Programmation par contraintes pour la vendange sélective

Nicolas Briot, Christian Bessiere, Philippe Vismara

Programmation par contraintes pour la vendange sélective

Nicolas Briot[1] **Christian Bessiere**[1] **Philippe Vismara**[1,2]

[1] LIRMM, Université de Montpellier, 161 rue Ada, 34095 Montpellier, France
[2] MISTEA, Montpellier SupAgro, INRA, 2 place Viala, 34060 Montpellier, France
{briot,bessiere,vismara}@lirmm.fr

Résumé

En viticulture de précision, la vendange sélective consiste à récolter séparément deux qualités de raisin dans une même vigne. On dispose pour cela d'une machine à vendanger comportant deux trémies et on connait la localisation des zones de qualités ainsi qu'une estimation des quantités à récolter. Le problème consiste à optimiser le trajet de la machine à vendanger tout en respectant de nombreuses contraintes comme la prise en compte du sens de traitement des rangs, la capacité de stockage de la machine, son rayon de braquage, le nombre de vidanges, etc. Après une formalisation du problème de la vendange sélective, cet article présente une modélisation sous la forme d'un problème de satisfaction de contraintes. La dernière section donne des résultats préliminaires fournis par le solveur AbsCon.

Abstract

In precision viticulture, the problem of differential harvesting occurs on vineyard with two qualities of grapes. Given estimated areas of different quality and quantity, the objective is to optimize the routing of the grape harvester under several constraints. This constraints are : harvest a certain amount of first quality grapes, capacity of the machine, turning radius, number of emptying, etc. After describing the problem of differential harvest, the paper models it as a constraint satisfaction problem. The last section is devoted to preliminary results with AbsCon solver.

1 Introduction

En viticulture de précision, de nombreuses études ont proposé de définir des zones de qualité intra-parcellaire [11]. Ces zones peuvent être déterminées à partir de photos aériennes, de capteurs en champ, de prélèvements ou des récoltes antérieures. Prendre en compte ces zones de qualité intra-parcellaire permet d'optimiser le coût et la qualité des vendanges. Cette approche a justifié le récent développement de prototypes de machine à vendanger capables de gérer deux qualités différentes de grappes, comme le prototype EnoControlTM system (NewHolland Agriculture, PA, USA). Ce type de machine possède deux réservoirs appelés *trémies* qui permet de stocker deux qualités différentes lors d'une même récolte.

Optimiser la récolte consiste à minimiser le temps de travail de la vendangeuse, ce qui correspond aux temps de trajets et aux temps de vidanges de la machine. Idéalement, le but à atteindre est de remplir les deux trémies de manière équilibrée afin de vidanger le moins possible la machine à vendanger. En effet, suivant la répartition des zones de qualité dans le vignoble et le rendement des vignes, la trémie qui contiendra une certaine qualité peut se remplir plus vite que l'autre. En plus des contraintes de capacité, il faut prendre en compte le rayon de braquage de la machine : plus le saut de rangs est petit, plus la machine doit manoeuvrer et perd du temps. Toutes ces contraintes rendent le problème combinatoire et complexe à résoudre. Des problèmes semblables, comme le problème du voyageur de commerce [1, 7] ou les problèmes de tournées de véhicules ont déja pu être traités. En programmation par contraintes, une étude de problèmes de tournées de véhicules a ainsi été étudiée en [8]. D'autres approches moins génériques, comme la recherche opérationnelle, ont été testées notamment en agriculture [2, 3]. Toutefois, le problème de la vendange sélective ne rentre dans aucun des problèmes déjà étudiés et reste inédit. Devant la multitude des contraintes et les différentes variantes possibles du modèle, l'utilisation de la programmation par contraintes semble pertinente afin de modéliser et de résoudre le problème.

Après une formalisation du problème de la vendange sélective, ce papier présente une modélisation de celui-ci sous la forme d'un problème d'optimisation de contraintes. Enfin, des résultats préliminaires fournis par le solveur AbsCon sont présentés en dernière partie.

2 Le problème de la vendange sélective

Le problème de la vendange sélective (ou *Differential Harvest Problem*) se pose sur un vignoble composé de plusieurs zones classées en deux catégories : les *A-zones* qui contiennent les *A-grappes*, c'est-à-dire les raisins de qualité supérieure et les *B-zones*, qui produisent le reste.

On considère une machine à vendanger (ou vendangeuse) qui possède deux réservoirs identiques appelés *trémies*, qui ont chacune une capacité maximale notée *Cmax*. Grâce à ce type de machine, les deux catégories de grappes peuvent être séparées et donc la qualité supérieure préservée. Dès le début de la récolte, une des trémies est dédiée à recevoir seulement les A-grappes (trémie a) et l'autre le reste. Une fois que la machine a rempli une de ses trémies, elle doit vidanger ses deux réservoirs dans des bennes situées autour de la parcelle. Généralement, les bennes sont emmenées en cave immédiatement après chaque vidange afin de préserver la qualité des grappes. Elles sont immédiatement remplacées par d'autres bennes en attentes sur la parcelle.

De par sa conception, la vendangeuse met un certain temps ($\approx$ 10 sec) pour acheminer le fruit pris de la vigne jusqu'à une de ses trémies. Ce temps est appelé *latence*. Ainsi, au moment où la machine à vendanger passe d'une zone à l'autre, la qualité du raisin récolté ne peut pas être garantie, car la latence impose un mixage des raisins récoltés à ce moment-là. Deux scénarios sont alors possibles : soit la machine passe d'une A-zone à une B-zone, alors les raisins ramassés dès l'entrée de la nouvelle zone sont considérés comme B-grappes et vont dans la trémie b ; soit la machine passe d'une B-zone à une A-zone et dans ce cas, les grappes (situées sur la A-zone) sont considérées comme B-grappes durant le temps de latence et iront dans la trémie correspondante. Ainsi pour un même rang, le type et le nombre de transitions peuvent être différents suivant le sens de récolte. La quantité de raisin de chaque catégorie peut dont varier. Par exemple, soit un rang ayant pour séquence les zones : $A-B-A-B$ (voir figure 1) : si la machine récolte le rang de gauche à droite, il y a seulement une transition $B-A$; inversement, si la machine passe de droite à gauche, il y a cette fois deux transitions $B-A$ et donc deux fois plus de A-grappes dans la trémie b.

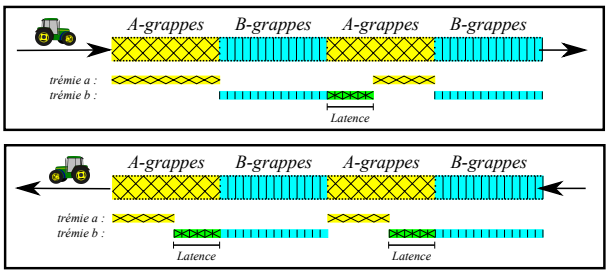

FIGURE 1 – Les quantités de grappes dépendent du sens de récolte du rang.

Notons *Rmin*, le volume de raisin de qualité A à atteindre. *Rmin* correspond à un volume minimal désiré par la cave pour compléter un manque de raisin de première catégorie. La vendange se déroule en deux phases. La première phase impose une vendange sélective jusqu'à ce que le seuil *Rmin* soit atteint. Pendant cette phase, la trémie a doit contenir uniquement du raisin A. La deuxième phase débute après la dernière vidange des trémies qui a permis d'atteindre le volume *Rmin* de raisin de qualité A. À partir de là, les deux trémies peuvent contenir les deux catégories de raisin sans aucune distinction. Suivant *Rmin*, il y a trois possibilités pour remplir les deux trémies : dans un premier temps, les deux trémies servent à séparer les deux catégories de raisins (voir figure 2.a). Dans le cas où la trémie a serait pleine, il est possible de continuer la récolte en dirigeant les raisins A dans l'autre trémie (voir figure 2.b). Ces raisins doivent être considérés par la suite comme du raisin déclassé. Enfin, lorsque *Rmin* est atteint, les deux trémies servent à récolter les grappes sans distinction (voir figure 2.c).

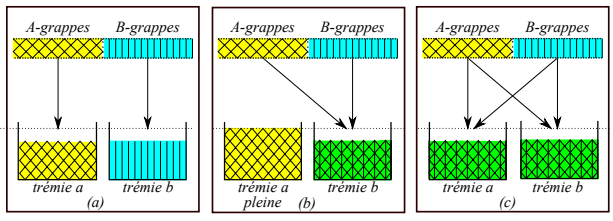

FIGURE 2 – 3 cas possibles pour remplir les trémies. (a) A-grappes et B-grappes sont séparées. (b) quand la trémie a est pleine, les A-grappes et B-grappes sont mixées dans la trémie b. (c) Une fois que *Rmin* est atteint, A-grappes et B-grappes sont mélangées dans les deux trémies.

La vigne, composée de n rangs, est modélisée en considérant les deux extrémités de chaque rang. Un rang

$i \in \{0, \ldots, n-1\}$ est représenté par ses extrémités $2i$ et $2i+1$. Pour chaque rang, notons $Q^A_{2i \to 2i+1}$ et $Q^B_{2i \to 2i+1}$ (resp. $Q^A_{2i+1 \to 2i}$ et $Q^B_{2i+1 \to 2i}$) les quantités de grappes contenues dans le rang i lorsque la machine récolte dans le sens $2i \to 2i+1$ (resp. $2i+1 \to 2i$). Ces quantités tiennent compte de la latence évoquée plus haut. Une autre donnée importante est le temps de trajet de la machine entre chaque rang et la benne. Notons $d(p,q) = d(q,p)\ \forall p, q \in \{0,1,\ldots,2n-1\} \cup \{2n\}$ le temps de passage d'une extrémité p à une extrémité q (avec l'extrémité $2n$ qui représente l'emplacement de la benne). Ce coût dépend du rayon de braquage de la machine à vendanger.

Définition 1. *Problème de la Vendange Sélective : Soit un vignoble représenté par une matrice de coût entre extrémités de rangs (ou benne), une estimation de la quantité de raisin A et B dans chaque rang suivant l'orientation, un seuil Rmin de A-grappes à atteindre et une machine à vendanger à deux trémies de capacité maximale* $2 \times Cmax$*. Le problème de la vendange sélective consiste à trouver une séquence de rangs qui minimise le déplacement total de la vendangeuse et qui permet de récolter tout le raisin dont au moins Rmin A-grappes.*

3 Modélisation CSP

Nous pouvons modéliser le problème de la vendange sélective comme un problème de satisfaction de contraintes ou *COP* décrit par : un ensemble de variables, un domaine représentant les valeurs possibles de chaque variable et un ensemble de contraintes sur ces variables. Le problème d'optimisation de contraintes consiste à rechercher une instanciation des variables avec une valeur de leurs domaines qui vérifie toutes les contraintes du problème tout en optimisant une fonction objectif.

Soit n le nombre de rangs du vignoble, λ le nombre passages à la benne avec γ le nombre de vidanges où le raisin est trié ($\gamma \leq \lambda \leq n$). Nous appelons *site* un rang ou une benne. Notons $R = \{0, \ldots, n-1\}$, l'ensemble des rangs et $\mathcal{S} = R \cup \{n, \ldots, n+\lambda-1\}$ l'ensemble des sites du vignoble. Le problème de la vendange sélective peut-être défini comme suit :

3.1 Variables

Nous créons, pour chaque site $i \in \mathcal{S}$:

- P_i et S_i les variables qui représentent respectivement le site précédant, et le site suivant le site i. Le domaine est $D(P_i) = D(S_i) = \mathcal{S} \setminus \{i\}$;
- Mix_i, la variable booléenne qui représente le mode de récolte. Si $Mix_i = 0$, les A-grappes sont préservées (première phase) sinon, ces grappes sont mélangées avec les B-grappes (deuxième phase). Le domaine de cette variable est : $D(Mix_i) = \{0,1\}$;
- U^A_i et U^B_i, représentent les quantités totales de A-grappes et de B-grappes récoltées depuis la dernière vidange jusqu'au site i inclus. On a : $D(U^A_i) = D(U^B_i) = \{0, \ldots, 2 \times Cmax\}$. La variable U^B_i peut dépasser $Cmax$ uniquement en mode non-trié ;
- T_i, la variable représentant le temps de trajet entre le site i et le site précédent. $D(T_i) = \mathbb{N}$;

Nous créons, pour chaque rang $r \in R$:

- Ori_r, la variable booléenne représentant l'orientation du rang r (0 représente la direction $2r \to 2r+1$ et 1 l'inverse). $D(Ori_r) = \{0,1\}$;
- u^A_r (resp. u^B_r), la variable représentant la quantité de raisin de qualité A (resp. B) contenue dans le rang r suivant son orientation. $D(u^A_r) = D(u^B_r) = \mathbb{N}$;

3.2 Contraintes

La première contrainte est la contrainte globale alldifferent (1) qui contraint de passer une et une seule fois dans chaque rang. Elle porte sur l'ensemble des variables précédant :

$$AllDifferent(P_0, \ldots, P_{n+\lambda-1}) \tag{1}$$

La contrainte (2) est une contrainte de channeling entre les variables précédant et suivant :

$$P_{S_i} = i \qquad S_{P_i} = i \qquad \forall i \in \mathcal{S} \tag{2}$$

Les contraintes (3) et (4) imposent le mode de vendange (sélective ou pas) suivant l'indice correspondant à une benne de la phase 1 (n à $n+\gamma-1$) ou de la phase 2 ($n+\gamma$ à $n+\lambda-1$). La contrainte (5) est une containte « element » qui transmet le mode de vendange entre successeurs.

$$Mix_i = 0 \qquad \forall i \in \{n, \ldots, n+\gamma-1\} \tag{3}$$

$$Mix_i = 1 \qquad \forall i \in \{n+\gamma, \ldots, n+\lambda-1\} \tag{4}$$

$$Mix_r = Mix_{S_r} \qquad \forall r \in R \tag{5}$$

La contrainte de table suivante permet d'attribuer les quantités de raisin suivant le sens de traitement du rang. $\forall i \in R$, on a :

Ori_i	u^A_i	u^B_i
0	$Q^A_{2i \to 2i+1}$	$Q^B_{2i \to 2i+1}$
1	$Q^A_{2i+1 \to 2i}$	$Q^B_{2i+1 \to 2i}$

Les quantités de raisin récoltées (contenues dans les trémies) depuis une vidange sont calculées en fonction

du volume contenu dans le rang et de la quantité précédente. Les quantités associées à une benne valent 0 :

$$U_i^\alpha = 0 \qquad \forall \alpha \in \{A, B\} \;\; \forall i \in \mathcal{S} \setminus R \qquad (6)$$
$$U_i^\alpha = U_{P_i}^\alpha + u_i^\alpha \qquad \forall \alpha \in \{A, B\} \; \forall i \in R \qquad (7)$$

Les quantités calculées en (7) sont limitées par la capacité des trémies (8). La quantité maximale de B-grappes diffère suivant le mode de récolte (9).

$$U_i^A + U_i^B \leq 2 \times Cmax \qquad \forall i \in R \qquad (8)$$
$$U_i^B \leq (1 + Mix_i) \times Cmax \qquad \forall i \in R \qquad (9)$$

La contrainte (10) permet de remplir le contrat de récolte $Rmin$. Elle concerne uniquement le mode de récolte séparé, c'est-à-dire les vidanges de n à $n + \gamma - 1$. Il faut prendre en compte que les A-grappes qui sont dans la trémie a. Cette quantité correspond à la partie de U_i^A qui ne dépasse pas la capacité de la trémie a.

$$\sum_{i=n}^{n+\gamma-1} min(U_{p_i}^A, Cmax) \geq Rmin \qquad (10)$$

Il est possible de reformuler la contrainte (10) sans la fonction *minimum*. Nous ajoutons une nouvelle variable nommée $Ac_i \; \forall i \in \{n, \ldots, n+\gamma-1\}$ de domaine $D(Ac_i) = \{0, \ldots, Cmax\}$ avec :

$$Ac_i \leq Cmax \qquad (10'a)$$
$$Ac_i \leq U_{P_i}^A \qquad \forall i \in \{n, \ldots, n+\gamma-1\} \qquad (10'b)$$
$$\sum_{i=n}^{n+\gamma-1} Ac_i \geq Rmin \qquad (10'c)$$

Il est possible d'interdire les trajets entre deux extrémités qui ne sont pas du même côté du vignoble par la contrainte (11) :

$$Ori_i = 1 - Ori_{P_i} \qquad \forall i \in R \qquad (11)$$

Enfin, l'optimisation porte sur le temps de trajet de la machine. On pose la contrainte (12), que l'on codera par une contrainte de table entre le site précédent, son orientation et le site i :

$$T_i = d(2P_i + Ori_{P_i}, 2i + (1 - Ori_i)) \quad \forall i \in \mathcal{S} \qquad (12)$$

Au final, on souhaite optimiser l'expression suivante :

$$Minimiser \; : \; \sum_{i=0}^{n+\lambda} T_i \qquad (13)$$

4 Améliorations du modèle

Nous avons présenté dans la section précédente une modélisation du problème de la vendange sélective. Nous avons opté pour des variables représentant les rangs précédents et suivants. Cette modélisation possède une symétrie de valeurs. Elle est liée aux bennes de même type qui sont interchangeables. Ces permutations donnent des solutions de même coût. Afin d'éliminer ces symétries, nous ajoutons au modèle les contraintes suivantes :

$$P_i < P_{i+1} \qquad \forall i \in \{n, \ldots, n+\gamma-2\} \qquad (14)$$
$$P_i < P_{i+1} \qquad \forall i \in \{n+\gamma, \ldots, n+\lambda-1\} \qquad (15)$$

La contrainte (14) (respectivement (15)) impose un ordre croissant sur les valeurs des rangs précédant une vidange séparée (resp. non-séparée).

Par ailleurs, comme il est dit dans [8] pour les problèmes de tournées de véhicules, le modèle proposé dans la section 3 présente un autre défaut. Lors de la recherche, des cycles ne passant par aucune benne risquent d'être explorés inutilement. Il est possible d'interdire cette configuration par la contrainte globale *cycle*. Cette contrainte, introduite dans [5] et [10] permet, en ayant un ensemble de variables représentant des successeurs (ou prédécesseurs) d'avoir un certain nombre de cycles entre elles. Ici, nous imposons qu'il y ait un seul cycle sur les variables représentant les sites précédents et un seul sur les variables représentant les sites suivants. La combinaison de la contrainte *cycle* avec la contrainte $Alldifferent$ permet d'avoir un seul et unique prédécesseur et successeur. Dans notre cas, une modélisation avec ou sans cette contrainte possède le même ensemble de solution mais l'efficacité du filtrage est alors amoindrie.

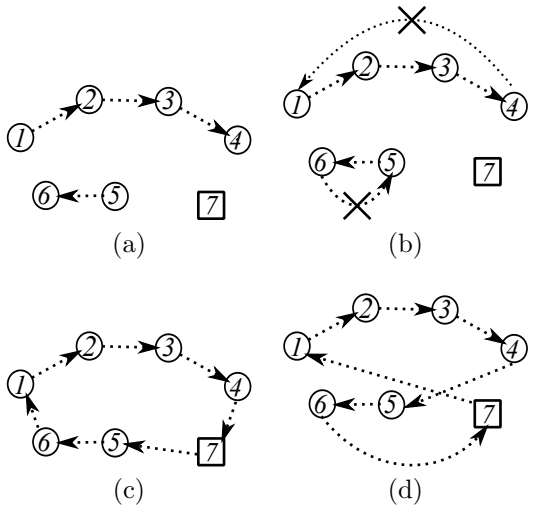

FIGURE 3 – La contrainte de cycle sur les variables successeurs entre les rangs (cercles) et une benne (carré).

Par exemple, dans la figure 3, à partir de la solution partielle (a), les cycles présents dans la solution (b)

peuvent être filtrés. Autrement dit, imposer un seul cycle permet de retirer les valeurs (figure 3.b) : $1 \in \mathrm{D}(S_4)$ et $4 \in \mathrm{D}(P_1)$, $5 \in \mathrm{D}(S_6)$ et $6 \in \mathrm{D}(P_5)$ ($7 \notin \mathrm{D}(S_7)$ et $\mathrm{D}(P_7)$) et les seules solutions sont données en (c) et (d). A notre modélisation, nous ajoutons les contraintes (16) et (17) :

$$Cycle(1, (P_0, \ldots, P_{n+\lambda})) \quad (16)$$
$$Cycle(1, (S_0, \ldots, S_{n+\lambda})) \quad (17)$$

5 Travaux connexes

Une première modélisation de ce problème a été présentée dans [4] avec laquelle nous avons pu comparer les résultats. Dans cette précédente modélisation, une variable est associée à chaque étape pour décrire l'extrémité visitée à cette étape. Des variables booléennes, associées à chaque étape, indiquent si la machine doit faire une vidange à ce moment-là ou non. Le basculement entre la récolte sélective et la récolte normale est imposé en fixant l'indice du rang après lequel la récolte devient simple. Ainsi, la recherche consiste à lancer plusieurs résolutions en parallèles avec les différents indices possibles pour le basculement et à conserver le résultat optimal. Dans cet article, nous avons montré l'interêt de la démarche qui permet jusqu'à 40% de gain de temps par rapport à une trajectoire traditionelle de la machine à vendanger.

6 Résultats

Nous avons implémenté les deux modélisations avec le solveur AbsCon [9], écrit en Java, auquel nous avons adjoint une contrainte *cycle* garantissant une permutation circulaire entre les variables. Nos tests ont été effectués sur un MacBook Pro avec 2,6 GHz Intel Core i5 et 8 Go de RAM. Les données ont été fournies par Montpellier SupAgro et sont issues d'un vignoble provenant de l'INRA du Pech-Rouge (Gruissan, Aude, France) (figure 4). La table 1 représente le temps de calcul et le nombre de nœuds pour la résolution du problème avec les deux modèles. Le premier modèle correspond à celui présenté dans [4]. Les colonnes 2 et 3 correspondent à la modélisation présentée ici sans et avec les contraintes de cycle. Les nombres de rangs testés (n=10 et n=16) correspondent à un sous-ensemble de rangs de la même parcelle. Pour chaque modèle, nous avons utilisé l'heuristique de choix de valeurs *Random*, qui donne les meilleurs temps de calculs et nous avons imposé un cutoff de 10 échecs au départ, multiplié par 2 à chaque restart.

Ces résultats montrent que sur de petites instances réelles, le modèle proposé permet de trouver la solution optimale dans un temps relativement raisonnable

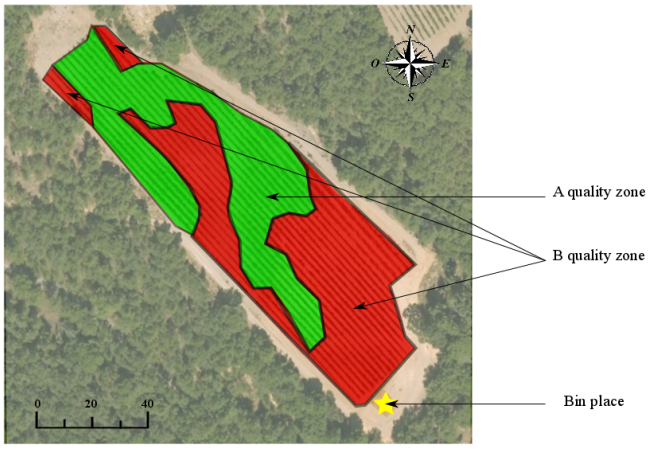

FIGURE 4 – Un vignoble avec deux qualités : B-grappes en rouge et A-grappes en vert.

et améliore sensiblement les temps de calculs du modèle présenté dans [4]. L'ajout des contraintes de cycle permet de diminuer l'espace de recherche et le temps de résolution par rapport à la même approche sans ces contraintes. Ces résultats restent préliminaires et doivent être confrontés au passage à l'échelle, mais sont encourageants pour la suite. De plus, il est important de noter que notre approche est exacte et calcule la solution optimale du problème. Pour de plus grandes instances, nous pourrions recourir à des méthodes locales de type LNS (Large Neigborhood Search) qui ont donné de bons résultats sur un problème similaire [6].

7 Conclusion

Dans cet article, nous avons présenté le problème de la vendange sélective issu de l'agriculture de précision. Après une formalisation de ce problème, une modélisation sous la forme d'un problème d'optimisation sous contraintes à été présentée. Ce nouveau modèle permet de résoudre le problème de manière optimale en des temps raisonnables sur de petites instances réelles. Il reste à prouver la validité de l'approche sur de plus grandes instances.

	Modèle de [4]		Modèle sans (16) et (17)		Modèle avec (16) et (17)	
n	tps	# Noeuds	tps	# Noeuds	tps	# Noeuds
10	1 100s	6 624 876	413s	48 050	41s	8 038
16	326 143s	647 119 328	67 826s	8 688 417	11 841s	177 894

TABLE 1 – Résultats sur 10 et 16 rangs. Comparaison entre le modèle de [4] et celui proposé dans cet article.

Références

[1] Pascal Benchimol, Willem Jan van Hoeve, Jean-Charles Régin, Louis-Martin Rousseau, and Michel Rueher. Improved filtering for weighted circuit constraints. *Constraints*, 17(3) :205–233, 2012.

[2] Dionysis Bochtis and Claus G. Sørensen. The vehicle routing problem in field logistics : part i. *Biosystems engineering*, 104 :447–457, 2009.

[3] Dionysis Bochtis and Claus G. Sørensen. The vehicle routing problem in field logistics : part ii. *Biosystems engineering*, 105 :180–188, 2010.

[4] Nicolas Briot, Christian Bessiere, Bruno Tisseyre, and Philippe Vismara. Integration of operational constraints to optimize differential harvest in viticulture. In *Proc. 10th European Conference on Precision Agriculture*, July 2015, à paraitre.

[5] Yves Caseau and François Laburthe. Solving small tsps with constraints. In Lee Naish, editor, *Logic Programming, Proceedings of the Fourteenth International Conference on Logic Programming, Leuven, Belgium, July 8-11, 1997*, pages 316–330. MIT Press, 1997.

[6] Luca Di Gaspero, Andrea Rendl, and Tommaso Urli. Constraint-based approaches for balancing bike sharing systems. In *Principles and Practice of Constraint Programming*, volume 8124 of *Lecture Notes in Computer Science*, pages 758–773. Springer, 2013.

[7] Jean-Guillaume Fages. *Exploitation de structures de graphe en programmation par contraintes.* PhD thesis, Ecole des Mines de Nantes, 2014.

[8] Philip Kilby and Paul Shaw. Vehicle routing. In Francesca Rossi, Peter Van Beek, and Toby Walsh, editors, *Handbook of constraint programming*, chapter 23, pages 799, 834. Elsevier, 2006.

[9] Sylvain Merchez, Christophe Lecoutre, and Frédéric Boussemart. Abscon : A prototype to solve csps with abstraction. In Toby Walsh, editor, *Principles and Practice of Constraint Programming - CP 2001, 7th International Conference, Paphos, Cyprus, 2001*, volume 2239 of *Lecture Notes in Computer Science*, pages 730–744. Springer, 2001.

[10] Gilles Pesant, Michel Gendreau, Jean-Yves Potvin, and Jean-Marc Rousseau. An exact constraint logic programming algorithm for the traveling salesman problem with time windows. *Transportation Science*, 32(1) :12–29, 1998.

[11] Bruno Tisseyre, Hernan Ojeda, and James Taylor. New technologies and methodologies for site-specific viticulture. *Journal International des Sciences de la Vigne et du Vin*, 41 :63–76, 2007.