Impact de la granularité spatio-temporelle des données sur l'optimisation des tournées de livraison en ville

Omar Rifki, Nicolas Chiabaut, Christine Solnon

Impact de la granularité spatio-temporelle des données sur l'optimisation des tournées de livraison en ville*

Omar Rifki[1,2] **Nicolas Chiabaut**[1] **Christine Solnon**[2]

[1] Université de Lyon, ENTPE / IFSTTAR, LICIT, UMR _T 9401, F-69518, Lyon, France
[2] Université de Lyon, INSA-Lyon, CNRS, LIRIS, UMR5205, F-69621, France
nicolas.chiabaut@entpe.fr {omar.rifki, christine.solnon}@insa-lyon.com

Résumé

L'optimisation des tournées de livraisons se ramène essentiellement à un problème de voyageur de commerce (TSP) dans un graphe où les sommets correspondent aux points à livrer, et les coûts des arcs aux temps de trajet entre deux points. Le TSP dépendant du temps (TD-TSP) est une extension du TSP dans laquelle le coût d'un arc dépend de l'heure de départ du nœud d'origine. Cette extension est particulièrement pertinente dans un contexte urbain, car les vitesses de déplacement varient en fonction de l'heure de la journée. Les fonctions de coût dépendantes du temps sont construites à partir de données issues de capteurs, et nous pouvons considérer pour cela différents niveaux de granularité temporelle (fréquence des mesures considérée) et spatiale (nombre et positionnement des capteurs dans le réseau routier). L'objectif de cet article est d'étudier l'impact de la granularité spatio-temporelle des fonctions de coût sur la qualité des tournées calculées ainsi que sur les temps de calcul. Pour cela, nous avons généré différents jeux de données en utilisant un logiciel de micro-simulation du trafic urbain, permettant d'obtenir des données très réalistes à différents niveaux de granularité spatio-temporelle. Nous considérons également différentes approches de résolution exactes (programmation linéaire en nombres entiers, *Branch&Bound* et programmation dynamique) et heuristiques (programmation dynamique restreinte et *limited discrepancy search*).

1 Introduction

Nous considérons un problème très général d'optimisation de tournées de livraison consistant à minimiser le temps nécessaire pour visiter un ensemble de sites de livraison ou de reprise de marchandises. Ce problème est généralement modélisé par un graphe orienté, où les sommets correspondent aux points à visiter, les arcs aux plus courts chemins entre sommets, et les coûts des arcs aux durées des plus courts chemins. La tournée optimale correspond alors au plus court circuit visitant chaque sommet du graphe exactement une fois, *i.e.*, la solution du problème du voyageur de commerce (*Traveling Salesman Problem* ou *TSP*).

Dans le TSP classique, les coûts des arcs sont supposés constants, *i.e.*, les temps de trajet sont les mêmes quelle que soit l'heure de la journée. Cette supposition n'est pas réaliste car les conditions de circulation varient du fait des embouteillages aux heures de pointe. En pratique, les plus courts chemins (*i.e.*, successions de tronçons routiers) entre les sites peuvent changer au cours de la journée, et leur durée également. Pour combler ce manque de réalisme, les fonctions définissant les coûts des arcs doivent être *dépendantes du temps (Time-Dependent ou TD)*, *i.e.*, le coût d'un arc doit dépendre de l'heure à laquelle il est traversé. Le TSP avec des fonctions de coût dépendantes du temps est appelé TD-TSP [9, 12, 13].

Dans cet article, nous supposons que les données de temps de trajet dépendantes du temps sont définies à l'aide de fonctions constantes par morceaux, avec des pas de temps de longueurs égales : si l'horizon temporel est de longueur totale H et si la longueur du pas de temps est s, alors il y a H/s pas de temps et, pour chacun de ces pas de temps et chaque tronçon de route possible, la durée de traversée du tronçon est supposée constante pendant ce pas de temps. Dans les faits, ce modèle est bien adapté aux données de trafic, car il

*Travail soutenu par le projet ELUD du Labex IMU. Les remerciements sont dus aussi à Matthis Manthe pour un travail précédent sur les algorithmes de résolution.

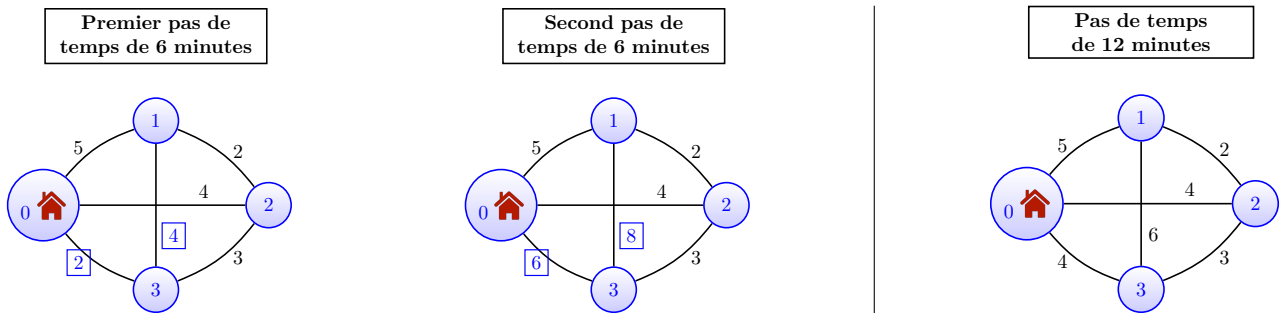

FIGURE 1 – Illustration de l'impact de la granularité des pas de temps sur les tournées calculées. Le réseau considéré a 4 sommets : le dépôt noté 0 et trois adresses de livraison $\{1, 2, 3\}$. Les arêtes représentent les plus courts chemins. À gauche : deux pas de temps de 6 minutes chacun, où les coûts des arcs $(0, 3)$ et $(1, 3)$ augmentent entre le premier et le deuxième pas ; la meilleur tournée est $\langle 0, 3, 1, 2, 0\rangle$ (afin de traverser les arcs $(0, 3)$ et $(3, 1)$ avant qu'ils ne soient congestionnés). À droite : un seul pas de temps de 12 minutes tel que les coûts des arcs $(0, 3)$ et $(1, 3)$ sont des moyennes des deux pas de gauche ; la meilleure tournée devient $\langle 0, 1, 2, 3, 0\rangle$.

correspond au schéma habituel d'estimation des durées de trajet. La figure 1 montre, à travers un exemple simple, comment la modification de la longueur s des pas de temps de 6 à 12 minutes peut impacter la qualité d'une tournée. Elle peut également avoir un impact sur l'efficacité de résolution des algorithmes utilisés pour calculer la solution optimale. Par ailleurs, les fonctions de coût sont construites à partir de données issues de capteurs, et les caractéristiques de ces fonctions diffèrent selon le nombre de capteurs et leur positionnement dans le réseau routier.

Notre objectif est d'étudier l'impact de la granularité spatio-temporelle des données sur la qualité des solutions calculées d'une part et les temps de calcul d'autre part. Dans la section 2, nous décrivons le TD-TSP et les approches de résolution considérées : exactes (programmation linéaire en nombres entiers, *Branch&Bound* et programmation dynamique), et heuristiques (programmation dynamique restreinte et *limited discrepancy search*). Dans la section 3, nous décrivons les jeux de données que nous avons générés. Nous avons utilisé pour cela un logiciel de micro-simulation du trafic urbain, permettant de générer des jeux réalistes où nous pouvons faire varier d'une part la taille des pas de temps (pour la dimension temporelle) et d'autre part le nombre de capteurs utilisés pour évaluer les vitesses (pour la dimension spatiale). Dans la section 4, nous évaluons l'impact de la granularité spatio-temporelle des données sur la qualité des tournées calculées et sur les temps de calcul des algorithmes.

2 Le problème du TD-TSP

2.1 Description du problème

Nous considérons des graphes orientés et complets dont les sommets correspondent aux adresses de livraison (nous montrons dans la section 3 comment ces graphes sont construits à partir d'un réseau de transport routier). On note n le nombre de sommets, V l'ensemble des sommets et $0 \in V$ le dépôt à partir duquel le tour commence.

Introduit en 1992 par Malandraki et Daskin [22], le TD-TSP est une généralisation du TSP qui vise à trouver le circuit hamiltonien de durée minimale dans le cas où le coût de chaque arc (i, j) varie dans le temps et dépend de l'heure de départ t du sommet i. Ce coût sera noté c_{ij}^t. Étant donné un chemin $P = \langle v_1, \ldots, v_k\rangle$, une heure de départ t_0 et un sommet $v_i \in P$, l'heure d'arrivée sur v_i est notée $at(v_i, t_0, P)$ et est définie récursivement par :

$$\begin{aligned} at(v_1, t_0, P) &= t_0 \\ \forall i \in [2, k],\ at(v_i, t_0, P) &= at(v_{i-1}, t_0, P) + d(v_{i-1}) \\ &\quad + c_{v_{i-1}, v_i}^{at(v_{i-1}, t_0, P) + d(v_{i-1})}, \end{aligned}$$

où $d(v_i)$ est la durée d'arrêt associée au sommet v_i. La durée de parcours associée au chemin P et à une heure de départ t_0 est notée $ts(t_0, P)$ et est définie par :

$$ts(t_0, P) = at(v_k, t_0, P) - \sum_{i=1}^{k} d(v_i) - t_0.$$

L'objectif du TD-TSP est de trouver un circuit $P = \langle v_0, \ldots, v_n\rangle$ tel que P commence et termine sur le sommet 0 (*i.e.* , $v_0 = v_n = 0$) ; $\{v_0, \ldots, v_{n-1}\}$ est une permutation de V ; et la durée de parcours de P en partant de v_0 à t_0 (*i.e.*, $ts(t_0, P)$) est minimale.

2.2 Approches de résolution

Différentes approches ont été proposées pour résoudre le TD-TSP, depuis son introduction par [22]. Certaines sont basées sur des méta-heuristiques telles

que la recherche taboue [31], le recuit simulé [30] ou les algorithmes de colonies de fourmis [10]. Dans cette section, nous nous focalisons sur les approches exactes, *i.e.*, la programmation par contraintes (*Constraint Programming - CP*), la programmation linéaire en nombres entiers (*Integer Linear Programming - ILP*), *Branch&Bound* (*B&B*) et la programmation dynamique (*Dynamic Programming - DP*), ainsi que sur les approches heuristiques dérivées de ces méthodes exactes, *i.e.*, *Limited Discrepancy Search* (*LDS*), et la programmation dynamique restreinte (*Restricted DP - RDP*).

CP. Melgarejo *et al.* ont introduit dans [24] la contrainte globale *TDNoOverlap*, qui garantit qu'un ensemble de tâches ne se chevauchent pas lorsque les temps de transition entre les tâches dépendent du temps. Cette contrainte peut être utilisée pour résoudre le TD-TSP. Elle est bien plus efficace que les modèles CP classiques pour le TD-TSP (basés sur les contraintes *allDifferent*) puisqu'elle propage les relations de précédence entre les tâches. Cependant, *TDNoOverlap* ne passe pas bien à l'échelle en l'absence de fenêtres temporelles (*i.e.*, contraintes relatives aux heures d'arrivée et de départ). Par exemple, elle n'est pas toujours en mesure de résoudre des instances avec $n = 20$ sommets dans un délai de 900 secondes. Par conséquent, nous ne considérons pas CP dans notre étude expérimentale.

ILP. La première formulation linéaire en nombres entiers du problème est due à Malandraki et Daskin [22]. Par la suite, plusieurs approches ont été conçues. Cordeau *et al.* [8] ont notamment proposé une méthode permettant de calculer une borne inférieure du problème et de résoudre ainsi le TD-TSP à l'aide d'un algorithme *Branch & Cut* (B&C). Cependant certaines instances ayant 40 sommets ne sont pas résolues en un temps raisonnable, même lorsque les fonctions de coût ne comportent que trois pas de temps. Arigliano *et al.* [1] ont étendu le modèle de [8] aux fenêtres temporelles (TD-TSPTW), et Montero *et al.* [25] ont mis au point un algorithme B&C pour résoudre un problème similaire. Récemment, Vu *et al.* [32] et Boland *et al.* [4] ont proposé une méthode de résolution basée sur des graphes étendus dans le temps. A notre connaissance, cette méthode représente actuellement l'état de l'art pour le TD-TSPTW : elle est capable de résoudre en quelques secondes des instances ayant jusqu'à $n = 40$ sommets, même lorsque le nombre de pas de temps est égal à 73, alors que l'approche de [25] ne passe pas à l'échelle dans ce cas. Cependant, l'approche de [32] ne passe pas à l'échelle lorsqu'on considère des instances sans fenêtres temporelles. Nous l'avons testé [1] sur nos instances (qui n'ont pas de fenêtres temporelles) et avons constaté qu'elles ont besoin de plusieurs heures pour converger, même dans le cas de petites instances.

1. Nous remercions Vu *et al.* d'avoir partagé leur code source.

Algorithme 1 : B&B(P)

Input : Un chemin $P = \langle v_0, \ldots, v_i \rangle$ allant du dépôt $v_0 = 0$ à un sommet v_i

1 Soit t_0 l'heure de départ du dépôt 0
2 Soit $t*$ l'heure d'arrivée de la meilleure tournée trouvée
3 Soit $C = V \setminus \{v_0, \ldots, v_i\}$ l'ensemble des sommets qui ne sont pas encore visités par P
4 **si** $C = \emptyset$ **alors**
5 **si** $at(v_i, t_0, P) + d(v_i) + c_{v_i,v_0}^{at(v_i,t_0,P)+d(v_i)} < t*$ **alors**
6 mettre à jour $t*$
7 **sinon**
8 **pour** *chaque sommet* $v_j \in C$ **faire**
9 $h_{v_j} \leftarrow at(v_j, t_0, P.\langle v_j \rangle) + d(v_j) + bound(v_j, C)$
10 **pour** *chaque sommet* $v_j \in C$ *considéré par valeur croissante de* h_{v_j} **faire**
11 **si** $h_{v_j} < t*$ **alors** B&B($P.\langle v_j \rangle$);

Par conséquent, étant donné que les approches ILP existantes ne sont pas capables de résoudre en un temps raisonnable des instances ayant plus de 3 pas de temps et pas de fenêtres temporelles, nous ne les considérons pas dans notre étude expérimentale (sauf pour le cas statique avec un seul pas de temps).

B&B. Nous considérons une approche B&B classique décrite dans l'algorithme 1. Le paramètre en entrée P est un chemin du dépôt au sommet actuel v_i (initialement, $P = \langle 0 \rangle$). Si P contient tous les sommets et si l'heure d'arrivée au dépôt est inférieure à la meilleure solution trouvée jusqu'à présent, la meilleure solution est mise à jour (lignes 4-6). Sinon, pour chaque sommet non visité $v_j \in C$, B&B calcule une borne inférieure h_{v_j} de l'heure d'arrivée de la meilleure tournée qui visite successivement $v_0, \ldots, v_i$, puis v_j, puis tous les sommets restants de $C \setminus \{v_j\}$, et retourne enfin au dépôt (lignes 8-9). Le calcul de cette borne inférieure est détaillé ci-dessous. Enfin, l'algorithme itère sur chaque sommet non visité $v_j \in C$ par ordre croissant de la borne inférieure h_{v_j}. Si h_{v_j} est inférieure à la meilleure solution trouvée jusqu'à présent, l'algorithme appelle récursivement B&B avec v_j ajouté à la fin du chemin P (lignes 10-11).

La fonction $bound(v_j, C)$ calcule une borne inférieure de la durée du plus court chemin qui part de v_j à l'heure $t_j = at(v_j, t_0, P.\langle v_j \rangle) + d(v_j)$, puis visite exactement une fois tous les sommets de $C \setminus \{v_j\}$, et termine enfin au dépôt. Ce type de borne est généralement calculé en résolvant des relaxations du problème initial. Pour le TSP asymétrique (ATSP), le problème d'affectation (*Assignment Problem - AP*) est un problème classique

qui relâche la contrainte de connectivité du chemin : l'objectif de AP est de sélectionner un ensemble d'arcs dans $\{(v_k, v_l) : v_k \in C, v_l \in \{0\} \cup C \setminus \{v_j\}\}$ tel que (i) v_j possède exactement un arc sortant ; (ii) 0 a exactement un arc entrant ; (iii) chaque sommet de $C \setminus \{v_j\}$ a exactement un arc entrant et un arc sortant ; et (iv) la somme des coûts des arcs sélectionnés est minimale. Cette relaxation peut être résolue en $\mathcal{O}(n^3)$ en utilisant des variantes améliorées de l'algorithme Hongrois [6].

Cependant, pour le TD-TSP, le coût d'un arc (v_k, v_l) dépend de l'heure de départ de v_k qui n'est pas connue (sauf pour v_j). Pour assurer que le coût du AP est une borne inférieure du chemin optimal, nous devons considérer le plus bas coût possible pour chaque arc (v_k, v_l), *i.e.*, $c^{min}_{v_k v_l} = \min_{l_k \leq t \leq u_k} c^t_{v_k v_l}$ où l_k (resp. u_k) est l'heure de départ possible la plus petite (resp. la plus grande) à partir de v_k. Bien sûr, plus $u_k - l_k$ est restreinte, plus $c^{min}_{v_k v_l}$ est élevé, et meilleure est la borne calculée par AP. Dans cet article, nous calculons l_k et u_k comme suit : l_k correspond au temps d'arrivée à v_k si v_k est visité juste après v_j (*i.e.*, $l_k = at(v_j, t_0, P.\langle v_j \rangle) + d(v_j) + c^{at(v_j, t_0, P.\langle v_j \rangle) + d(v_j)}_{v_j v_k}$) ; et u_k est l'heure de départ au plus tard de v_k qui permet de retourner au dépôt à $t*$ (*i.e.*, $u_k = t* - c^{t* - c^{min}_{v_k v_0}}_{v_k v_0}$), où $t*$ est l'heure d'arrivée de la meilleure tournée trouvée. Notons que nous pourrions encore réduire l'intervalle $[l_k, u_k]$ en inférant des relations de précédence entre les sommets, comme proposé dans [24]. Cependant, cela n'a pas (encore) été implémenté.

Comme nous devons résoudre AP pour chaque sommet candidat $v_j \in C$, et que les différentes instances de AP à résoudre ne diffèrent que d'un seul sommet, nous utilisons la version incrémentale de l'algorithme Hongrois qui résout plus efficacement AP en réparant une affectation existante [11].

LDS. La recherche à divergence limitée LDS [15] n'explore que partiellement l'arbre de recherche construit par B&B. Elle suppose qu'il existe une heuristique pouvant être utilisée à chaque sommet de l'arbre afin de classer toutes les décisions possibles, de la meilleure à la plus mauvaise : pour chaque nœud la divergence d'une décision est égale à son rang (où la meilleure décision a le rang 0) ; et la divergence d'une branche de l'arbre de recherche (de la racine à un nœud donné) est égale à la somme des divergences des décisions prises à chacun de ses nœuds. L'idée de LDS est d'explorer les branches en augmentant la divergence : LDS explore d'abord la branche sans divergence (la meilleure décision est choisie à chaque nœud) ; puis toutes les branches dont la divergence est égale à 1 ; puis celles dont la divergence est égal à 2, etc.

LDS est connue pour explorer certains nœuds plusieurs fois. Pour surmonter ce problème, nous considérons une variante de LDS introduite par Beck et Perron dans [2], appelée *Discrepancy-Bounded Depth First Search* (*DBDFS*) : l'idée est d'effectuer un parcours en profondeur sur tous les nœuds avec des chemins qui ont jusqu'à un certain nombre borné de divergences.

Pour le TD-TSP, les décisions possibles à chaque nœud de l'arbre de recherche sont les sélections d'un sommet non visité $v_j \in C$, et l'heuristique utilisée pour ordonner les décisions est h_{v_j} (calculée à la ligne 9 de l'algorithme 1). Cette heuristique renvoie une borne inférieure de la tournée optimale commençant par $P.\langle v_j \rangle$. Au lieu de définir la divergence de v_j simplement par son rang, nous la définissons comme suit : $h_{v_j} - \min_{v_k \in C} h_{v_k}$ (comme celle proposé dans [5], par exemple). En d'autres termes, plus h_{v_j} est proche de la meilleure valeur heuristique, plus la divergence est petite. Par conséquent, notre algorithme LDS est obtenu à partir de l'algorithme 1 en modifiant la condition pour l'appel récursif de $B\&B$ (ligne 11) comme suit :

$$h_{v_j} < t* \text{ et } d_{tot} + h_{v_j} - \min_{v_k \in C} h_{v_k} < d_{max}$$

où d_{tot} est la divergence totale du chemin P (elle est incrémentalement calculée en ajoutant $h_{v_j} - \min_{v_k \in C} h_{v_k}$ à chaque appel récursif), et d_{max} est un paramètre de l'algorithme qui définit la divergence maximal.

DP. La méthode de programmation dynamique proposée par Held et Karp pour résoudre le TSP [16] peut être facilement étendue au TD-TSP. Plus précisément, pour chaque sommet $v_i \in V$ et chaque sous-ensemble de sommets $S \subseteq V \setminus \{0, v_i\}$, soit $p(v_i, S)$ le temps d'arrivée au plus tôt d'un chemin qui commence au sommet 0 à t_0, visite chaque sommet de S exactement une fois, et se termine au sommet v_i. Les équations de Bellman qui définissent récursivement $p(v_i, S)$ sont :

$$\begin{aligned} \text{si } S = \emptyset,\ p(v_i, S) &= c^{t_0}_{0 v_i} \\ \text{sinon, } p(v_i, S) &= \min_{v_j \in S} p(v_j, S \setminus \{v_j\}) + d(v_j) + \\ &\quad c^{p(v_j, S \setminus \{v_j\}) + d(v_j)}_{v_j v_i} \end{aligned}$$

La solution du TD-TSP est donnée par $p(0, V \setminus \{0\})$, et peut être calculée itérativement en considérant des sous-ensembles S de cardinalités croissantes : nous calculons d'abord $p(v_i, \emptyset)$ pour chaque sommet v_i ; puis, à chaque niveau k (avec k allant de 1 à $n-1$), nous calculons $p(v_i, S)$ pour chaque sommet v_i et chaque sous-ensemble S de cardinalité k. Nous utilisons un tableau de n bits pour représenter chaque sous-ensemble S, de sorte que le $j$ème bit soit égal à 1 ssi $j \in S$. La complexité temporelle de cet algorithme est $\mathcal{O}(n^2 \cdot 2^n)$ et sa complexité spatiale est $\mathcal{O}(n \cdot 2^n)$.

RDP. Pour éviter une explosion de DP, à la fois en termes de temps et de mémoire, RDP limite le nombre de résultats mémorisés à chaque niveau k. Cette idée est utilisée pour résoudre le TD-TSP dans [23]. Elle est également utilisée dans [3] pour calculer des solutions approchées de divers problèmes d'optimisation avec des diagrammes de décision restreints.

RDP a un seul paramètre H qui correspond au nombre maximal de résultats mémorisés à chaque niveau k : pour chaque résultat $p(v_i, S)$ mémorisé au niveau précédent $k-1$ et chaque sommet $v_j \in V \setminus (S \cup \{v_i\})$, RDP calcule $p(v_j, S \cup \{v_i\})$ et, si le nombre total de résultats calculés dépasse H, il mémorise les H meilleurs résultats. Comme pour DP, nous utilisons des vecteurs de bits pour représenter les sous-ensembles. Nous utilisons également une file de priorité pour mémoriser les H meilleurs résultats et une table de hachage pour un accès en temps constant à l'index d'un résultat dans la file de priorité. La complexité spatiale de cet algorithme est $\mathcal{O}(H)$ et sa complexité temporelle est $\mathcal{O}(n^2 H \cdot \log H)$ (il y a n itérations et, à chaque itération, il y a au plus H résultats et pour chacun de ces résultats, il y a au plus n successeurs possibles ; le facteur $\log H$ provient du fait que nous utilisons une file de priorité pour stocker les résultats).

3 Description des données

3.1 Limitations des données existantes

Pour comprendre et évaluer les effets des conditions de trafic et de la granularité des coûts dépendants du temps sur les tournées optimales, il est primordial de disposer de données réalistes. En effet, les algorithmes du TD-TSP sont généralement évalués sur des données artificielles, et dans la plupart des cas, ils ne prennent en compte que peu de pas de temps.

Par exemple, les données introduites dans [8], qui sont largement utilisées pour évaluer les algorithmes du TD-TSP, ne considèrent que trois pas de temps : le premier et le troisième pas correspondent respectivement aux heures de pointe du matin et du soir, tandis que le deuxième pas correspond au milieu de la journée, lorsque la densité du trafic est plus faible. De plus, ces données sont générées d'une manière aléatoire et, même si des hypothèses réalistes ont été émises pour générer des temps de parcours (en considérant trois zones concentriques, avec un réglage différent pour chaque zone et des scénarios différents correspondant à des modèles de trafic différents), il est probable que les conclusions tirées de ces données artificielles ne seraient plus valables sur des données réelles. En particulier, une question importante abordée dans notre étude expérimentale est la suivante : pouvons-nous calculer de meilleurs tours si nous considérons des données de trafic en entrée définies à un niveau de granularité plus fin ? Pour répondre à cette question, nous avons besoin de données réalistes, à différents niveaux de granularité temporelle et spatiale avec des variations réalistes pour les temps de parcours.

Un benchmark plus réaliste est introduit dans [24]. Il a été généré à l'aide de données de trafic réelles provenant de capteurs (boucles magnétiques) qui mesurent les conditions de circulation sur 630 tronçons routiers de la ville de Lyon. Cependant, ce benchmark présente trois principales faiblesses.

Premièrement, la majorité des tronçons ne sont pas équipés de capteurs et, dans ce cas, la vitesse a été interpolée en fonction des capteurs les plus proches. De plus, les capteurs considérés (boucles magnétiques) ont des faiblesses bien connues en théorie du trafic, en particulier pour estimer les temps de trajet [21]. En effet, ils enregistrent des conditions de trafic de manière ponctuelle et isolée. Par conséquent, si une file d'attente se produit dans une rue mais n'atteint pas le capteur, elle ne sera pas enregistrée, ce qui entraînera une surestimation de la vitesse. Au contraire, si une file d'attente dépasse les capteurs, elle ne mesurera que la congestion, ce qui n'est pas totalement représentatif des conditions de circulation dans la rue et conduit à une sous-estimation de la vitesse.

Deuxièmement, la fonction de coût dépendante du temps entre deux adresses de livraison a été obtenue en considérant que la durée d'un chemin est égale à la somme des temps de parcours de ses tronçons de route. Cela sous-estime le temps de parcours réel car le temps passé sur les sommets (correspondant aux carrefours entre tronçons) n'est pas pris en compte, alors que le temps nécessaire pour traverser un carrefour ou pour virer à gauche est un facteur important d'augmentation des temps de parcours aux heures de pointe.

Enfin, le benchmark de [24] a été généré à l'aide de données mesurées entre 6h00 et 12h30. Si cela est suffisant pour les petites tournées, avec un maximum de 30 points de livraison, des tournées avec 50 ou 100 adresses de livraisons ont généralement des heures d'arrivée dépassant 12h30, même en supposant que la tournée commence à 6h00.

Un objectif de cette étude est de fournir à la communauté de chercheurs un jeu de données pour évaluer l'impact de la granularité spatio-temporelle des données sur la qualité des solutions trouvées d'une part et sur les performances des algorithmes de résolution du TD-TSP d'autre part.

3.2 Estimation de la durée d'un tronçon

Pour obtenir un accès complet à des données réalistes, nous utilisons un simulateur dynamique et mi-

FIGURE 2 – Le réseau routier lyonnais considéré dans cette étude et les positions réelles des capteurs placés par la collectivité de la Métropole de Lyon (en jaune).

croscopique du trafic, appelé SYMUVIA [7], sur une partie du réseau de transport de Lyon, visible en figure 2. SYMUVIA simule toute la complexité du trafic en tenant compte des différentes classes de véhicules, du comportement de conduite individuelle, des phénomènes de changement de voie, etc. Il utilise une loi de suivi de voiture basée sur le modèle de transport de Newell [26] et ses extensions [19, 20]. Même si la simulation n'est qu'une approximation du monde réel, elle permet d'avoir un accès aux détails les plus fins et d'émuler toutes les mesures possibles de la dynamique du trafic : temps de parcours individuels, vitesses des liaisons, données mesurées par les capteurs (boucles magnétiques), etc. Afin d'obtenir une simulation réaliste, les données utilisées par la simulation (nombre de véhicules, etc) ont été calculées à partir des données mesurées par les capteurs réels (dont les positions sont données dans la figure 2). La simulation permet d'obtenir, pour chaque tronçon de route r du réseau et chaque instant t de la journée, deux valeurs correspondant à ce qui aurait été mesuré à l'instant t par un capteur réel qui serait positionné le long du tronçon r, à savoir la concentration K (véh./m) et le débit Q (véh./s), avec des conditions de trafic identiques à la simulation. Les deux valeurs sont le taux d'occupation et le débit observé sur le tronçon, et elles permettent d'en déduire la durée de traversée du tronçon à l'instant t.

3.3 Sources d'imprécision des données

Bien évidemment, au moment où un transporteur planifie une tournée de livraison, il ne connaît pas les conditions exactes de circulation qui seront observées pendant la réalisation de la tournée, et il doit utiliser un modèle prédictif pour estimer ces conditions de circulation. La conception de ces modèles prédictifs en milieu urbain est un sujet de recherche très actif que nous n'abordons pas ici (voir, par exemple, [29] pour une comparaison de différents modèles prédictifs utilisant les données issues des capteurs visualisés dans la figure 2). Nous supposons dans cette étude que nous disposons d'un modèle prédictif parfait pour chaque capteur, et nous considérons deux cas : le cas (optimiste) où chaque tronçon du réseau est équipé d'un capteur, et le cas (réaliste) où les capteurs sont positionnés exactement aux mêmes endroits que dans le réseau lyonnais actuel (cf figure 2), ce qui correspond à une couverture de 7,35% des tronçons. Dans ce cas, les données des tronçons non couverts par un capteur sont générées par interpolation en considérant le capteur le plus proche (en terme de distance Euclidienne).

Nous obtenons les deux jeux de données suivants : D^+ est le jeu obtenu avec une couverture totale des tronçons par les capteurs, et D^- est le jeu obtenu avec une couverture partielle de $7,35\%$ des tronçons. Chacun de ces jeux donne, pour chaque tronçon de route et chaque période de 30 secondes, la durée de traversée du tronçon à cette période.

Enfin, pour chaque jeu $D^* \in \{D^+, D^-\}$, nous avons généré 5 fonctions de coût notées D^*Sl où $l \in \{6, 12, 24, 60, 720\}$ est la longueur du pas de temps (en minutes), définissant la fréquence à laquelle les durées de traversée sont calculées : la fonction $D^*Sl(r)$ définissant la durée de traversée d'un tronçon r est une fonction constante par morceaux, chaque morceau étant un intervalle de l minutes, et la durée de traversée de r à l'intervalle $[t, t+l[$ est obtenue à travers les moyennes des concentrations K et des débits Q des $2*l$ périodes de 30 secondes de $[t, t+l[$ dans le jeu D^*. Le cas où $l = 720$ correspond à des coûts constants, puisque notre horizon temporel [7h00,19h00] est de 12 heures = 720 minutes.

3.4 Calcul de la durée d'un plus court chemin

La durée de parcours d'un chemin est obtenue en faisant la somme des durées des tronçons qui le composent. Afin de mieux intégrer les durées de traversée des carrefours, les tronçons ont été définis de telle sorte que les carrefours ne sont pas positionnés aux extrémités des tronçons, mais au milieu des tronçons.

Une propriété importante pour pouvoir calculer efficacement un plus court chemin avec des données dépendantes du temps est la propriété FIFO : la fonction de coût d'un tronçon (i, j) est FIFO si pour tout couple d'heures (t_1, t_2) tel que $t1 < t2$, on a $t_1 + c_{i,j}^{t_1} < t_2 + c_{i,j}^{t_2}$ (autrement dit, on ne peut pas arriver plus tôt sur j en partant plus tard de i). Si les données sont FIFO, alors les plus courts chemins peuvent être calculés en

	n	$\frac{D^-}{D^+}$	B&B		LDS								DP	RDP					
					$d=0.1$		$d=0.2$		$d=0.5$		$d=1.0$			$H=10^3$		$H=10^4$		$H=10^5$	
			$t1$	$t2$	$t1$	%	$t1$	%	$t1$	%	$t1$	%	$t2$	$t1$	%	$t1$	%	$t1$	%
D^+	10		**0.0**	**0.0**	**0.0**	3.9	**0.0**	2.7	**0.0**	0.3	**0.0**	0.0	**0.0**	**0.0**	0.7	**0.0**	0.1	**0.4**	0.0
	20		**46.1**	**111.8**	**0.0**	8.8	**0.1**	5.3	**2.5**	1.3	**39.2**	0.0	**0.4**	**0.0**	12.7	**0.4**	10.7	**12.7**	6.6
	30		-	-	**0.1**	8.6	**4.0**	5.5	**569.4**	1.5	**5230.8**	2.8	**997.8**	**0.1**	16.5	**1.8**	13.1	**83.9**	9.4
	50		-	-	**1.4**	7.7	**22.3**	4.1	**3073.8**	0.4	**5433.7**	3.9	-	**0.1**	14.8	**5.1**	10.8	**241.0**	9.5
D^-	10	6.8	**0.0**	**0.0**	**0.0**	9.5	**0.0**	7.2	**0.0**	7.6	**0.0**	6.9	**0.0**	**0.0**	7.1	**0.0**	6.8	**0.3**	6.8
	20	13.1	**40.7**	**105.8**	**0.0**	20.9	**0.0**	17.2	**0.9**	14.3	**28.1**	13.5	**0.4**	**0.0**	23.6	**0.5**	19.1	**15.8**	15.7
	30	14.4	-	-	**0.1**	21.4	**1.6**	19.1	**325.3**	14.2	**5083.6**	16.4	**998.3**	**0.1**	29.6	**1.9**	25.7	**72.9**	22.7
	50	-	-	-	**1.14**	17.9	**16.1**	15.8	**1849.1**	11.1	**5637.4**	13.5	-	**0.1**	27.7	**5.8**	24.0	**303.0**	20.9

TABLE 1 – Comparaison de B&B, LDS (avec $d \in \{0.1, 0.2, 0.5, 1.0\}$), DP et RDP (avec $H \in \{10^3, 10^4, 10^5\}$), pour $l = 6$min, $n \in \{10, 20, 30, 50\}$ et $D^* \in \{D^+, D^-\}$. $t1$ est le temps pour trouver le meilleur tour et $t2$ le temps pour prouver l'optimalité. Pour DP, $t1 = t2$. Pour LDS et RDP, le meilleur tour trouvé T n'est pas nécessairement optimal et % = *écart*(T, T^{D^+S6}). Les tours optimaux calculés avec D^+ et D^- sont différents et $\frac{D^-}{D^+}$ = *écart*(T^{D^-S6}, T^{D^+S6}). Les temps sont en secondes et '-' indique que le temps est supérieur à 3h.

adaptant l'algorithme de Disjkstra, sinon le problème devient $\mathcal{NP}$-difficile [18].

Si les données réelles de circulation sont naturellement FIFO, le fait de les discrétiser par pas de temps constants peut les rendre non FIFO. Ainsi, nous utilisons les algorithmes de [17,24] pour rendre nos fonctions de coût FIFO avant de calculer les plus courts chemins.

3.5 Description des instances du TD-TSP

Nous considérons quatre tailles $n \in \{10, 20, 30, 50\}$. Pour chaque valeur de n, nous avons généré 30 instances en sélectionnant au hasard pour chaque instance, n adresses géographiques de la ville de Lyon. Pour chaque couple d'adresses (u, v), chaque jeu de donnée D^*Sl (avec $* \in \{+, -\}$ et $l \in \{6, 12, 24, 60, 720\}$), et chaque heure de départ possible t, nous avons calculé la durée de plus court chemin pour aller du u à v en partant de u à l'instant t. Nous considérons que toutes les tournées commencent à $t_0 = 7h00$ et ne peuvent pas terminer après $19h00$. Une durée d'arrêt fixe de 6 (resp. 3) minutes est associée à chaque adresse de livraison si $n \in \{10, 20, 30\}$ (resp. $n = 50$).

4 Évaluation expérimentale

Dans cette section, nous présentons un aperçu des premiers résultats obtenus (qualité des solutions et temps de calcul) en fonction du jeu de données D^*Sl considéré, avec $* \in \{+, -\}$ et $l \in \{6, 12, 24, 60, 720\}$.

Conditions expérimentales. Tous les algorithmes ont été implémentés en `C` et compilés avec `gcc -O3`, à l'exception de l'approche ILP, qui a été codée en `C ++` et a utilisé `Gurobi 8.1.0` [14]. Ce dernier a été exécuté avec ses paramètres par défaut avec un seul thread. Toutes les expérimentations ont été exécutées sur un processeur `Intel (R) Xeon (MD) Platinum 8175M à 2,50 GHz` et `32 Go` de mémoire. Les exécutions sont toutes limitées à trois heures de temps CPU.

Mesure de performance. Nous notons T^{D^*Sl} le tour optimal calculé avec le jeu D^*Sl tel que $* \in \{+, -\}$ et $l \in \{6, 12, 24, 60, 720\}$. Pour pouvoir comparer les durées de tours optimaux calculés avec des jeux de données différents, nous évaluons la durée de chaque tour optimal T^{D^*Sl} en utilisant les données de D^+S6, et nous notons $ts^{D^+S6}(T^{D^*Sl})$ la durée du tour T^{D^*Sl} évaluée avec les données de D^+S6. Cela nous permet de comparer nos tournées dans la même base et avec la meilleure approximation possible des conditions de trafic réelles, puisque D^+ est le jeu le plus précis, et que $l = 6$ min est le plus petit pas de temps considéré.

Considérons par exemple la fonction de coût définie en partie droite de la figure 1 (avec $l = 12$), et supposons que cette fonction de coût corresponde au jeu D^+. Dans ce cas, le meilleur tour est $T^{D^+S12} = \langle 0, 1, 2, 3, 0\rangle$, et sa durée est 14. Si on évalue la durée de ce tour avec la fonction de coût définie en partie gauche (avec $l = 6$), on obtient $ts^{D^+S6}(T^{D^+S12}) = 16$. Le meilleur tour avec $l = 6$ est $T^{D^+S6} = \langle 0, 3, 1, 2, 0\rangle$, et sa durée est $ts^{D^+S6}(T^{D^+S6}) = 12$.

Pour évaluer la qualité d'un tour, nous évaluons son écart en pourcentage au tour optimal sur les données les plus précises, *i.e.*, T^{D^+S6}. L'écart en pourcentage entre deux tours T_1 et T_2 est :

$$\textit{écart}(T_1,T_2) = \frac{ts^{D^+S6}(T_1) - ts^{D^+S6}(T_2)}{ts^{D^+S6}(T_2)} \times 100. \quad (1)$$

Les tours optimaux T^{D^+S6} sont connus pour $n \leq 30$. Pour $n = 50$, aucun algorithme exact ne termine dans la limite de trois heures. Dans ce cas, nous considérons à la place de T^{D^+S6} une solution de référence, qui est la meilleure solution trouvée par toutes les approches dans la limite de 3 heures de temps CPU.

Passage à l'échelle par rapport à n quand $l = 6$mn. Le tableau 1 compare les résultats expérimentaux de B&B, LDS, DP, et RDP quand on considère la plus petite granularité temporelle $l = 6$mn. Le paramètre de divergence d du LDS est choisi selon une approche de mise au point parmi les valeurs de 0.1 à 1.5, avec un intervalle régulier de 0.1. DP prouve l'optimalité en moins d'une seconde (resp. 1000 secondes) pour $n \leq 20$ (resp. $n = 30$). Pour $n = 50$, DP ne peut plus être utilisé à cause de sa complexité spatiale.

B&B est moins performant que DP, et nécessite plus de trois heures pour $n = 30$. En revanche, LDS calcule rapidement de bonnes approximations. Considérons tout d'abord les résultats obtenus avec D^+. Pour $n = 30$ (resp. $n = 50$) et pour une divergence $d = 0.2$, nous obtenons des tours à 5.5% de l'optimum en 4 secondes (resp. à 4.1% du meilleur tour en 22 secondes). Les tours optimaux calculés avec les données spatialement dégradées, D^-, sont 6.8% plus longs que ceux calculés avec les données spatialement complètes quand $n = 10$, et cet écart augmente quand n augmente pour atteindre 14.4% quand $n = 30$. Cela montre l'importance de disposer d'une information spatiale complète pour le modèle prédictif utilisé. Les solutions approchées calculées avec LDS ont des écarts supérieurs, et ces écarts tendent à augmenter quand on diminue d (sauf pour les plus grosses instances). RDP n'est pas compétitif avec LDS : il met plus de temps pour calculer de moins bonnes solutions.

Dans la figure 3, nous comparons l'évolution en fonction du temps de la qualité des solutions calculées par LDS (avec $d \in \{.1, .2, .5, 1\}$) quand $n = 50$. Des solutions de bonne qualité sont obtenues rapidement avec $d = .5$, mais pour de plus longues limites de temps de meilleures solutions sont trouvées avec de plus grandes valeurs de d. A noter que dans la limite de 3 heures, LDS avec $d = 1$ est moins performant que LDS avec $d = .5$ car il met trop de temps à converger.

Impact de la granularité spatio-temporelle. Le TSP statique (quand $l = 720$) est plus facile à résoudre que le TD-TSP, et ILP peut résoudre efficacement des instances avec des centaines de noeuds [28]. Aussi convient-il d'examiner l'intérêt d'optimiser des tours avec des coûts dépendant du temps. En d'autres termes, pouvons-nous trouver de meilleures solutions plus rapidement en optimisant les tours avec des coûts constants ? Plus généralement, quel est l'impact de la granularité de l sur l'évolution de la qualité des tours en fonction du temps de calcul ? Pour répondre à cette question, la figure 4 compare l'évolution de l'écart entre le meilleur tour trouvé et le tour optimal T^{D^+S6} en fonction du temps, pour différentes granu-

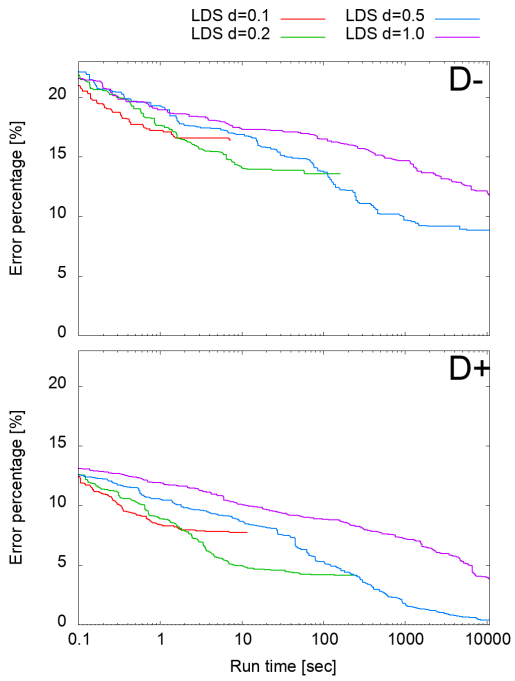

FIGURE 3 – Impact du paramètre d de LDS quand $n = 50$. Pour chaque point (x, y), $y = \textit{écart}(T_x, T^{D^+S6})$ où T_x est le meilleur tour trouvé par LDS en x secondes avec le jeu D^-S6 (en haut) et D^+S6 (en bas).

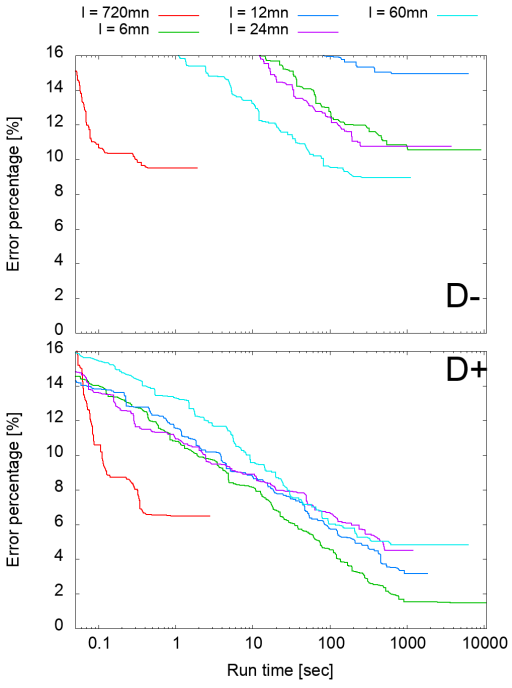

FIGURE 4 – Impact de la longueur l du pas de temps quand $n = 30$. Pour chaque point (x, y), $y = \textit{écart}(T_x, T^{D^+S6})$ où T_x est le meilleur tour trouvé en x secondes avec le jeu D^-Sl (en haut) et D^+Sl (en bas).

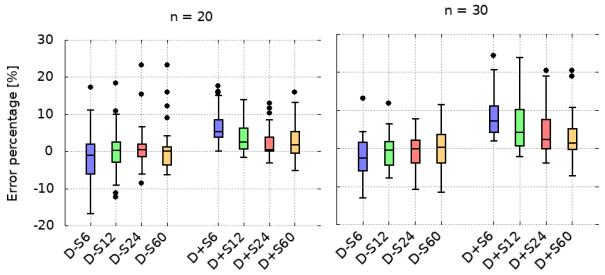

FIGURE 5 – Distribution de *écart*(T^{D^*S720}, T^{D^*Sl}) pour $l \in \{6, 12, 24, 60\}$, $* \in \{-, +\}$, et $n = \{20, 30\}$.

larités $l \in \{6, 12, 24, 60, 720\}$ et pour les deux jeux de données D^+ et D^-. Quand $l = 720$, le TSP est statique et nous utilisons une approche ILP *Branch&Cut* dans laquelle les sous-tours sont éliminés en commençant par ceux ayant la plus petite cardinalité [27]. Cette approche nous permet de trouver la solution optimale en moins de 0.4s en moyenne pour $n = 30$. Quand $l < 720$, nous utilisons l'approche LDS avec $d = 0.5$ qui est l'approche présentant le meilleur compromis entre qualité et temps de résolution.

ILP résolvant très rapidement le problème statique, les tours calculés avec $l = 720$ sont meilleurs que les tours calculés par LDS avec $l < 720$, pour des petites limites de temps. En revanche, pour des limites de temps supérieures à 100s et quand on utilise les données spatialement complètes D^+, les tours optimisés avec $l = 720$ sont 6% plus longs, en moyenne, que ceux optimisés avec $l = 6$. Cela montre l'intérêt d'optimiser avec des coûts dépendants du temps, même si cela rend le problème plus difficile. Cependant, cet intérêt diminue si on augmente l, et il devient nul si on considère les données spatialement incomplètes D^-.

Afin d'examiner plus en détails l'écart entre la solution statique optimale T^{D^*S720}, et celle calculée avec $l \in \{6, 12, 24, 60\}$, *i.e.*, T^{D^*Sl}, nous visualisons la distribution de *écart*(T^{D^*S720},T^{D^*Sl}) sur la figure 5. Les valeurs supérieures à 0 correspondent à des instances où le tour calculé avec des données statiques est moins bon que le tour calculé avec des données dépendantes du temps. Considérons tout d'abord le cas où les données sont spatialement complètes (D^+). Dans ce cas, l'écart médian baisse quand l augmente, mais il est toujours positif, et le fait d'utiliser des données dépendantes du temps peut faire gagner jusqu'à plus de 20% pour certaines instances. Quand $l \geq 12$ l'écart peut devenir négatif pour certaines instances, mais la perte est toujours inférieure à 8%. Quand on compare les distributions des écarts pour $n = 20$ avec celles pour $n = 30$, on remarque que les écarts ont tendance à augmenter quand on augmente n.

En revanche, quand les données sont spatialement incomplètes (D^-), l'écart médian est proche de 0, et pour certaines instances la perte peut être supérieure à 10%. Dans ce cas, il n'est pas intéressant d'optimiser les tours avec des données dépendantes du temps : le problème du TD-TSP est plus difficile à résoudre, et les tours ne sont pas nécessairement meilleurs lorsqu'on les évalue avec les données les plus proches possibles des conditions qui seront observées au moment de la réalisation du tour.

En étudiant la position des capteurs considérés dans le jeu D^- (correspondant aux capteurs effectivement déployés sur le réseau routiers lyonnais), nous constatons que ces capteurs sont souvent placés sur les axes congestionnés du réseau, ce qui conduit à une surestimation des durées de traversée des tronçons non équippés de capteurs (puisque ces durées sont estimées par interpolation par rapport aux tronçons équippés de capteurs, généralement plus congestionnés).

5 Conclusion

Notre étude expérimentale montre à la fois la difficulté et l'intérêt d'évaluer des algorithmes sur des données réelles. Les données réelles de trafic sont généralement très incomplètes car de nombreux tronçons ne sont pas équipés de capteurs (et certains capteurs existants sont mal positionnés par rapport aux feux de circulation de sorte qu'ils ne donnent pas une bonne indication de la vitesse effective). Notons que les données issues de traceurs GPS embarqués dans des véhicules ne sont généralement pas plus complètes. La plateforme SYMUVIA nous a permis d'obtenir des données à la fois réalistes et complètes, et ces données nous ont permis de montrer qu'il est intéressant d'optimiser des tournées avec des données dépendantes du temps dans le cas où les données sont parfaites (*i.e.*, les prévisions sont identiques aux conditions qui seront observées au moment de réaliser la tournée), et dans le cas où on considère une granularité temporelle fine (*i.e.*, des pas de temps de 6 minutes). Dans ce cas, les tournées optimisées avec des données dépendantes du temps sont en moyenne 6.5% plus rapides que celles optimisées avec des données statiques, et le gain peut aller jusqu'à plus de 20% pour certaines instances quand $n = 30$. En revanche, l'intéret diminue lorsque la longueur du pas de temps l augmente, et il devient nul lorsque les données sont spatialement incomplètes et que les données manquantes sont obtenues par interpolation. Nous étudions actuellement l'impact de la position des capteurs sur la qualité des tournées calculées. Notre objectif est de trouver les meilleures positions, permettant de capturer une image plus fidèle des conditions de trafic, et permettant de calculer de meilleures tournées avec les données issues de ces capteurs.

Notre étude expérimentale a également montré que

le TD-TSP est un problème difficile pour lequel les approches ILP et CP ne passent pas à l'échelle dans le cas où il n'y a pas de plages horaires associées aux livraisons. Dans ce cas, l'approche exacte la plus efficace est DP, mais cette approche est limitée aux instances ayant au plus 30 points de livraison. L'approche heuristique LDS permet d'obtenir plus rapidement de meilleures approximations que l'approche RDP, mais il reste encore de la marge pour améliorer ces approches. En particulier, nous étudions des pistes pour améliorer RDP en intégrant une information heuristique au moment de choisir les H états intermédiaires mémorisés à chaque étape. Nous souhaitons également évaluer l'efficacité des approches basées sur la recherche locale, connues pour être efficaces pour le TSP classique.

Nous souhaitons par ailleurs étendre cette étude au TD-TSPTW, pour lequel il y a des plages horaires associées aux livraisons. Pour ce problème, les approches ILP récentes passent bien mieux à l'échelle et il serait intéressant d'évaluer leurs performances sur nos jeux de données.

Références

[1] A. Arigliano, G. Ghiani, A. Grieco et E. Guerriero : Time dependent traveling salesman problem with time windows : Properties and an exact algorithm. 2015.

[2] J. C. Beck et L. Perron : Discrepancy-bounded depth first search. *In CPAIOR*, pages 8–10, 2000.

[3] D. Bergman, A. Ciré, W.-J. van Hoeve et J. N. Hooker : *Decision Diagrams for Optimization*. Artificial Intelligence : Foundations, Theory, and Algorithms. Springer, 2016.

[4] N. Boland, M. Hewitt, D. Vu et M. Savelsbergh : Solving the traveling salesman problem with time windows through dynamically generated time-expanded networks. *In CPAIOR*, pages 254–262. Springer, 2017.

[5] E. Burns, K. Rose et W. Ruml : Best-first search for bounded-depth trees. *In Proceedings of SOCS*. AAAI, 2011.

[6] G. Carpaneto et P. Toth : Primal-dual algrorithms for the assignment problem. *Discrete Applied Mathematics*, 18(2):137 – 153, 1987.

[7] N. Chiabaut, M. Kung, M. Menendez et L. Leclercq : Perimeter control as an alternative to dedicated bus lanes : A case study. *Transportation Research Record*, 2018.

[8] J.-F. Cordeau, G. Ghiani et E. Guerriero : Analysis and branch-and-cut algorithm for the time-dependent travelling salesman problem. *Transportation Science*, 48:46–58, 2014.

[9] A. V. Donati, R. Montemanni, N. Casagrande, A. E. Rizzoli et L. M. Gambardella : Time dependent vehicle routing problem with a multi ant colony system. *EJOR*, 185(3):1174–1191, mar 2008.

[10] H. Kanoh et J. Ochiai : Solving time-dependent traveling salesman problems using ant colony optimization based on predicted traffic. *In DCAI*, volume 151, pages 25–32. Springer, 2012.

[11] A. Stentz G. Ayorkor Korsah et M Bernardine Dias : The dynamic hungarian algorithm for the assignment problem with changing costs. Rapport technique CMU-RI-TR-07-27, Carnegie Mellon University, Pittsburgh, PA, July 2007.

[12] M. Gendreau, G. Ghiani et E. Guerriero : Time-dependent routing problems : A review. *Computers & Operations Research*, 64:189–197, dec 2015.

[13] L. Gouveia et S. Voss : A classification of formulations for the (time-dependent) traveling salesman problem. *EJOR*, 83(1):69–82, may 1995.

[14] LLC Gurobi Optimization : Gurobi optimizer reference manual, 2018.

[15] W. D. Harvey et M. L. Ginsberg : Limited discrepancy search. *In IJCAI*, 1995.

[16] M. Held et R. M. Karp : A dynamic programming approach to sequencing problems. *In Proceedings of the 1961 16th ACM National Meeting*, ACM, pages 71.201–71.204, 1961.

[17] S. Ichoua, M. Gendreau et J-Y Potvin : Vehicle dispatching with time-dependent travel times. *EJOR*, 144(2):379–396, 2003.

[18] D. E. Kaufman et R. L. Smith : Fastest paths in time-dependent networks for intelligent vehicle-highway systems application. *Journal of Intelligent Transportation Systems*, 1:1–11, 1993.

[19] J. A. Laval et L. Leclercq : Microscopic modeling of the relaxation phenomenon using a macroscopic lane-changing model. *Transportation Research Part B : Methodological*, 42(6):511–522, 2008.

[20] L. Leclercq : Bounded acceleration close to fixed and moving bottlenecks. *Transportation Research Part B : Methodological*, 41(3):309–319, 2007.

[21] L. Leclercq, N. Chiabaut et B. Trinquier : Macroscopic fundamental diagrams : A cross-comparison of estimation methods. *Transportation Research Part B : Methodological*, 62:1 – 12, 2014.

[22] C. Malandraki et M. S. Daskin : Time dependent vehicle routing problems : Formulations, properties and heuristic algorithms. *Transportation Science*, 26(3):185–200, 1992.

[23] C. Malandraki et R. B. Dial : A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. *EJOR*, 90(1):45 – 55, 1996.

[24] P. Melgarejo, P. Laborie et C. Solnon : A time-dependent no-overlap constraint : Application to urban delivery problems. *In CPAIOR*, volume 9075 de *LNCS*, pages 1–17. Springer, 2015.

[25] A. Montero, I. Méndez-Díaz et J. Miranda-Bront : An integer programming approach for the time-dependent traveling salesman problem with time windows. *Computers & Operations Research*, 88:280–289, 2017.

[26] G. Newell : A simplified car-following theory : a lower order model. *Transportation Research Part B : Methodological*, 36(3):195–205, 2002.

[27] Staněk R. Pferschy U. : Generating subtour elimination constraints for the tsp from pure integer solutions. *Central European journal of operations research*, 25:231–260, 2017.

[28] R. Roberti et P. Toth : Models and algorithms for the asymmetric traveling salesman problem : an experimental comparison. *EURO J. Transportation and Logistics*, 1(1-2):113–133, 2012.

[29] J. Salotti, S. Fenet, R. Billot, N.-E. El Faouzi et C. Solnon : Comparison of traffic forecasting methods in urban and suburban context. *In ICTAI*. IEEE, 2018.

[30] J. Schneider : The time-dependent traveling salesman problem. *Physica A : Statistical Mechanics and its Applications*, 314(1):151–155, 2002.

[31] G. Stecco, J-F Cordeau et E. Moretti : A tabu search heuristic for a sequence-dependent and time-dependent scheduling problem on a single machine. *J. Scheduling*, 12(1):3–16, 2009.

[32] D. Vu, M. Hewitt, N. Boland et M. Savelsbergh : Solving time dependent traveling salesman problems with time windows, 2018.