces instants (en les prenant malgré tout en compte dans la vérification de la contrainte), la complexité ne dépasse donc pas $O(nm^2_{\max}H^2)$.

4 Conclusion

Nous avons défini les voyages dans les graphes dynamiques où il est possible de revenir dans le passé (pour un coût associé). Nous avons donné des conditions nécessaires sur la fonction de coût pour permettre l'existence de solutions au problème du voyage minimisant le délai et le coût, et nous avons défini deux problèmes de calcul de voyages optimaux contraints dans ce modèle. Enfin, nous avons donné des algorithmes résolvant ces problèmes dont les complexités sont polynomiales en fonction d'un sous-ensemble fini d'arêtes temporelles. Une perspective intéressante est de considérer le problème *en ligne* associé, c'est-à-dire dans lequel le voyageur découvre le graphe au fur et à mesure de son exploration. De plus, il serait intéressant de considérer des coûts d'attente non nuls (vieillissement) pour éviter les décès prématurés des voyageurs.

Références

[BLT23] Quentin Bramas, Jean-Romain Luttringer, and Sébastien Tixeuil. Offline constrained backward time travel planning. In Shlomi Dolev and Baruch Schieber, editors, *Stabilization, Safety, and Security of Distributed Systems - 25th International Symposium, SSS 2023, Jersey City, NJ, USA, October 2-4, 2023, Proceedings*, volume 14310 of *Lecture Notes in Computer Science*, pages 466–480. Springer, 2023.

[Fer02] Afonso Ferreira. On models and algorithms for dynamic communication networks : The case for evolving graphs. In *Quatrièmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (ALGOTEL 2002)*, pages 155–161, Mèze, France, May 2002. INRIA Press.