Variables de Séquence pour les problèmes de tournée de véhicules

A Delecluse, Pierre Schaus, Pascal van Hentenryck

Variables de Séquence pour les problèmes de tournée de véhicules

A. Delecluse[1,2], P. Schaus[1], P. Van Hentenryck[3]

[1] ICTEAM, UCLouvain, Belgique
[2] TRAIL, Belgique
[3] Georgia Institute of Technology, USA

6 mai 2022

Résumé

Nous proposons une variable ciblant les problèmes de tournées de véhicules : la variable de séquence. La représentation de son domaine permet une recherche sur base d'insertions dans une route partiellement formée, ainsi que l'implémentation d'algorithmes simples, mais puissants, permettant de respecter des temps de transition entre les visites ou des capacités dans un véhicule. Nos expériences démontrent que cette variable est suffisamment flexible pour modéliser des problèmes d'itinéraires très contraints, tout en les résolvant de manière efficace.

Mots-clés

Variable de séquence, Voyageur de commerce, livraison, transport de patients, programmation par contraintes.

1 Introduction

Les problèmes de tournées de véhicules (PTV) [19] apparaissent fréquemment dans la distribution de biens dans la chaîne logistique. De par l'augmentation de l'urbanisation et des défis écologiques, il est également attendu que les offres de transport flexibles, telles que le transport à la demande, soient davantage développées à l'avenir. Cela soulève de nouveaux enjeux pour l'optimisation, en particulier le développement d'outils génériques et réutilisables pour les nombreux contextes et variants du PTV.
La programmation par contraintes (PPC) est une des approches les plus flexibles pour la modélisation de PTV. L'approche standard consiste en un modèle de successeurs, introduisant une variable par lieu visité qui représente la visite suivante dans le parcours d'un véhicule. En dépit de sa simplicité, ce modèle souffre de deux limitations majeures : l'impossibilité de représenter des visites optionnelles sans ajouts de valeurs spéciales ainsi que le rajout d'une visite au milieu d'un itinéraire partiellement formé. Le but de la variable de séquence est de pallier à ces deux limitations :

1. Elle peut facilement modéliser l'exclusion de visites, similairement à une variable d'ensemble.
2. Inspirée de l'idée du graphe d'insertion [3], elle permet l'ajout d'une visite au milieu d'un itinéraire partiellement formé, permettant l'emploi d'algorithmes de recherche d'insertions en profondeur [3, 8] pour pouvoir réinsérer de manière optimale un ensemble de visites relaxées dans une recherche en large voisinage (RLV).

Le premier auteur est un doctorant

Nous commençons par un survol de travaux antérieurs sur approches avec séquences, avant de détailler notre variable, quelques contraintes applicables sur son domaine et son emploi sur 3 variants de PTV.

2 Travaux antérieurs

Dans [18], les auteurs ont introduit une variable de séquence pour des problèmes d'horaires et de PTV. La représentation du domaine de cette variable étend directement celle du sous-ensemble lié pour des variables d'ensemble [7], de part une partition des visites en requises, possibles et exclues, ainsi qu'une séquence partielle et un ensemble d'insertion
Bien que non publié, IBM ILOG CP Optimizer [11] dispose également de variables de séquences pour décider l'ordre de visites, davantage axé sur la planification mais néanmoins utilisé pour des problèmes de tournées de vehicules. Leurs fonctionalités et contraintes sont brièvement décrites [9, 10] sans pour autant donner leur implémentation exacte. Selon leur Interface de Programmation [5, 6], elles se basent sur une structure tête-queue, maintenant séparément l'agrandissement de la tête et de la queue pour rajouter des variables d'intervalle au début ou à la fin de la séquence, respectivement. Cette implémentation semble similaire à celle de Google OR-Tools [15] et ses propres variables de séquences [16]. Elle a été employée pour résoudre le problème du transport de patients dans [4] et [12].

3 Variable de séquence

Nous introduisons les notations sur les séquences avant de formaliser le domaine de notre variable et décrire comment l'implémenter dans un solutionneur de PPC. Notre variable se base essentiellement sur celle de [18] mais en y enlevant le set requis. Par conséquent, une visite possible doit être directement planifiée à un endroit précis dans une séquence partiellement formée, et ne peut pas être simplement requise. Cette modification, en dépit de sa simplicité, permet de simplifier grandement le raisonnement fait par les contraintes, leur complexité temporelle ainsi que l'implémentation d'heuristiques, tout en perdant relativement peu

de flexibilité en pratique. La variable que nous proposons peut être vue comme une généralisation de l'idée du *graphe d'insertion* [3], plus générique et encapsulée par l'implémentation interne du domaine de la variable de séquence.

3.1 Notations

Les notations sont largement issues de [18] mais réintroduites par souci de clarté. Chaque lieu pouvant être visité est qualifié de *nœud*, et leur ensemble est décrit par $\mathcal{X}$. Une séquence sur $\mathcal{X}$ est notée $\overrightarrow{S}$ et l'ensemble de toutes les séquences de $\mathcal{X}$ par $\overrightarrow{\mathcal{P}}(\mathcal{X})$. La notation $p \overset{\overrightarrow{S}}{\prec} q$ signifie que le nœud p précède q dans $\overrightarrow{S}$, et $p \xrightarrow{\overrightarrow{S}} q$ que p précède directement q dans $\overrightarrow{S}$. Elles seront simplement écrites $p \prec q$ et $p \rightarrow q$ lorsque le contexte le permet.

Une séquence peut être agrandie par l'utilisation d'un opérateur $insertion(\overrightarrow{S}, p, q)$, avec $q \notin S, p \in S$, qui résulte en l'insertion de q juste après p dans la séquence. Plus formellement, assumons $\overrightarrow{S} = \overrightarrow{S}_1 \cdot p \cdot \overrightarrow{S}_2$. La super-séquence résultante de l'opération est $\overrightarrow{S}' = \overrightarrow{S}_1 \cdot p \cdot q \cdot \overrightarrow{S}_2$. Cette opération est également notée $\overrightarrow{S} \underset{(q,p)}{\Longrightarrow} \overrightarrow{S}'$.

Étant donné I, un ensemble de tuples (q, p), correspondant chacun à une insertion potentielle dans $\overrightarrow{S}$, $\overrightarrow{S} \underset{I}{\Longrightarrow} \overrightarrow{S}'$ signifie que $\overrightarrow{S}'$ peut être obtenue en appliquant une insertion de I sur $\overrightarrow{S} : \exists (p, q) \in I \mid \overrightarrow{S} \underset{(p,q)}{\Longrightarrow} \overrightarrow{S}'$.

Plus généralement, la *dérivation en zéro étape ou plus* est définie par $\overrightarrow{S} \overset{*}{\underset{I}{\Longrightarrow}} \overrightarrow{S}' \equiv \overrightarrow{S} = \overrightarrow{S}' \vee \left(\exists (p, q) \in I \mid \overrightarrow{S} \underset{(p,q)}{\Longrightarrow} \overrightarrow{S}'' \wedge \overrightarrow{S}'' \overset{*}{\underset{I \setminus \{(p,q)\}}{\Longrightarrow}} \overrightarrow{S}' \right)$. Notons que I peut contenir des tuples qui ne correspondent pas à une insertion possible dans, $\overrightarrow{S}$ mais à la place dans une insertion possible dans une super séquence de $\overrightarrow{S}$.

3.2 Domaine

Définition 1. *Le domaine d'une variable de séquence Sq est représenté par $\langle \overrightarrow{S}, I, P, E \rangle$, avec $\overrightarrow{S}$ un ensemble de nœuds ordonnancés correspondant à la séquence et formant un tour partiel, des points d'insertions $I \subseteq \mathcal{X} \times \mathcal{X}$ et deux sous-ensembles de nœuds $P, E \subseteq \mathcal{X}$ pour les nœuds potentiellement insérés et exclus de la séquence, respectivement. Le domaine de Sq, également noté $D(Sq)$, est défini comme $\langle \overrightarrow{S}, I, P, E \rangle \equiv \left\{ \overrightarrow{S}' \in \overrightarrow{\mathcal{P}}(P \cup S) \mid \overrightarrow{S} \overset{*}{\underset{I}{\Longrightarrow}} \overrightarrow{S}' \right\}$ et capture tous les dérivations possibles valides du tour partiel $\overrightarrow{S}$ en utilisant les insertions de I.*

À son initialisation, la séquence est composée d'un tour partiel de deux nœuds, $\alpha \cdot \omega$, pour le début α et la fin ω du parcours, et aucune insertion n'est permise après ω pour garantir que ω demeure le dernier nœud visité. P est donc égal à $\mathcal{X} \setminus \{\alpha, \omega\}$, $E = \phi$ et l'ensemble d'insertions est $I = \{(p, q) \in P \times \mathcal{X} \mid p \neq \omega\}$. Imposer un nœud de début et de fin dans la séquence permet une modélisation facile des problèmes où le trajet d'un véhicule doit finir à son point de départ ou à un autre endroit (α se trouve au même endroit que ω ou non) et évite à l'interface de programmation de faire face au cas spécial de séquences vides, requérant l'ajout d'un symbole fictif comme dans [18].

Nous maintenons une cohérence assez faible sur le domaine, facile à calculer et capturant les invariants suivants :

$$S \cup P \cup E = X \wedge S \cap P = S \cap E = P \cap E = \phi \quad (1)$$

$$\forall (p, q) \in I : q \in P \wedge p \notin E \quad (2)$$

$$\forall q \in P : \exists p \in S \cup P \mid (p, q) \in I \quad (3)$$

(1) Les nœuds de la séquence partielle S, de l'ensemble possible P et de l'ensemble exclu E forment une partition de $\mathcal{X}$; (2) les insertions valides sont constituées de nœuds possibles placés après des nœuds non exclus (qui ne sont pas forcément encore présents dans la séquence partielle) ; (3) un nœud possible peut toujours être inséré après un autre mœud. Cette cohérence ne détecte pas si tous les arcs de I sont déconnectés de $\overrightarrow{S}$ et devraient être exclus.

3.3 Implémentation et structures de données

L'implémentation du domaine $\langle \overrightarrow{S}, I, P, E \rangle$ doit être réversible pour des solutionneurs à trace comme MiniCP [14], et ses mises à jours et itérations aussi efficaces que possible.

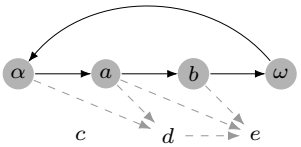
Le partitionnement entre les ensembles S, P, E est implémenté par un seul ensemble réversible [17], permettant la suppression et restauration en temps constant. Les points d'insertions I sont quant à eux partitionnés en un ensemble $I^x = \{p \in (S \cup P) : (p, x) \in I\}$ par nœud $x \in \mathcal{X}$, chacun composé des prédécesseurs valides du nœud x. Ils sont également implémentés par des ensembles réversibles. Les successeurs des nœuds sont quant eux accessibles et modifiables en tant constant via un tableau d'entiers réversibles. Les invariants maintenus par ces structures de données sont décrits dans les équations (4) à (7) et sont équivalents aux invariants (1) à (3). La Figure 1 illustre le domaine.

$$S \cup P \cup E = X \wedge S \cap P = S \cap E = P \cap E = \phi \quad (4)$$

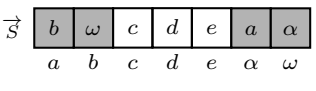
$$p \in E \implies I^p = \phi \wedge \forall x : p \notin I^x \quad (5)$$

$$p \in S \implies I^p = \phi \quad (6)$$

$$I^x = \phi \implies x \in S \vee x \in E \quad (7)$$



node x	I^x	$\lvert I^x \cap S \rvert$	$\lvert I^x \cap P \rvert$
a	ϕ	0	0
b	ϕ	0	0
c	ϕ	0	0
d	$\{\alpha, a\}$	2	0
e	$\{a, b, d\}$	2	1



$P = \{d, e\}, E = \{c\}$

FIGURE 1 – Variable de séquence, avec son ordonnancement $\overrightarrow{S} = \{\alpha, a, b, \omega\}$ et ses insertions potentielles (pointillées) pour les nœuds $x \in P$. Se trouve une table montrant les insertions des nœuds, les successeurs de la séquence (uniquement valide pour les nœuds $\in S$) ainsi que l'appartenance des nœuds à P et E.

4 Contraintes globales

Nous énumérons simplement quelques contraintes essentielles au PTV sans pour autant donner leur implémentation exacte, afin de conserver une taille raisonnable pour ce résumé. Les lecteurs souhaitant plus de détails peuvent se référer directement à l'implémentation [1] ou dans le futur à la version longue du papier, qui est en attente de relecture pour une autre conférence. Les contraintes globales que nous avons définies sont des contraintes de

Dépendance s'assurant qu'un ensemble de nœuds soient tous présents ou absents d'une séquence ;

Précédence veillant à ce qu'un ordonnancement précis de nœuds soit respecté dans la séquence ;

Disjointe garantissant qu'un nœud ne soit visité que par une seule séquence parmi un ensemble ;

Cumulative respectant une capacité maximale disponible, très similaire à l'approche de [18].

TempsDeTransition. Permet d'ajouter une dimension temporelle aux PTV. Elle est appliquée sur une séquence, une matrice de distance $tr \in R^{n \times n}$, une distance de parcours l et associe à chaque nœud $x \in \mathcal{X}$ une fenêtre de temps $début_x$ durant laquelle la visite du nœud doit se produire ainsi qu'une durée $durée_x$ pour y effectuer une action :

$$\texttt{TempsDeTransition}(Sq, [début], [durée], [[tr]], l) \equiv \left\{ \overrightarrow{S} \in D(Sq) \;\middle|\; \begin{pmatrix} \forall i,j \in \overrightarrow{S}, i \overset{\overrightarrow{S}}{\prec} j \implies \\ début_i + durée_i + tr_{i,j} \leq début_j \\ l = \sum_{i,j \in \overrightarrow{S} \mid i \to j} tr_{i,j} \end{pmatrix} \right\} \tag{8}$$

Nous considérons qu'il est possible d'atteindre un nœud avant le début de sa fenêtre de temps sans y commencer la tâche associée, expliquant l'emploi d'inégalités dans (8).

Filtrage Le pseudo-code pour le filtrage est présenté dans l'Algorithme 2. Nous commençons par mettre à jour les fenêtres de temps *début* des nœuds visités actuellement (ligne 2). Par la suite, nous calculons la longueur actuelle de la séquence et enlevons les insertions invalides : un prédécesseur qui ne permet pas d'atteindre le nœud dans sa fenêtre de temps (ligne 8), de rejoindre le successeur actuel dans la séquence (ligne 13) ou qui excéderait la longueur maximale permise (ligne 17). La complexité de ce filtrage est $\mathcal{O}(|P| \cdot |S|)$. En pratique, le filtrage s'exécute plus rapidement, car $I^x \cap S$ est récupéré en $\Theta(\min(|S|, |I^x|))$ dans notre implémentation. Comme nous ne raisonnons pas sur un ensemble de nœuds requis comme dans [18], nous nous débarrassons du problème NP-complet consistant à vérifier si un chemin valide visitant tous les nœuds requis existe.

Algorithme 1 : `TempsDeTransition`$(Sq = \langle \overrightarrow{S} I, P, E \rangle, [début], [durée], [[tr]], l)$

```
1  for i ∈ S⃗ do
2  |  mise à jour de la fenêtre de temps début_i
3  longueur ← distance actuelle de la séquence
4  min(l) ← longueur
5  for x ∈ P do
6  |  for p ∈ I^x ∩ S do
7  |  |  arr_x ← min(début_p) + durée_p + tr_{p,x}
8  |  |  if arrivée_x > max(début_x) then
9  |  |  |  enlève p de I^x
10 |  |  else
11 |  |  |  q ← succ(Sq, p)
12 |  |  |  arr_q ← max(arr_x, min(début_x)) +
   |  |  |    durée_x + tr_{x,q}
13 |  |  |  if arr_q > max(début_q) then
14 |  |  |  |  enlève p de I^x
15 |  |  |  else
16 |  |  |  |  détour ← tr_{p,x} + tr_{x,q} − tr_{p,q}
17 |  |  |  |  if detour + longueur > l then
18 |  |  |  |  |  enlève p de I^x
```

5 Expériences

Notre variable permet de modéliser des problèmes de collecte et livraison (PCL), problèmes de transport de patients (PTP), ou encore le voyageur de commerce avec fenêtre de temps (PVCFT). Chacun de ces problèmes est modélisé avec une Variable de Séquence par véhicule ainsi qu'une contrainte `TempsDeTransition` sur ces variables. Dans le cadre du PCL et du PTP, une contrainte `Cumulative` est employée pour respecter la capacité maximale des véhicules.

Nos résultats avec une recherche par large voisinage sont présentés dans la Table 1 pour le PCL, la Table 2 pour le PTP et la Table 3 pour le PVCFT. Pour le PCL, nos résultats sont compétitifs avec l'état de l'art [8], avec un léger avantage sur les plus petites instances. Nos variables sont par ailleurs toujours plus performantes que celles de CP Optimizer (modèle issu de [18]), qui ne parvient pas toujours à fournir de solution améliorante. Pour le PTP, nous arrivons à améliorer les meilleurs résultats publiés[4]. Enfin, pour le PVCFT, notre implémentation est à même de battre l'état de l'art et trouver de nouvelles meilleures solutions pour le problème sur 32 instances issues de la suite standard de tests [13]. Nous ne montrons toutefois que les 10 nouvelles solutions sur le jeu d'instances *AFG* [2].

classe a		LNS-FFPA		Séquence		CPO	
m	n	Moyenne	Meilleur	Moyenne	Meilleur	Moyenne	Meilleur
3	24	191.76	191.40	**190.89**	**190.21**	196.11	196.00
4	36	**291.71**	**291.71**	294.72	292.72	318.97	318.97
5	48	308.95	306.97	**307.09**	**304.38**	327.37	327.00
6	72	532.55	524.97	**531.84**	**519.76**	579.79	579.77
7	72	**554.57**	550.42	554.65	**548.72**	614.02	614.00
8	108	**752.29**	**742.08**	794.86	755.00	924.04	923.86
9	96	**622.19**	614.65	625.68	**611.15**	740.26	740.26
10	144	**950.16**	**929.31**	1011.42	962.21	/	/
11	120	**699.32**	**687.99**	718.58	709.49	861.74	861.73
13	144	**878.33**	**864.81**	901.71	874.56	1042.82	1042.82
Moyenne		**578.18**	**570.43**	593.14	576.82	t/o	t/o

TABLE 1 – Valeurs objectives pour le PCL, lorsqu'une solution initiale est fournie. L'état de l'art (LNS-FFPA) et un modèle employant les variables de CP Optimizer (CPO) sont indiqués. / signifie qu'aucune solution améliorante n'a été trouvée. Dix essais ont été faits par instance et modèle.

Instances					SCHED+MSS	Séquence
Difficulté	Nom	$\|H\|$	$\|V\|$	$\|R\|$	Sol	Sol
Facile	RAND-E-8	32	12	128	**128**	**128**
Facile	RAND-E-9	36	14	144	142	**143**
Facile	RAND-E-10	40	16	160	**157**	156
Moyenne	RAND-M-8	64	8	128	83	**91**
Moyenne	RAND-M-9	72	8	144	81	**93**
Moyenne	RAND-M-10	80	9	160	99	**113**
Difficile	RAND-H-8	128	8	128	75	**87**
Difficile	RAND-H-9	144	8	144	72	**84**
Difficile	RAND-H-10	160	8	160	72	**84**

TABLE 2 – Résultats expérimentaux pour le PTP. $|H|$, $|V|$, $|R|$ sont le nombre d'hôpitaux, véhicules et requêtes, respectivement. L'objectif est le nombre de patients servis (Sol). SCHED+MSS est le meilleur modèle référencé [4]

Instance	Précédent	Nouvel	Temps [s]
rbg132.2	8200	8194	37.76
rbg132	8470	8468	0.76
rbg201a	12 967	12 948	152.53
rbg233.2	14 549	14 523	24.20
rbg092a	7160	7158	2.70
rbg152.3	9797	9796	0.41
rbg193.2	12 167	12 159	242.54
rbg193	12 547	12 538	55.57
rbg233	15 031	14 994	264.70
rbg172a	10 961	10 956	113.83

TABLE 3 – Solutions au PVCFT sur les instances AFG [2].

6 Conclusion

Nous avons présenté une version simplifiée de la Variable de Séquence introduite précédemment [18], une approche flexible pour la modélisation et la résolution de PTV. Nous avons détaillé son domaine et expliqué les contraintes globales qui peuvent y être appliquées. Nos résultats expérimentaux sur trois PTV montrent que nous sommes compétitifs avec les travaux similaires sur des variables de séquences, tout en étant suffisamment efficace pour trouver de nouvelles solutions améliorantes à un problème très étudié tel que le PVCFT. Nos prochains travaux sur cette approche se focaliseront sur son application à d'autres PTV, à de la planification ainsi qu'à l'amélioration de nos algorithmes.

Références

[1] *MiniCP Sequences - Anonymous GitHub*, Sep 2021. `https://anonymous.4open.science/r/minicp-sequences-5EE3/README.md`, [Online ; accessed 26. Feb. 2022].

[2] Ascheuer, Norbert: *Hamiltonian path problems in the online optimization of flexible manufacturing systems*. Thèse de doctorat, 1996.

[3] Bent, Russell et Pascal Van Hentenryck: *A two-stage hybrid local search for the vehicle routing problem with time windows*. Transportation Science, 38(4) :515–530, 2004.

[4] Cappart, Quentin, Charles Thomas, Pierre Schaus et Louis Martin Rousseau: *A Constraint Programming Approach for Solving Patient Transportation Problems*. Dans Hooker, John (rédacteur) : *Principles and Practice of Constraint Programming*, pages 490–506, Cham, 2018. Springer International Publishing, ISBN 978-3-319-98334-9.

[5] Center, IBM Knowledge: *Interval variable sequencing in CP Optimizer*, Mar 2021. `https://www.ibm.com/docs/en/icos/12.9.0?topic=concepts-interval-variable-sequencing-in-cp-optimizer`, [Online ; accessed 13. Jan. 2022].

[6] Center, IBM Knowledge: *Search API for scheduling in CP Optimizer*, Mar 2021. `https://www.ibm.com/docs/en/icos/12.9.0?topic=c-search-api-scheduling-in-cp-optimizer#85`, [Online ; accessed 13. Jan. 2022].

[7] Gervet, Carmen: *Interval Propagation to Reason about Sets : Definition and Implementation of a Practical Language*. Constraints, 1 :191–244, mars 1997.

[8] Jain, Siddhartha et Pascal Van Hentenryck: *Large neighborhood search for dial-a-ride problems*. Dans *International Conference on Principles and Practice of Constraint Programming*, pages 400–413. Springer, 2011.

[9] Laborie, Philippe et Jerome Rogerie: *Reasoning with Conditional Time-Intervals.* Dans *FLAIRS conference*, pages 555–560, 2008.

[10] Laborie, Philippe, Jerome Rogerie, Paul Shaw et Petr Vilím: *Reasoning with Conditional Time-Intervals. Part II : An Algebraical Model for Resources*. Dans *FLAIRS Conference*, 2009.

[11] Laborie, Philippe, Jérôme Rogerie, Paul Shaw et Petr Vilím: *IBM ILOG CP Optimizer for Scheduling*. Constraints, 23(2) :210–250, apr 2018, ISSN 1383-7133. `https://doi.org/10.1007/s10601-018-9281-x`.

[12] Liu, Chang, Dionne M. Aleman et J. Christopher Beck: *Modelling and Solving the Senior Transportation Problem*. Dans Hoeve, Willem Jan van (rédacteur) : *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 412–428, Cham, 2018. Springer International Publishing, ISBN 978-3-319-93031-2.

[13] López-Ibáñez, Manuel: *Instances for the TSPTW*, Sep 2020. `https://lopez-ibanez.eu/tsptw-instances`, [Online ; accessed 15. Feb. 2022].

[14] Michel, L., P. Schaus et P. Van Hentenryck: *MiniCP : a lightweight solver for constraint programming*. Mathematical Programming Computation, 13(1) :133–184, 2021. `https://doi.org/10.1007/s12532-020-00190-7`.

[15] Perron, Laurent et Vincent Furnon: *OR-Tools*. `https://developers.google.com/optimization/`.

[16] Perron, Laurent et Vincent Furnon: *OR-Tools Sequence Var*. `https://developers.google.com/optimization/reference/constraint_solver/constraint_solver/SequenceVar`.

[17] Saint-Marcq, Vianney le Clément de, Pierre Schaus, Christine Solnon et Christophe Lecoutre: *Sparse-sets for domain implementation*. Dans *CP workshop on Techniques foR Implementing Constraint programming Systems (TRICS)*, pages 1–10, 2013.

[18] Thomas, Charles, Roger Kameugne et Pierre Schaus: *Insertion Sequence Variables for Hybrid Routing and Scheduling Problems*. Dans *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 457–474. Springer, 2020.

[19] Toth, Paolo et Daniele Vigo: *The vehicle routing problem*. SIAM, 2002.