# Approche hybride de résolution pour le Time-Dependent Traveling Salesman Problem with Time Windows

Romain Fontaine[1], Christine Solnon[1], Jilles Dibangoye[1]

Université de Lyon, INSA Lyon, CITI, F-69621 Villeurbanne, France
{romain.fontaine,christine.solnon,jilles-steeve.dibangoye}@insa-lyon.fr

**Mots-clés** : *Voyageur de commerce, Fonction time-dependent, Programmation dynamique*

## 1 Introduction

Le *Time Dependent Traveling Salesman Problem* (TD-TSP) est une généralisation du problème du voyageur de commerce où les temps de trajet varient au cours de la journée, ce qui permet de tenir compte du trafic pour planifier des tournées en contexte urbain. Le *TD-TSP with Time-Windows* (TD-TSPTW) généralise ce problème en ajoutant des contraintes sur les heures de passage aux points de livraison. L'importance de considérer des temps de trajet variables dans un contexte urbain a été étudiée dans [7], et il a été montré que cela permet un meilleur respect de ces contraintes. Cette généralisation est donc désirable mais rend le problème plus difficile à résoudre : les approches basées sur la programmation linéaire en nombre entiers [2, 9] ou sur la programmation par contraintes [1] passent mal à l'échelle, tandis que les approches heuristiques sont peu adaptées pour prendre en compte des contraintes supplémentaires telles que des fenêtres temporelles. Dans cet article, nous introduisons une approche hybride pour le TD-TSPTW, basée sur la programmation dynamique, qui a pour ambition de trouver rapidement de bonnes solutions tout en permettant de réaliser des preuves d'optimalité.

## 2 Programmation dynamique pour le TD-TSPTW

La programmation dynamique permet de résoudre le problème du voyageur de commerce en exploitant les équations de Bellman suivantes [3] :

$$p(i,\mathcal{S}) = \begin{cases} c_{v_0 i} & \text{si } \mathcal{S} = \emptyset \\ min_{j \in \mathcal{S}}\ p(j, \mathcal{S} \setminus \{j\}) + c_{ji} & \text{sinon.} \end{cases}$$

où $c_{ij}$ dénote la durée pour aller de $i$ à $j$, et où le sous-problème $p(i,\mathcal{S})$ correspond à la durée du plus court chemin partant du sommet initial $v_0$, visitant chaque sommet de $\mathcal{S}$ une fois, et arrivant à $i$. La solution est obtenue en calculant $p(v_0,\mathcal{P})$, où $\mathcal{P}$ représente l'ensemble des sommets à visiter. Ces équations se généralisent facilement à différentes variantes du TSP comportant, par exemple, des contraintes de fenêtres temporelles, de précédence ou de capacité [8]. Elles s'étendent aussi facilement au cas time-dependent [6] : il suffit de remplacer $c_{v_0 i}$ par $c^{t_0}_{v_0 i}$ dans la première équation et $c_{ji}$ par $c^{p(j,\mathcal{S}\setminus\{j\})}_{ji}$ dans la seconde équation (où $t_0$ est l'instant de départ du sommet initial $v_0$ et $c^t_{ij}$ dénote la durée pour aller de $i$ à $j$ quand on part de $i$ à l'instant $t$). Cette approche exacte a une complexité en temps en $\mathcal{O}(n^2 \cdot 2^n)$ et une complexité en mémoire en $\mathcal{O}(n \cdot 2^n)$ (où $n$ est le nombre de sommets). Elle ne permet donc pas de résoudre des instances de plus de 30 sommets sans fenêtres temporelles en un temps raisonnable.

## 3 Description de l'approche hybride

Nous introduisons une approche hybride dans le but de profiter des avantages de la programmation dynamique tout en réduisant ses inconvénients. Cette approche a été développée

à partir d'algorithmes de recherche arborescente (dits "anytime" dans [5]) qui explorent l'espace des sous-problèmes de manière progressive et informée : des heuristiques admissibles sont utilisées afin éviter d'explorer les sous-problèmes ne contenant pas de solutions ainsi que pour explorer en premier les sous-problèmes les plus prometteurs. Afin d'accélérer la convergence, cette approche est combinée avec une procédure de recherche locale, visant à améliorer les solutions trouvées, et une procédure de propagation des contraintes de fenêtres temporelles, visant à réduire l'espace de recherche en inférant des contraintes de précédence. Cette approche permet d'obtenir relativement rapidement des solutions approchées, et est capable de prouver l'optimalité de certaines instances. Plusieurs heuristiques ont été implémentées afin d'obtenir différents compromis entre vitesse de convergence et quantité de mémoire utilisée. Certaines heuristiques ont aussi été implémentées de manière parallèle afin d'accélérer la résolution.

Les premiers résultats montrent que cette approche est compétitive par rapport à l'approche de [9], basée sur la programmation linéaire en nombres entiers (ILP), dans le cas de la fonction objectif *makespan*. En effet, elle permet de prouver l'optimalité pour l'ensemble des instances du benchmark - dont certaines instances ouvertes - et ce plus rapidement que l'approche ILP. Les instances de ce benchmark ont des fenêtres temporelles de largeur variable mais relativement courtes, et comportent jusqu'à 100 sommets. Nous avons également montré que notre approche est plus efficace que l'approche générique basée sur la programmation dynamique introduite dans [4] dans le cas où les temps de trajet sont constants (TSPTW).

# 4 Conclusion

Ces résultats montrent la pertinence de notre approche pour résoudre le TD-TSPTW, tout en proposant divers compromis entre vitesse de convergence et capacité à prouver l'optimalité. Il sera nécessaire de comparer cette approche à l'état de l'art en utilisant d'autres instances - notamment lorsque les fenêtres temporelles sont plus larges et qu'il est ainsi plus difficile de prouver l'optimalité - ou encore en utilisant des benchmarks plus réalistes, comme celui présenté dans [7]. Les principes sur lesquels se base cette approche peuvent être réutilisés pour résoudre d'autres problèmes plus généraux comme par exemple le *Vehicle Routing problem*, le *Pickup and Delivery Problem*, où encore le *Dial-A-Ride Problem*.

# Références


[1] P. Aguiar-melgarejo. *A Constraint Programming Approach for the Time Dependent Traveling Salesman Problem*. PhD thesis, INSA Lyon, 2018.

[2] A. Arigliano, G. Ghiani, A. Grieco, E. Guerriero, and I. Plana. Time-dependent asymmetric traveling salesman problem with time windows : Properties and an exact algorithm. *Discrete Applied Mathematics*, 261 :28–39, may 2019.

[3] R. Bellman. Dynamic Programming Treatment of the Travelling Salesman Problem. *Journal of the ACM (JACM)*, 9(1) :61–63, 1962.

[4] X. Gillard, V. Coppé, P. Schaus, and A. A. Cire. Improving the Filtering of Branch-and-Bound MDD Solver. *CPAIOR*, 12735 LNCS :231–247, 2021.

[5] L. Libralesso. *Anytime tree search algorithms for combinatorial optimization*. PhD thesis, Université Grenoble Alpes, 2020.

[6] C. Malandraki and R. B. Dial. A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. *European Journal of Operational Research*, 90(1) :45–55, 1996.

[7] O. Rifki, N. Chiabaut, and C. Solnon. On the impact of spatio-temporal granularity of traffic conditions on the quality of pickup and delivery optimal tours. *Transportation Research Part E : Logistics and Transportation Review*, 142, 2020.

[8] J. J. van Hoorn. *Dynamic Programming for Routing and Scheduling : Optimizing Sequences of Decisions*. PhD thesis, Amsterdam : Vrije Universiteit, 2016.

[9] D. M. Vu, M. Hewitt, N. Boland, and M. Savelsbergh. Dynamic Discretization Discovery for Solving the Time-Dependent Traveling Salesman Problem with Time Windows. *Transportation Science*, 54(3) :703–720, 2020.