Un algorithme tabou pour résoudre, grâce aux groupes d'opérations permutables, un problème d'ordonnancement et de routing robuste

Azeddine Cheref, Teddy Bouchard, Jean-Charles Billaut, Christian Artigues

10ème Conférence Internationale de MOdélisation, Optimisation et SIMulation - MOSIM'14 - 5 au 7 Novembre 2014 - Nancy -France « de l'Économie linéaire à l'économie circulaire »

Un algorithme tabou pour résoudre, grâce aux groupes d'opérations permutables, un problème d'ordonnancement et de routing robuste

A. Cheref[123] **T. Bouchard**[1] **J-C. Billaut**[1] **C. Artigues**[23]

[1]Université François-Rabelais de Tours / CNRS
LI EA 6300, OC ERL CNRS 6305
64 av. J. Portalis
37200 Tours - France
jean-charles.billaut@univ-tours.fr

[2]CNRS, LAAS
7 avenue du colonel Roche
F-31400 Toulouse, France
{acheref, artigues}@laas.fr

[3]Univ de Toulouse, LAAS
F-31400 Toulouse, France

RÉSUMÉ : *Nous abordons dans cet article un problème conjoint d'ordonnancement et de routing de véhicule, pour minimiser le plus grand retard algébrique. Une incertitude porte les données (incertitude prise en compte par des scénarios) et une solution* robuste *est recherchée. L'approche proposée consiste à utiliser le concept de groupes d'opérations permutables pour converger selon le scénario vers la solution la plus intéressante (ici la règle FIFO est utilisée pour donner priorité à l'opération disponible le plus tôt). Deux algorithmes Tabou sont proposés, l'un sans groupe et l'autre avec des groupes. Les expérimentations numériques montrent l'intérêt de l'approche par les groupes, qui s'avère la plus robuste.*

MOTS-CLÉS : *Ordonnancement, routing, robustesse, méthode tabou*

1 INTRODUCTION

Dans cet article, on aborde de façon conjointe un problème d'ordonnancement de production et un problème de routing. Les tâches à ordonnancer doivent être livrées pour une certaine date, et on suppose qu'on ne dispose que d'un unique véhicule. On cherche une solution au problème global qui minimise le plus grand retard. D'autre part, on se place dans un environnement incertain, où les données sont connues selon des scénarios possibles. On cherche donc à mettre au point une approche robuste, c'est-à-dire garantissant la meilleure performance possible pour tous les scénarios envisagés.

Le problème est extrait d'une problématique réelle dans le contexte de la production et la distribution de préparations de chimiothérapies injectables. Au CHRU de Tours, une fois les poches ou seringues préparées, celles-ci sont emmenées dans les services (voire dans des services sur un autre site hospitalier de la ville) par une personne, chargée d'effectuer ces livraisons. Produire ces poches uniquement en fonction de leur destination permettrait sans doute d'optimiser la livraison, mais cela ne correspond pas nécessairement aux rendez-vous convenus avec les patients. Inversement, produire les poches en fonction des dates d'administration souhaitées afin de réduire l'attente des patients, sans tenir compte du lieu de la livraison n'est pas non plus totalement satisfaisant. Cette problématique se retrouve dans tout système où la production et la livraison doivent être décidées de façon coordonnée. La littérature sur les approches intégrées contient assez peu d'articles lorsque les problèmes considérés sont à un niveau opérationnel. Dans la plupart des cas, les articles traitent cette problématique à un niveau stratégique, dans le cadre d'une chaîne logistique. Le survey de (Chen, 2010) permet d'avoir une très bonne vision de la littérature sur cette catégorie de problèmes.

Parfois, le problème est ramené à un problème d'ordonnancement par batchs avec dates de fin communes (Cheng, 1996). Le batch est supposé constituer une tournée. (Averbakh, 2010) considère un système de production-distribution temps réel, où des clients commandent en temps réel des travaux, qui doivent être produits et livrés par le fabricant. Les travaux sont regroupés en batchs de taille limitée pour constituer une livraison à un client. L'objectif est de minimiser le coût total, composé du temps de séjour pondéré et du coût de livraison. Notre problème est proche de celui abordé dans (Ullrich, 2013), mais cet auteur n'intègre pas les aspects de robustesse que nous prenons en compte ici. Dans (Viergutz, 2014), les auteurs considèrent un système de production à la commande composé d'une machine unique, qui réalise un produit unique ayant une courte durée de vie après fabrication. Les commandes doivent être livrées à des clients répartis géographiquement, dans une fenêtre de temps impérative donnée. Un seul véhicule effectue les livraisons. Le problème consiste à sélectionner les commandes à réaliser de sorte à maximiser la satisfaction de la demande. Le problème est donc

davantage abordé comme un problème de sélection.

Les *approches robustes* sont largement abordées dans la littérature. (Kouvelis, 1997) présente de façon détaillée les différentes approches possibles pour la robustesse. Dans le domaine de l'ordonnancement, nous renvoyons aux surveys de Herroelen et Leus sur la génération d'ordonnancements robustes proactifs pour le problème de RCPSP : (Herroelen, 2004) sur l'ordonnancement sous incertitude, l'ordonnancement réactif, l'ordonnancement de projet stochastique, l'ordonnancement de projet flou et l'analyse de sensibilité et (Herroelen, 2005) sur l'ordonnancement avec prise en compte des aléas et ré-ordonnancement. Nous proposons dans (Artigues, 2013) une approche flexible basée sur les groupes d'opérations permutables, et montrons comment ce genre d'approche permet d'obtenir des solutions d'une plus grande robustesse, mais uniquement pour le problème d'ordonnancement à une machine.

Dans la Section 2, le problème est décrit de façon formelle et les notations sont introduites. Dans la Section 3, l'algorithe Tabou proposé est décrit. La Section 4 reporte les résultats des expérimentations numériques et montre l'intérêt des groupes de tâches permutables pour ce problème difficile.

2 FORMALISATION DU PROBLEME ET ILLUSTRATION

On considère un ensemble $\{J_1, J_2, ..., J_n\}$ de n travaux à ordonnancer sur une machine unique. A chaque travail J_j, on associe une date de début au plus tôt r_j^s, une durée d'exécution p_j^s, une destination j (la destination 0 correspond au site de production) et une date de livraison souhaitée d_j^s, selon un scenario s, appartenant à un ensemble de scenarios $\mathcal{S}$. On connaît $t_{i,j}^s$ $(0 \leq i, j \leq n)$ le temps nécessaire pour aller de la destination i à la destination j dans le scenario $s \in \mathcal{S}$.

Définition : Un *groupe de tâches permutables* (voir par exemple (Billaut, 1996)) est un ensemble de tâches pour lequel la séquence d'exécution des tâches au sein du groupe n'est pas définie a priori.

Un groupe permet donc, lors de l'exécution réelle de l'ordonnancement, d'avoir une flexibilité dans l'ordre d'exécution des tâches. L'intérêt de la méthode est que quelque soit l'ordre d'exécution choisi dans le groupe, la performance de l'ordonnancement de groupes est toujours garantie. Dans le contexte que nous considérons, avec incertitude sur les données, on prend comme hypothèse qu'au sein d'un groupe, lors de l'exécution de l'ordonnancement sous le scenario s, les tâches sont exécutées dans l'ordre FIFO, donc dans l'ordre des r_j^s croissants. L'aspect "flexibilité" des groupes de tâches permutables est ici utilisé pour modifier la séquence au sein d'un groupe en fonction du scénario, ce qui correspond à une attitude qui peut être mise en oeuvre en temps réel. Les scénarios réduisent donc un peu l'intérêt des groupes, qui permettent aussi de la flexibilité pour modéliser d'autres formes d'incertitude.

Exemple : Considérons l'instance de cinq tâches avec deux scenarios décrite Tableau 1.

j	1	2	3	4	5
r_j^1	0	1	2	0	10
p_j^1	3	2	9	7	12
d_j^1	28	32	29	30	33
j	1	2	3	4	5
r_j^2	1	0	1	1	10
p_j^2	3	3	8	6	13
d_j^2	28	31	30	30	33

Tableau 1: Instance à cinq tâches et deux scenarios

On considère la séquence de groupes $(\{J_1, J_2, J_3\}, \{J_4, J_5\})$. Les tâches J_1, J_2 et J_3 précèdent les tâches J_4 et J_5. L'ordre des tâches $\{J_1, J_2, J_3\}$ n'est pas fixé, celui des tâches $\{J_4, J_5\}$ non plus. Il n'est pas facile de représenter cet ordonnancement sous forme d'un diagramme de Gantt, puisque selon l'ordre d'exécution des tâches, les dates de début et de fin des tâches changent. Cet ordonnancement de groupes permet de caractériser à lui seul 12 ordonnancements. Seuls deux de ces ordonnancements seront utilisés par la suite, chacun correspondant à un scénario.

En supposant que la règle FIFO est appliquée, la solution qui sera mise en oeuvre pour le scenario 1 est la séquence $S = (J_1, J_2, J_3, J_4, J_5)$, et pour le scénario 2, c'est la séquence $S = (J_2, J_1, J_3, J_4, J_5)$. $\diamond$

Indépendamment des groupes, il est possible ensuite de constituer des batchs (ou lots) pour les tournées, et des tournées pour chaque batch.

Une solution S au problème est définie par :

- un ordonnancement de groupes noté $g(S)$,
- une affectation à des batchs pour les tournées notée $b(S)$,
- une tournée pour chaque batch, notée $t(S)$.

Notons que les batchs ici sont des *batchs de livraison*, ou encore des "lots" et non des batchs de production comme on trouve dans la littérature en *batch scheduling* (le terme "lot" peut aussi renvoyer aux problèmes de *lot-sizing*...). Notons également que dès lors que $t(S)$ est connu, on en déduit immédiatement $b(S)$, donc l'affectation aux tournées est implicite dès lors

que la tournée est connue. Enfin, pour éviter les confusions, le terme "groupe" est réservé aux groupes de tâches permutables, et le terme "batch" est réservé aux tournées de livraison.

Exemple : Reprenons l'instance de cinq tâches avec deux scénarios décrite Tableau 1 avec $g(S) = (\{J_1, J_2, J_3\}, \{J_4, J_5\})$. On considère les matrices des temps de transport indiquées dans le Tableau 2.

$$(t^1_{i,j}) = \begin{pmatrix} 0 & 6,1 & 5,4 & 9,2 & 10,6 & 8,1 \\ 6,1 & 0 & 1,4 & 5,1 & 6,3 & 8,6 \\ 5,4 & 1,4 & 0 & 4,5 & 5,8 & 7,2 \\ 9,2 & 5,1 & 4,5 & 0 & 1,4 & 6,3 \\ 10,6 & 6,3 & 5,8 & 1,4 & 0 & 7,1 \\ 8,1 & 8,6 & 7,2 & 6,3 & 7,1 & 0 \end{pmatrix}$$

$$(t^2_{i,j}) = \begin{pmatrix} 0 & 6,1 & 8,9 & 9,2 & 8,1 & 8,1 \\ 6,1 & 0 & 3,6 & 5,1 & 6,3 & 8,6 \\ 8,9 & 3,6 & 0 & 2,2 & 5 & 8,1 \\ 9,2 & 5,1 & 2,2 & 0 & 3,2 & 6,3 \\ 8,1 & 6,3 & 5 & 3,2 & 0 & 3,2 \\ 8,1 & 8,6 & 8,1 & 6,3 & 3,2 & 0 \end{pmatrix}$$

Tableau 2: Matrice des temps de transport selon les deux scenarios

Considérons la solution S définie par :

- $g(S) = (\{J_1, J_2, J_3\}, \{J_4, J_5\})$,
- $b(S) = (\{J_1, J_2\}, \{J_3, J_4\}, \{J_5\})$ pour indiquer qu'une tournée emportera les travaux J_1 et J_2, une autre tournée J_3 et J_4, et enfin une dernière tournée J_5 seul.
- $t(S) = ((J_2, J_1), (J_4, J_3), (J_5))$ précise l'ordre dans lequel les sites sont visités par chaque tournée.

Cette solution est représentée figure 1. $\diamond$

Figure 1: Solution pour le scénario 1

On note C^s_j la date de fin de production de la tâche J_j et D^s_j sa date de fin de livraison, selon le scenario $s \in \mathcal{S}$. On définit le retard algébrique de la tâche J_j selon le scenario s par $L^s_j = D^s_j - d^s_j$. On définit le critère à optimiser :

$$L_{\max} = \max_{s \in \mathcal{S}} \left(\max_{j=1}^{n} L^s_j \right)$$

Exemple : On obtient donc pour l'exemple les valeurs indiquées dans le Tableau 3. $\diamond$

On cherche l'ordonnancement de groupes, les compositions des batchs et les tournées de chaque batch, de sorte à minimiser la somme des retards.

j	1	2	3	4	5
C^1_j	3	5	14	21	33
D^1_j	11,8	10,4	33	31,6	50,3
T^1_j	0	0	4	1,6	17,3

Tableau 3: Dates de fin des tâches pour l'exemple (premier scénario)

3 METHODES DE RESOLUTION

3.1 Solution initiale

Pour construire la solution initiale, les travaux sont d'abord triés selon EDD. Ensuite, sur les bases de cette séquence, trois solutions sont construites :

- une solution où chaque tâche constitue un groupe (donc il n'y a aucune flexibilité),
- une solution avec deux groupes : les $n/2$ premières tâches sont regroupées en un groupe et les tâches restantes en un second groupe,
- enfin une solution avec trois groupes.

Ensuite, pour la livraison, un batch est composé d'une seule tâche. Il n'y a donc pas de problème de routing à résoudre, puisque le véhicule va livrer la tâche et revient.

3.2 Algorithme Tabou

L'algorithme Tabou implémenté correspond à la méthode standard (Glover, 1989). Nous décrivons ci-dessous le codage d'une solution et les voisinages qui sont implémentés. Le critère d'arrêt est un temps limite d'exécution.

3.2.1 Codage

Le problème de routing des tâches d'un batch est systématiquement résolu par l'algorithme du plus proche voisin. Il suffit donc de connaître l'ordonnancement de groupes et la composition des batchs pour reconstruire une solution complète.

Une solution du problème S est codée de façon directe par un vecteur v de taille $2n$. La première partie de taille n contient l'affectation dans les groupes, autrement dit v_j pour $1 \leq j \leq n$ est le numéro du groupe dans lequel la tâche J_j est affectée. Les groupes sont supposés être séquencés dans l'ordre de leur numérotation. Dans la seconde partie du vecteur on trouve l'affectation aux batchs, autrement dit v_j pour $n+1 \leq j \leq 2n$ est le batch dans lequel se trouve la tâche J_{j-n}.

Exemple: La figure 2 illustre la solution de l'exemple précédent et son codage.

Figure 2: Codage d'une solution

3.3 Voisinages

Sur la base de ce codage, plusieurs voisins ont été définis. Quelques algorithmes sont donnés pour plus de précision.

Le premier voisinage est "SWAP_Gpe". Ce voisinage effectue l'échange de deux valeurs dans la première partie du codage, autrement dit l'échange de l'affectation de deux tâches à des groupes.

SWAP_Gpe
Pour $i = 1$ à $n - 1$ Faire
  Pour $j = i + 1$ à n Faire
    Tester swap(v_i, v_j) :
    $v' = (v_1, ..., v_{i-1}, v_j, v_{i+1}, ..., v_{j-1}, v_i, v_{j+1}, ...)$
  Fin Pour
Fin pour

Le second voisinage est "BF_GpeIns" (pour "*Backward / Forward Group Insertion*"), il change une valeur v_j dans la première partie du codage ($1 \leq j \leq n$) pour $v_j \pm \delta$. De ce fait, une tâche se trouve affectée à un groupe précédent ou suivant.

BF_GpeIns
Pour $j = 1$ à n Faire
  Pour $d = -\delta$ à $+\delta$ Faire
    Tester bfgi$(v_j + d)$:
    $v' = (v_1, ..., v_{j-1}, v_j + d, v_{j+1}, ...)$
  Fin Pour
Fin pour

En pratique, tous les groupes possibles sont testés. Le voisinage "CUT_Gpe" génère plusieurs voisins en scindant un groupe en deux à partir de chaque position possible. Le voisinage "GPE_Gpe" regroupe deux groupes consécutifs en un seul.

Concernant les tournées, le voisinage "SWAP_Batch" effectue l'échange entre les valeurs v_j et v_k ($n + 1 \leq j \leq 2n$). De ce fait, l'affectation de deux tâches aux tournées se voient permutées. Le voisinage "CUT_Batch" génère plusieurs voisins en scindant un batch en deux à partir de chaque position possible. Le voisinage "GPE_Batch" regroupe deux batchs consécutifs en un seul.

3.4 Algorithme Tabou sans groupe

Afin de pouvoir mesurer l'intérêt de faire des groupes de tâches permutables, un autre algorithme Tabou a été développé. Le codage est également en $2n$ éléments, mais la première partie correspond à la position de la tâche dans la séquence. La deuxième partie est identique au cas "avec groupes". Les voisinages sont similaires : "SWAP_Tache" fait l'échange des positions de deux tâches et "Insert" insère une tâche à une certaine position. Les voisinages de type "CUT_Gpe" ou "GPE_Gpe" n'ont pas lieu d'être.

4 EXPERIMENTATIONS

4.1 Génération des données

On considère des instances avec un nombre de travaux $n \in \{10, 20, 50, 100, 200\}$. Un scénario de référence ($s = 1$) est généré aléatoirement avec des durées $p_j^1 \in [1, 100]$. Ensuite, les dates de début au plus tôt r_j^1 sont générées dans l'intervalle $[1, \gamma P]$ où $P = \sum_{j=1}^{n} p_j^1$ et $\gamma = 1,5$, les dates de fin souhaitées d_j^1 sont générées dans l'intervalle $[(\alpha - \frac{\beta}{2})P, (\alpha + \frac{\beta}{2})P]$ avec $\alpha = 1$ et $\beta = 0.2$. Pour générer des distances, les coordonnées (x_j^1, y_j^1) des n destinations ont été générées aléatoirement dans l'intervalle $[1, 70]$ et la distance choisie entre deux destinations est la distance euclidienne (de sorte que la distance maximale entre deux destinations est inférieure ou égale à 100).

Pour générer les autres scénarios, une déviation par rapport au scénario de référence est appliquée :

- $p_j^s \in [(1 - \omega)p_j^1, (1 + \omega)p_j^1]$,
- $r_j^s \in [(1 - \omega)r_j^1, (1 + \omega)r_j^1]$,
- $d_j^s \in [(1 - \omega)d_j^1, (1 + \omega)d_j^1]$,
- $x_j^s \in [(1 - \omega)x_j^1, (1 + \omega)x_j^1]$,
- et $y_j^s \in [(1 - \omega)y_j^1, (1 + \omega)y_j^1]$, avec $\omega = 20\%$

Dix instances ont été générées par valeur de n. Deux catégories d'instances ont été générées, soit avec deux scénarios, soit avec six scénarios. Le temps de calcul a été limité à 300 secondes.

4.2 Résultats

Les résultats sont présentés Tableau 4 et 5. Les colonnes n et "scenarios" indiquent la taille des instances considérées. La colonne Δ est la déviation relative moyenne entre la valeur de la solution initiale et la valeur de la solution finale de l'algorithme Tabou.

$$\Delta = Average\left(\frac{ValInit - ValTabou}{ValInit}\right)$$

La colonne "#best" indique le nombre de fois où la méthode renvoit la meilleure solution. La colonne "ttb" indique le temps (en secondes) que met la méthode pour atteindre la meilleure solution ("*time to best*").

Les résultats montrent que les deux méthodes améliorent beaucoup la qualité de leur solution initiale, et dans les mêmes ordres de grandeur : en moyenne 67% pour la méthode sans les groupes pour $n \leq 100$ et 14% pour $n = 200$, et en moyenne 61% pour la méthode avec les groupes pour $n \leq 100$ et 15% pour $n = 200$.

n	scenarios	Δ	#best	ttb
10	2	72%	8	43,6
	6	61%	3	12,86
20	2	74%	5	57,38
	6	66%	1	24,65
50	2	72%	2	77,04
	6	67%	3	87,14
100	2	65%	1	300,4
	6	59%	2	301,54
200	2	16%	0	304,59
	6	13%	0	307,13

Tableau 4: Résultats des expérimentations sans les groupes

n	scenarios	Δ	#best	ttb
10	2	59%	7	18,59
	6	56%	7	10,54
20	2	66%	7	2,39
	6	63%	10	6,04
50	2	70%	10	66,17
	6	67%	9	103,61
100	2	59%	9	289,93
	6	48%	8	284,62
200	2	20%	10	306,73
	6	11%	10	314,96

Tableau 5: Résultats des expérimentations avec les groupes

La colonne "#Best" indique clairement qu'il est plus intéressant de faire des groupes de tâches permutables. Le tableau 6 indique l'écart relatif Δ' entre les deux méthodes : Tabou Sans Groupe ($ValTabouSG$) et Tabou Avec Groupes ($ValTabouAG$).

$$\Delta' min = Min\left(\frac{ValTabouSG - ValTabouAG}{ValTabouSG}\right)$$

$$\Delta' moy = Average\left(\frac{ValTabouSG - ValTabouAG}{ValTabouSG}\right)$$

$$\Delta' max = Max\left(\frac{ValTabouSG - ValTabouAG}{ValTabouSG}\right)$$

L'écart relatif moyen (toutes instances confondues) est de 15%, avec des écarts maximum parfois très importants (58%, 61%). Les écarts minimum négatifs indiquent un avantage pour la méthode sans les groupes, avec un avantage allant une fois jusqu'à 35% pour cette méthode.

n	scenarios	Δ'min	Δ'moy	Δ'max
10	2	-10%	-1%	8%
	6	-35%	5%	44%
20	2	-18%	2%	22%
	6	0%	16%	58%
50	2	0%	26%	56%
	6	-3%	24%	49%
100	2	-4%	15%	39%
	6	-24%	7%	41%
200	2	22%	34%	61%
	6	15%	25%	44%

Tableau 6: Comparaison des solutions avec et sans groupes

5 CONCLUSION

Nous avons abordé dans cet article un problème qui intègre à un niveau opérationnel un problème d'ordonnancement à une machine et un problème de routing avec un véhicule. Le problème consiste à trouver une solution au problème d'ordonnancement et au problème de routing, de sorte à minimiser le plus grand retard algébrique. De plus, on se place dans un environnement incertain. Les incertitudes portent sur les données du problème qui sont supposées connues selon plusieurs scénarios. On cherche donc une solution robuste, donc la solution la meilleure possible pour tous les scénarios.

L'approche proposée est celle des groupes d'opérations permutables, qui offre de la flexibilité séquentielle et qui permet – en temps réel – de choisir une séquence particulière. La séquence choisie au sein d'un groupe pour chaque sénario est ERD (*Earliest Release Date first*) et l'algorithme proposé pour fournir une solution est un algorithme Tabou. Sur la base des mêmes idées, un algorithme Tabou a été codé, n'offrant pas de flexibilité séquentielle (donc fournissant un ordonnancement classique). Les deux algorithmes sont comparés et les comparaisons mettent en évidence l'intérêt de faire des groupes de tâches permutables.

Les perspectives de recherche sont nombreuses, d'une part parce que le problème abordé est relativement original, et d'autre part parce que l'approche par les groupes de tâches permutables pour un gain de robustesse est aussi une approche originale. La première piste de recherche consiste à améliorer la qualité de la solution initiale. Plusieurs pistes sont possibles. Les deux algorithmes Tabou proposés ont été codés de sorte à donner les "mêmes chances" aux deux approches, mais il existe quand même des nuances. En pratique, il est difficile de dire si chaque algorithme donne à l'approche qu'il "défend" ses meilleures chances. Une première extension va viser à pousser chaque algorithme Tabou vers ses meilleures performances possibles, afin d'offrir une

comparaison plus "robuste". Des modèles de programmation linéaire ont été réalisés et quelques comparaisons pourront être faites pour des instances de petite taille. Il s'agira aussi d'intensifier davantage les expérimentations (plusieurs valeurs sont possibles pour les paramètres γ, ω, α et β). Enfin, d'autres scénarios pourront être générés, où peu de valeurs seront changées, mais avec des écarts plus importants, afin de se placer dans un contexte d'aléas (et non plus d'incertitudes), et de voir comment se comportent les méthodes.

REMERCIEMENTS

Ce travail a bénéficié d'une aide de l'Agence National de la Recherche pour le projet Athena portant la référence ANR-13-BS02-0006-01.

6 REFERENCES

Artigues C., Billaut J-C., Bouchard T., Drevon T. (2013). Caractérisation de solutions pour l'ordonnancement robuste. Congrès ROADEF 2013, Angers.

Averbakh, I. (2010). On-line integrated production-distribution scheduling problems with capacitated deliveries. European Journal of Operational Research, 200 (2), pp. 377-384.

Billaut J-C., Roubellat F., A new method for workshop real time scheduling, *International Journal of Production Research*, pp. 1555-1579, vol. 34, num. 6, 1996.

Chen, Z.-L., 2010. Integrated Production and Outbound Distribution Scheduling: Review and Extensions. Operations Research, 58(1):130-148.

Cheng T.C.E., Kovalyov M.Y., 1996. Batch scheduling and common due-date assignment on a single machine. Discrete Applied Mathematics, 70(3), 231-245.

Glover F., 1989. Tabu search, Part I, *ORSA Journal on Computing*, 1 (3), 190-206.

Herroelen, W., Leus, R. (2004). Robust and reactive project scheduling: A review and classification of procedures, *International Journal of Production Research*, 42 (8), pp. 1599-1620.

Herroelen W. and Leus R. (2005). Project scheduling under uncertainty, survey and research potentials, *European Journal of Operational Research*, 165 (2), pp. 289–306.

Kouvelis P., Yu G. (1997). Robust discrete optimisation and its applications, Kluwer Academic Publishers.

Ullrich C.A., 2013. Integrated machine scheduling and vehicle routing with time windows. European Journal of Operational Research, 227, pp. 152–165.

Viergutz C. and Knust S., 2014. Integrated production and distribution scheduling with lifespan constraints. Annals of Operations Research, 213, pp. 293–318, 2014.