CIGI QUALITA MOSIM 2023

Ordonnancement dynamique et contrôle intelligent d'un robot industriel

Anas NEUMANN [1,3] **Monia REKIK** [1,2,3] **Michael MORIN** [2,3]
Adnene HAJJI [1,3] **Robert PELLERIN** [4,5]

[1] *CIRRELT, Université Laval, Québec, QC, Canada*
[2] *CRISI, Université Laval, Québec, QC, Canada*
[3] *Département d'opérations et systèmes de décision, Université Laval, Québec, QC, Canada (e-mail: {monia.rekik, michael.morin, adnene.hajji}@fsa.ulaval.ca)*
[4] *CIRRELT, École Polytechnique Montréal, Montréal, QC, Canada*
[5] *Département de mathématiques et de génie industriel, École Polytechnique Montréal, Montréal, QC, Canada (e-mail: robert.pellerin@polymtl.ca)*

Résumé **-** Nous présentons, dans ce papier, un modèle mathématique pour l'ordonnancement dynamique d'un robot industriel de soudure. Le robot étudié réalise différents procédés en parallèle et dispose de plusieurs stations de chargement, de soudure, ainsi que d'un bras mécanique pour déplacer les pièces. Le robot est sujet à des temps de configuration et de déplacement du bras et des pièces. Nous proposons également une méthode d'optimisation basée sur une heuristique de construction puis d'amélioration locale à multiple voisinage. Cette méthode exploite les différentes capacités du robot : exécution en parallèle, déchargement de pièces moins urgentes, etc. Une série d'expérimentations démontre que notre méthode permet de résoudre des problèmes de taille réaliste de 190 opérations en temps réel (moins d'une seconde) insolvables à l'optimalité. Notre approche obtient aussi une solution optimale pour des instances de petite taille.

Abstract **-** In this paper, we present a mathematical formulation for the dynamic scheduling of an industrial welding robot. The studied robot executes various processes concurrently, using several loading and welding platforms as well as a mechanical arm to move the pieces. The robot is subject to capacity and setup constraints (time needed to move the arm). We also propose a solution method based on a greedy heuristic followed by a multiple-neighborhood local search. Our approach exploits capabilities of the robot such as parallel execution, unloading the less urgent pieces, etc. We tested our heuristic method to demonstrate its capacity to solve realistic instances with 190 operations in real time (less than a second). The same instances were unsolvable with an exact approach. When applied to small problems, our method was also able to find an optimal solution.

Mots clés **-** Ordonnancement dynamique, Robot industriel, Recherche locale, Exécution parallèle, Temps de configuration

Keywords **-** Dynamic scheduling, Industrial robot, Local search, Concurrent execution, Setup times

1. INTRODUCTION

Les robots industriels se distinguent des équipements de production et d'assemblage plus classiques notamment par leur capacité à réaliser de nombreuses tâches différentes (Huang and Li, 2009; Valente, 2016; Alcácer and Cruz-Machado, 2019). Pour réaliser ces diverses tâches, le robot industriel doit souvent opérer des changements de configuration ou d'outils (Huang and Li, 2009; Valente, 2016) ou encore opérer des déplacements tout en tenant compte de son état actuel et de l'historique des pièces en cours de production. C'est d'autant plus vrai dans un environnement d'ordonnancement dynamique avec la possible nécessité de changer une décision ou d'interrompre les tâches en cours (Julien et al., 1997; Larsen and Pranzo, 2019). Un modèle d'ordonnancement réaliste et précis devrait tenir compte de ces opérations de changement d'état et des spécificités des différentes tâches réalisées (durée, capacité occupée, etc.). Dans cette étude, nous nous intéressons à l'ordonnancement dynamique d'un robot de soudure. Le cas d'étude et les données de test ont été proposés par une entreprise industrielle multinationale.

Le robot étudié, illustré à la Figure 1, dispose de stations de chargement des pièces. À partir d'une certaine taille, les pièces dites "larges" ne peuvent être chargées que via la station 2. Le temps de chargement ou déchargement d'une pièce est noté L. Le robot dispose également de deux cellules de soudure destinées à deux procédés différents. Le procédé 1 peut être réalisé en deux modes. Lorsqu'une pièce est soudée en mode A, le robot manipulateur doit la tenir tout au long de l'opération de soudure (valable pour les deux procédés). Lorsqu'une pièce

est soudée en mode B, elle est fixée sur une plateforme nommée "positionneur". Fixer une pièce accroît le temps de total de l'opération, mais permet au robot d'être libre et de traiter, en parallèle, une autre pièce nécessitant le procédé 2. L'unique mode pour le procédé 2 est nommé mode C. Certaines pièces nécessitent uniquement l'un des deux procédés, mais certaines ont besoin des deux (parfois même plusieurs fois). Une pièce possède donc une séquence de plusieurs opérations. À chaque mouvement entre les trois stations de chargement et les cellules de soudure, le robot manipulateur prend un temps nommé M. L'objectif général est de réduire le retard moyen des pièces. En cas d'arrivée imprévue d'une pièce urgente, un réordonnancement est effectué. Ce réordonnancement tient compte de l'historique des chargements et des temps de déchargement pour évaluer si une interruption est avantageuse. Il s'agit donc d'un ordonnancement dynamique par régénération.

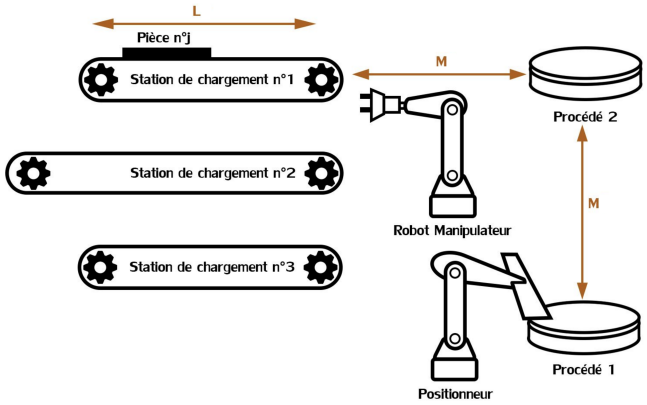

Figure 1. Illustration générale du cas d'étude

Dans cet article, nous proposons une nouvelle formulation mathématique pour l'ordonnancement du robot, ainsi qu'une approche de résolution heuristique dédiée. La conception d'une formulation nouvelle a été motivée par trois principales limites des modèles d'ordonnancement industriels proposés dans la littérature scientifique. L'exécution des opérations de chargement et de soudure peut être modélisée sous la forme d'un problème d'ordonnancement d'atelier flexible, ou Flexible Job Shop Scheduling Problem (FJSSP) multi-modes, dynamique, sans préemption et avec réutilisation de ressources à capacité unique (Nouri et al., 2016; Abdolrazzagh-Nezhad and Abdullah, 2017). En effet, les gammes opératoires varient et une même pièce peut nécessiter plusieurs fois la même opération. La flexibilité est due à la présence de plusieurs stations de chargement. De plus, le contrôle du bras manipulateur partage certaines caractéristiques clés avec le problème d'ordonnancement de ponts roulants, ou Hoist Scheduling Problem (HSP) : (i) l'inter-blocage entre les opérations ; (ii) les différents états du bras : occupé dans une position, libre dans une position, ou en déplacement entre deux positions ; ou encore, (iii) les durées dues aux mouvements du bras. Plus particulièrement, la classification proposée par Manier and Bloch (2003) présente une version hybride entre FJSSP et HSP similaire à notre problème : le HSP dynamique, multi-produits, avec duplication des stations et re-circulations. Cependant, ce modèle a lui-même trois limites face au problème étudié. Premièrement, les stations de chargement doivent être libres durant tout le parcours d'une pièce et réutilisées pour le déchargement. Dans ce contexte, au maximum trois pièces peuvent entrer dans le système même si les stations semblent libres. Une fois les pièces d'un HSP placées dans une station pour être exécutées, le palan est libre d'aller déplacer d'autres pièces. Au contraire, le bras du robot peut rester bloqué pour tenir une pièce durant son exécution. Finalement, notre problème intègre plusieurs contraintes supplémentaires telles que la limite de taille des pièces acceptées par certaines stations ou les durées de positionnement des pièces exécutées en parallèle.

Le reste de l'article est organisé comme suit : La section 2 présente la formulation mathématique du modèle d'optimisation. La section 3 détaille une méthodologie de résolution basée sur une méthode de recherche heuristique. La section 4 comporte une série de tests et expérimentations pour valider la qualité de notre méthodologie. On y présente également un prototype architectural d'implémentation du système. Finalement, la section 5 conclut l'article, fait le bilan et propose des axes d'amélioration.

2. FORMULATION DU PROBLÈME

Dans cette section, nous modélisons le problème sous la forme d'un programme linéaire en nombres entiers (ou MILP). Le tableau 1 documente l'ensemble des paramètres (constantes) utilisés par le modèle ainsi que les variables de décision. Les principaux types de données sont les pièces à traiter, leurs opérations, les deux procédés, les trois modes d'exécution et les trois stations de chargement. Le modèle peut facilement être adapté pour d'autres modes, procédés ou stations. Les principales décisions sont (i) l'affectation des pièces aux stations; (ii) la sélection d'un mode d'exécution et le choix du parallélisme; (iii) la séquence du robot; (iv) les dates de chargement et d'exécution; et (v) les possibles déchargements de pièces en cours. Finalement, l'équation (1) représente l'objectif à minimiser sous les contraintes (2) à (21), soit le retard réel total Z (ce qui équivaut à minimiser le retard moyen).

$$\min Z = \sum_{p=1}^{n} \delta_p \tag{1}$$

L'annexe A liste les déplacements du bras mécanique ainsi que les temps de chargement/déchargement. Les contraintes (2) forcent la présence d'une relation de précédence unique entre deux opérations. Les contraintes (3)–(7) calculent la date d'exécution au plus tôt d'une opération en tenant compte des opérations qui la précèdent (dans la structure de la pièce ainsi que dans la séquence du robot), des modes et possibles parallélismes, ainsi qu'aux temps dus aux mouvements, positionnement en mode B ou déchargements. Avec les contraintes (8) seules les opérations de procédé 1 sont réalisées en mode A ou B, les opérations de procédé 2 sont réalisées en mode C. Dans le modèle mathématique, les modes sont dénotés de manière numérique de 1 à 3. Grâce aux contraintes (9), la première opération ne démarre qu'une fois la pièce chargée. L'attribution unique d'un mode à une opération est formulée dans les contraintes (10). Les contraintes (11) limitent l'exécution en parallèle aux opérations de

Tableau 1. Paramètres et variables de décision

Ensembles	
P	Pièces à traiter. On note $\|P\| = n$
O	Opérations à réaliser
$O_p \subseteq O$	Opérations liées à la pièce $p \in P$
W	Procédés (dans notre cas $\|W\| = 2$)
E	Modes d'exécution (dans notre cas $\|E\| = 3$)
C	Stations de chargement (dans notre cas $\|C\| = 3$)
Paramètres	
$a_o^w \in [0,1]$	1 si l'opération $o \in O$ nécessite le procédé $w \in W$, 0 sinon
$l_p \in [0,1]$	1 si la pièce $p \in P$ est large, 0 sinon
$f_p \in O_p$	La première opération de la pièce $p \in P$
$v_o \in O_p$	L'opération qui précède l'opération $o \in O_p$, $p \in P : o \neq f_p$
$d_p \in \mathbb{R}^+$	Date due de la pièce $p \in P$
$t_o \in \mathbb{R}^+$	Durée de soudure de l'opération $o \in O$
$b_p \in \mathbb{R}^+$	Durée pour positionner la pièce $p \in P$ en mode B
$L \in \mathbb{R}^+$	Durée de chargement/déchargement sur l'une des trois stations
$M \in \mathbb{R}^+$	Durée d'un mouvement du robot manipulateur
$I \in \mathbb{R}^+$	Borne supérieure: $I = \sum_{p=1}^{n} \sum_{o=1}^{\|O\|} (t_o + b_p + 3M + 2L)$
Données historiques pour l'ordonnancement dynamique	
$lh_p^c \in [0,1]$	1 si la pièce $p \in P$ était chargée dans la station $c \in C$, 0 sinon
$bh_p \in [0,1]$	1 si la pièce $p \in P$ était positionnée en mode B, 0 sinon
$wh_p \in [0,1]$	1 si la pièce $p \in P$ était tenue par le robot, 0 sinon
Variables de décision	
$\beta_p^c \in \mathbb{R}^+$	Date d'entrée de la pièce $p \in P$ dans la station c
$\delta_p \in \mathbb{R}^+$	Retard de la pièce $p \in P$
$\eta_o \in \mathbb{R}^+$	Début de l'exécution de l'opération $o \in O$
$\gamma_p^c \in [0,1]$	1 si la pièce $p \in P$ est chargée dans la station c, 0 sinon
$\omega_o^m \in [0,1]$	1 si l'opération $o \in O$ est exécutée en mode m, 0 sinon
$\phi_{o,o'} \in [0,1]$	1 si l'opération $o \in O$ est exécutée avant l'opération $o' \in O$, 0 sinon
$\psi_o \in [0,1]$	1 si l'opération $o_1 \in O$ (procédé 2) est exécutée en parallèle d'une opération exécutée en mode B
$\varphi_p^c \in [0,1]$	1 si la pièce $p \in P$ est déchargée de la station c, 0 sinon

procédé 2 (et donc au mode C). L'attribution unique d'une station de chargement est garantie par les contraintes (12). Les contraintes (13) forcent l'utilisation de la station 2 aux pièces larges. Les contraintes (14)–(15) calculent le retard d'une pièce. Ce retard tient compte de la fin de la dernière opération d'une pièce, de sa date due et de l'état du robot qui doit la décharger. Les contraintes (16)–(19) calculent la date possible de chargement d'une pièce dans une station de chargement. Cette date tient compte des pièces précédentes et de l'état du robot manipulateur. Les contraintes (18)–(19) sont essentielles uniquement dans le cas dynamique et tiennent compte de l'historique des pièces en cours (possiblement à décharger). Les contraintes (20) forcent le déchargement d'une pièce si sa précédence envers une pièce non chargée a changé. Finalement, les contraintes (21) stipulent qu'une fois chargée, une pièce doit tout de même attendre la mise à disposition du robot manipulateur.

$$1 = \phi_{o,o'} + \phi_{o',o} \quad \forall (o,o') \in O^2 : o \neq o' \tag{2}$$

$$\eta_o \geq end(v_o) + M(3\psi_{v_o} + 1) \quad \forall o \in O_p,\ \forall p \in P : o \neq f_p \tag{3}$$

$$\eta_o \geq end(o') + 2M - I(1 + \omega_{o'}^2 - \phi_{o',o}) \quad \forall (o,o') \in O^2 : o \neq o' \tag{4}$$

$$\eta_o \geq end(o') + 2M - I(1 + \omega_o^3 - \phi_{o',o}) \quad \forall (o,o') \in O^2 : o \neq o' \tag{5}$$

$$\eta_o \geq end(o') + 2M - I(1 - \phi_{o',o} - \psi_o) \quad \forall (o,o') \in O^2 : o \neq o' \tag{6}$$

$$\eta_o \geq \eta_{o'} + (b_p \omega_{o'}^2 + 2M) \times (1 - bh_p) - I(1 - \phi_{o',o}) \quad \forall p \in P,\ \forall o \in O_p,\ \forall o' \in O : o \neq o' \tag{7}$$

$$\psi_o \geq a_o^2 \quad \forall o \in O \tag{8}$$

$$\eta_{f_p} \geq \sum_{c=1}^{3} \left(\beta_p^c - M \times lh_p^c \times (1 - \varphi_p^c) \times (bh_p + wh_p) \right) + M \quad \forall p \in P \tag{9}$$

$$1 = \sum_{m=1}^{3} \omega_o^m \quad \forall o \in O \tag{10}$$

$$\omega_o^3 = a_o^2 \quad \forall o \in O \tag{11}$$

$$1 = \sum_{c=1}^{3} \gamma_p^c \quad \forall p \in P \tag{12}$$

$$\gamma_p^2 \geq l_p \quad \forall p \in P \tag{13}$$

$$\delta_p \geq end(o) + L + M - d_p \quad \forall p \in P, \forall o \in O_p \tag{14}$$

$$\delta_p \geq free(o, o'') + L + 3M - d_p \quad \forall (p,p') \in P^2,\ \forall o \in O_p,\ \forall o' \in O_{p'} : p \neq p' \tag{15}$$

$$\beta_p^c \geq end(o') - prec(p', p, c) + 2L + M \quad \forall o' \in O_{p'},\ \forall (p,p') \in P^2 : p \neq p' \tag{16}$$

$$\beta_p^c \geq free(o', o'') - prec(p', p, c) + 2L + 3M \quad \forall o' \in O_{p'}, \forall o'' \in O_{p''},\ \forall (p,p',p'') \in P^3 : p \neq p' \neq p'' \tag{17}$$

$$\beta_p^{c'} \geq \sum_{c=1}^{3} \left(\varphi_p^c \left(M \times wh_p + 3M \times bh_p + 2L \right) \right) - I(1 - \gamma_p^{c'}) \quad \forall p \in P,\ \forall c' \in C \tag{18}$$

$$\beta_p^c \geq lh_{p'}^c (2L + M \times wh_{p'} + 3M \times bh_{p'}) - I(1 - \gamma_p^c) \quad \forall (p,p') \in P^2,\ \forall c \in C : p \neq p' \tag{19}$$

$$1 \leq \varphi_p^c + I(3 - lh_p^c - \phi_{f_{p'},f_p} - \gamma_{p'}^c) \quad \forall (p,p') \in P^2,\ \forall c \in C : p \neq p' \tag{20}$$

$$\eta_o \geq \sum_{p=1}^{n} \sum_{c=1}^{3} \left(\varphi_p^c (M \times wh_p + 2M \times bh_p) \right) \quad \forall o \in O \tag{21}$$

Afin de faciliter la lecture, les contraintes étudiées ci-dessus utilisent trois notations agrégées. $prec(p, p', c)$ permet de vérifier la précédence entre deux pièces p et p' dans la station de chargement $c \in C$. Cette précédence vérifie à la fois la bonne sélection de la station ainsi que la précédence entre les deux premières opérations des pièces dans la séquence du robot.

$$prec(p, p', c) = I(3 - \phi_{f_p,f_{p'}} - \gamma_p^c - \gamma_{p'}^c) \tag{22}$$

$end(o)$ permet de calculer la date de fin d'exécution d'une opération $o \in O$. Cette date tient compte de la date de démarrage de l'opération, de son temps d'exécution et du temps de positionnement si elle est exécutée en mode B. Ce temps de positionnement est ignoré s'il s'agit de la

première opération et que la pièce est déjà positionnée (contexte dynamique).

$$end(o) = \begin{cases} \eta_o + t_o + (b_p \times \omega_o^2) \times (1 - bh_p) & \text{Si } o = f_p \\ \eta_o + t_o + (b_p \times \omega_o^2) & \text{Si } o \in O_p \setminus f_p \end{cases} \quad (23)$$

Une pièce p pourrait avoir terminé toutes ses opérations, mais ne peut être libérée à travers la station. Ce cas arrive lorsque la dernière opération a été exécutée en mode B et que le robot est occupé à tenir une autre pièce p' ayant une opération o' en mode C. $free(o, o')$ permet ainsi de modéliser la date possible de libération de la pièce p. Ainsi, la date réelle de fin de l'opération o = MAX($end(o)$, $free(o, o')$). $free(o, o')$ vérifie également que o' est un successeur direct de o. Pour cela, on vérifie qu'il n'y a pas d'opération séquentielle entre o et o'.

$$free(o, o') = \begin{cases} end(o') - I(3 - \phi_{o,o'} - \omega_o^2 - \omega_{o'}^3) & \text{Si } n = 2 \\ end(o') & \\ -I\Big(4 - \phi_{o,o'} - \omega_o^2 - \omega_{o'}^3 - \psi_{o'} + & \\ \displaystyle\sum_{q \in P \setminus p,p'} \sum_{x \in O_q} a_x^1(\phi_{o,x} + \phi_{x,o'} - \phi_{o,o'})\Big) & \text{Sinon} \end{cases} \quad (24)$$

3. APPROCHE D'OPTIMISATION

Dans cette section, nous proposons une méthode heuristique pour résoudre le problème. Une des principales contraintes émises par notre partenaire étant le temps de calcul, notre méthode se limite à une complexité théorique polynomiale par rapport au nombre de pièces et opérations à réaliser: variant en $\Theta(|P| \times |O|^2)$ et $\Theta(|P| \times |O|^3)$ au pire des cas. Notre méthode se distingue des méta-heuristiques plus classiques (Gradient Descent, Tabu Search, etc.) par son exploitation des particularités du problème étudié (gain de temps grâce au parallélisme, contraintes sur la largeur des pièces ou sur les modes, etc.). Cette exploitation permet à la fois d'éviter la violation de contraintes, mais aussi d'orienter la recherche vers des décisions considérées préalablement comme avantageuses. L'objectif principal étant de minimiser le retard, la première idée est d'exécuter les pièces dans l'ordre de leurs dates dues. C'est donc à l'aide de cette règle de priorité que sont initialement construites les solutions. Cependant, paralléliser certaines opérations permet de réduire le makespan et les retards. Ce parallélisme n'est parfois possible qu'en enfreignant la règle de priorité utilisée au départ. La première étape d'amélioration consiste donc à changer de mode et réordonnancer à la recherche de parallélisme entre une opération de procédé 1 suivie d'une ou plusieurs opérations de procédé 2 (tout en respectant l'ordre des opérations d'une pièce et la disponibilité du robot ou des stations). De plus, hormis l'ordre des opérations, il peut être préférable de modifier l'ordre des pièces. Cette seconde étape de réordonnancement permet également de réduire le retard grâce au parallélisme, mais également du fait des variations entre les temps d'exécution.

L'algorithme 1 décrit la structure générale de notre méthode. Cette dernière est donc une méthode hybride entre une heuristique de construction et une recherche locale exploitant deux voisinages : un changement de mode (et parfois une mutation synchronisée du mode et de l'ordre entre deux opérations) ainsi que la permutation de deux pièces. Les deux phases d'amélioration locale ne sont pas exécutées séquentiellement mais de manière intégrée. Par exemple, après avoir permuté deux pièces, on peut également changer leurs opérations de mode pour activer ou annuler du parallélisme.

Algorithme 1 Structure générale de la méthode

PREMIÈRE PHASE: Construction d'une solution réalisable
1: Trier les pièces selon l'ordre croissant des dates dues
2: En cas d'égalité, trier les opérations concernées dans l'ordre croissant des durées d'exécution
3: Choisir uniquement des exécutions en mode A (procédé 1) et C (procédé 2)
4: Si le retard n'est pas nul, exécuter la phase 2

SECONDE PHASE: Permutation des opérations et changement de mode
1: Pour toute opération en mode A, tester si un changement en mode B réduit les retards (en utilisant la simulation, voir Algorithme 2)
2: Si le changement ne modifie pas le retard (pas de dégradation ni amélioration), favoriser le mode B
3: Afin de créer du parallélisme, le changement en mode B peut également signifier une permutation dans l'ordre des opérations avec les pièces adjacentes. Ces changements tiennent compte des pièces chargées et de l'ordre des opérations des pièces.
4: Une fois tous les parallélismes testés, et uniquement si le retard n'est pas nul, exécuter/continuer la phase 3

TROISIÈME PHASE: Permutation de deux pièces consécutives
1: Pour chaque changement, effectuer la phase 2 sur les opérations concernées uniquement
2: S'arrêter lorsque toutes les paires consécutives (et nouvelles paires) ont été testées ou que le retard est nul

Dans l'algorithme 1, on peut remarquer que notre méthode ne prend que trois types de décisions : l'ordre des pièces, l'ordre des opérations (en fonction de l'ordre des pièces) et le mode d'exécution. Certaines décisions telles que le choix d'une station et date de chargement, les possibles déchargements, les mouvements du robot ou encore les dates d'exécution sont ainsi déduites durant l'étape dite de "Simulation" (Algorithme 2). Cette étape permet également de mesurer les retards et donc d'évaluer la qualité de la solution actuelle. Exécutée à chaque nouvelle décision, cette étape de simulation doit être rapide.

4. EXPÉRIMENTATIONS

Cette section comporte plusieurs expérimentations permettant d'implémenter notre méthode, de tester sa capacité à s'adapter à certains cas connus ainsi que ses performances sur une instance de taille réaliste.

4.1 Environnement et prototype architectural

Dans un premier temps, nous avons implémenté notre modèle d'optimisation (à l'aide de IBM ILOG CPLEX 12.8) et notre méthode (en utilisant Java) et avons ajouté les composants nécessaires pour obtenir un prototype de système complet. L'architecture logicielle de ce prototype est illustrée à la figure 2. Comme on peut le voir, hormis les classes d'interface utilisateur, le premier module fonctionnel concerne la lecture et interprétation des données. Les fichiers de données générés par notre partenaire étant de type tableur (.xlsx), le module "*Parsing*" permet de traduire ces informations dans notre modèle de données.

Algorithme 2 Simulation récursive d'une solution

Initialisation: Date = 0
Début:

Date ← durée des mouvements du robot pour sortir les pièces non urgentes qui sont en contradiction avec la séquence choisie: M en cas de pièce unique, $3M$ en cas de double pièces
Libérer les stations de chargement (état ← libre). Selon l'état actuel du robot, cela prend 0, L, $M + L$ ou $3M + L$ unités de temps
Pour chaque pièce $p \in P$ **Faire**
  Si p n'est pas déjà chargée **alors**
    Charger, selon sa largeur, dans la première station disponible
    état de la station ← occupé
    Date ← Date + L
  Fin Si
  Pour chaque pièce $o \in O_p$ **Faire** ▷ Début alternatif
    Date ← Date + M à 3M (selon le robot) + b_p (si mode B)
    Exécuter au plus tôt
    Si o est en parallèle d'une opération o' en mode B **alors**
      Calculer la date fin de l'opération o
      Relancer la simulation à partir de o' et de cette date
    Fin Si
    Mettre à jour la date courante
  Fin Pour
  Sortir la pièce p et libérer la station
  Date ← Date + M à 4M (selon le robot) + L
  Calculer le retard δ_p de la pièce
Fin Pour
Calculer le retard total (somme)

Retourner l'ordonnancement final
Fin

Notre modèle de données est construit selon les principes de la Programmation Orientée Objets (POO). Ainsi le module "*Data*" est composé de six principales classes : (i) une instance de problème qui contient (ii) une liste de pièces ; chaque pièce contenant elle-même (iii) une liste d'opérations. Ensuite, l'héritage est exploité pour construire (iv) une solution liée à une instance, qui contient; (v) une liste de pièces ordonnancées et (vi) une liste d'opérations ordonnancées. Le module "*Solver*" correspond à l'implémentation de notre modèle pour une résolution exacte, et finalement, le module "*Heuristic*" correspond à l'implémentation de notre méthode. L'étape de simulation utilise également trois modules nommés "*Managers*" afin de gérer (i) les trois stations de chargement, (ii) l'état du robot manipulateur et (iii) la date courante. Ces modules de gestion permettent de tenir à jour l'état de ces différents éléments au fil de la simulation et de prendre les décisions secondaires (affectation d'une station, déplacement du robot, déchargement d'une pièce, etc.) tout en garantissant la faisabilité des dates retournées.

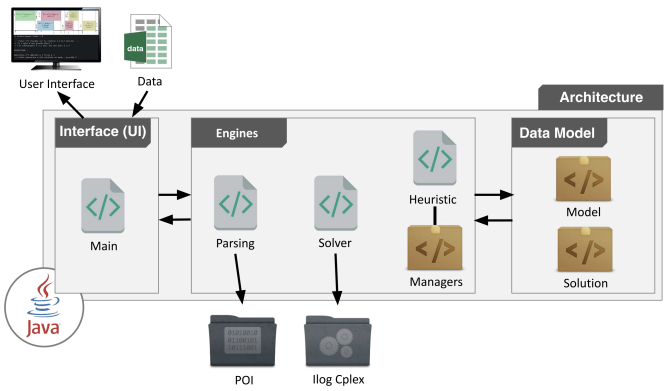

Figure 2. Architecture du système proposé

4.2 Validation des cas particuliers

Cette première analyse permet de valider le bon fonctionnement des deux étapes d'amélioration sur des instances de petite taille, comportant différents cas particuliers. Dans les tableaux de données et figures qui suivent, les valeurs sont données en secondes. Pour ces instances, aucune information sur la déviation n'est utile, car l'heuristique trouve la solution optimale (et aucun retard). De la même manière, les deux méthodes (exacte et heuristique) ont résolu ces instances dans un temps inférieur à 0.35 seconde. L'ensemble des tests ont été réalisés sur un ordinateur comportant 16 Go de mémoire RAM (dont 2 Go alloués à l'exécution du prototype en Java et 2 Go alloués à CPLEX) et un processeur Intel i7 de septième génération.

Le tableau 2 présente les premières instances testées. Ces dernières sont statiques et ne tiennent pas compte de l'historique. Nous souhaitons principalement vérifier (i) la bonne utilisation du parallélisme et (ii) la réorganisation des pièces et la nécessaire violation de la règle de priorité initialement utilisée. Les colonnes "*Proc. 1*" et "*Proc. 2*" décrivent les temps nécessaires par type de procédé. Les colonnes "*Op. 1*" et "*Op. 2*" précisent le type de procédé requis pour la première et la seconde opération. La colonne "*DD*" signifie Date Due. La colonne "*Pos.*" détaille le temps nécessaire pour positionner la pièce. Finalement, la colonne "*Large*" précise si la pièce est large ou non.

Tableau 2. Instances pour les cas statiques

Instance 1							
#	Pro. 1	Pro. 2	Large	DD	Op.1	Op.2	Pos.
1		6.0	non	7.9	2		0.5
2	5.0	2.0	oui	10.8	1	2	0.5
3	1.5		oui	13.3	1		0.5
Instance 2							
#	Pro. 1	Pro. 2	Large	DD	Op.1	Op.2	Pos.
1		6.0	non	6.8	2		0.5
2	5.0		oui	15.1	1		0.5
3	2.0	6.0	non	20.4	2	1	0.5
4	4.0	3.0	oui	25	2	1	0.5

Les figures 3 et 4 présentent les résultats obtenus sous la forme de diagrammes de Gantt. Afin de les comprendre, il faut préciser que dans l'ensemble des instances $L = 0.2$ et $M = 0.3$. De plus, les temps de positionnement des pièces 1 et 2 sont respectivement représentés comme "*P1*" et "*P2*".

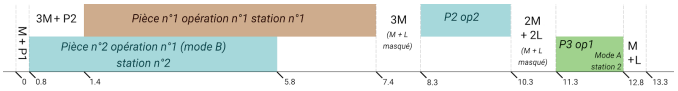

Figure 3. Exécution de l'instance 1

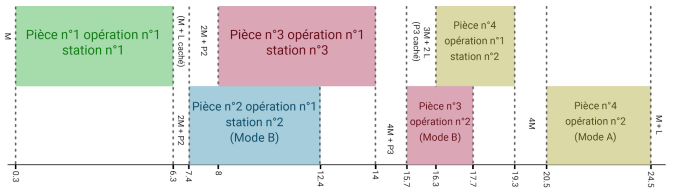

Figure 4. Exécution de l'instance 2

Le tableau 3 permet de vérifier la prise de bonnes décisions (déchargement) dans le contexte dynamique. L'instance

3 est également la seule qui nécessite la seconde phase d'amélioration. Cette instance ne contient que des pièces ayant une seule opération restante. La colonne "*Bloquée*" permet de préciser le numéro de la station de chargement initialement bloquée par chaque pièce. La colonne "*état*" permet aussi de spécifier l'état de la pièce. L'état "*EO*" signifie que la pièce n'est pas encore dans le système; "*E1*" indique que la pièce est en station de chargement; "*E3*" signifie qu'elle est tenue par le robot (Mode A ou C); finalement, "*E3*" indique qu'elle est sur le positionneur (Mode B). Le résultat de l'exécution de cette instance est présenté à la figure 5.

Tableau 3. Instance pour le cas dynamique

Instance 3							
#	Pro.	Op.	Large	DD	Bloquée	état	Pos.
1	6.0	1	non	6.5	1	E3	0.5
2	5.0	2	ou	12.1	2	E1	0.5
3	2.0	2	oui	3.5	aucune	E0	0.5

Comme on peut l'observer, les temps relatifs aux mouvements du robot et aux stations de chargement sont respectés. On peut aussi observer la bonne utilisation des modes et du parallélisme (qui engendre bien un temps supplémentaire de positionnement). Le mode B est aussi évité lorsqu'il n'est pas bonifiant (dernière tâche des instances 1 et 2). On peut également voir que les pièces de l'instance 3 ne sont plus ordonnées selon la date due (pièce 3 avant la pièce 2). Dans cette même instance, on peut finalement observer que l'état initial des pièces est bien pris en compte.

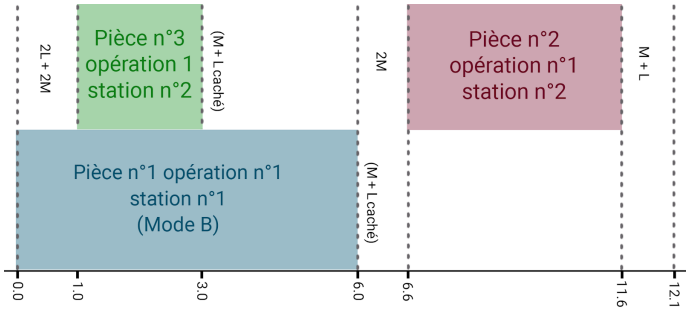

Figure 5. Exécution de l'instance 3

4.3 ***Tests sur un problème de plus grande taille***

Basé sur une combinaison des données de 81 pièces (et 98 opérations) fournies par notre partenaire industriel, nous avons généré une instance aléatoire de taille plus réaliste : 160 pièces et 195 opérations. La résolution heuristique de cette instance a nécessité uniquement 0.99 seconde et une exploitation négligeable de la mémoire. À cette échelle, tandis que le nombre d'opérations à été multiplié par 40 (en comparaison avec l'instance 2), le temps de calcul a n'a été multiplié que par 3. Par ailleurs, l'ensemble du processus nécessite moins de 2 secondes, de la lecture du fichier de données à l'affichage des résultats. Le système final peut ainsi être considéré comme temps réel. En contrepartie, la résolution exacte a échoué, après un peu plus de 30 minutes à cause de la limite de mémoire allouée.

5. CONCLUSION

Dans cet article, nous avons formulé un MILP représentant le comportement d'un robot industriel. Nous avons également proposé une méthode heuristique permettant de résoudre en temps réel une instance de taille réaliste, avec peu de mémoire et sous une complexité polynomiale. Notre méthode tient compte des différentes capacités et décisions du robot (séquence et dates, modes et parallélisme, annulations et déchargements, affectations des stations, etc.). De par la structure de sa phase de simulation au plus tôt, notre méthode réduit également le makespan dans la mesure où cela n'affecte pas les retards. Notre méthode a également l'avantage d'être facilement adaptable et permet de modifier les valeurs de L, M, ainsi que le nombre de stations et de procédés de soudures. Modifier la règle de priorité initiale pourrait également permettre de favoriser le makepsan, le nombre de mouvements du robot ou l'équilibre entre les stations. Finalement, l'architecture du prototype de système est organisée de manière modulaire afin d'accroître son évolutivité.

Notre méthode garantit la faisabilité de la solution proposée mais pas son optimalité. Il est ainsi possible d'obtenir des solutions sous-optimales. Plusieurs améliorations pourraient permettre de réduire ce phénomène. Premièrement, il serait intéressant d'accroître le nombre de positions testées. Notre troisième phase permet de reculer une pièce de plusieurs positions, mais d'avancer que d'une seule position. On pourrait soit relancer la troisième phase plusieurs fois (avec une complexité $\Theta(\varrho|P| \times |O|^3)$ pour ϱ relances) soit rechercher à chaque pièce sa meilleure position (en tenant compte des décisions précédentes). Cependant cette dernière option engendrerait une complexité de $\Theta(|P|^2 \times |O|^3)$ au pire des cas. De manière générale, il serait avantageux de permettre à l'utilisateur de configurer (en fonction de l'instance) le compromis entre vitesse et qualité des solutions, et cela, pour chaque phase d'amélioration.

Hormis la méthode de résolution, le projet évolue aujourd'hui autour de deux principaux axes de recherche. Premièrement, le modèle est lui-même sujet à amélioration. L'objectif optimisé, la diminution des retards, était la priorité de l'entreprise partenaire. Cependant, d'autres objectifs, notamment liés à la qualité des produits et à l'impact écologique, sont considérés comme importants. Il serait également souhaitable que le modèle offre un contrôle plus précis des mouvements du robot ainsi qu'une prise en compte de contraintes liées à son intégration dans une chaîne de production plus large. Deuxièmement, il est aujourd'hui essentiel de mener une expérimentation plus approfondie afin de mieux évaluer la performance des modèles et approches proposés. Tout d'abord, mesurer la déviation envers la solution optimale pourrait être obtenue en exécutant le modèle mathématique sur des serveurs de calcul aux capacités plus larges. Ensuite, un générateur d'instances aléatoires et paramétrables (taille en nombre de pièces, composition des pièces, niveau de parallélisme possible) aiderait à évaluer cette déviation et les temps de calcul dans différents contextes. La méthode ayant d'ores et déjà été testée sur des instances réelles, il ne semble pas que la taille de l'instance soit une limite.

ANNEXE A: MOUVEMENTS DU ROBOT

D1	Déchargement d'une pièce (en mode A ou C) : $M + L$
D2	Déchargement d'une pièce en mode B (sur le positionneur, bras occupé) : D1 + M
C1	Chargement d'une pièce (station vide, robot libre) : L
C2	Chargement d'une pièce (robot occupé) : D1 ou D2 + L
E1	Exécution d'une opération (sauf la première, station vide, robot libre) : M
E2	Exécution d'une opération (sauf la première, robot occupé) : D1 ou D2 + M
E3	Exécution de la première opération d'une pièce : E1 ou E2 + L
S1	Changement d'opération pour une pièce seule (en mode A ou C) : M
S2	Changement d'opération pour une pièce en mode B (sur le positionneur, bras occupé par une autre pièce) : $3M$
S3	Changement d'opération pour une pièce en mode C (tenue par le robot, positionneur occupé par une autre pièce) : $4M$

REMERCIEMENTS

Nous tenons à remercier la société Alstom pour son étroite collaboration, pour avoir proposé le sujet de recherche et avoir partagé des données de test.

Vidéo de présentation du projet :
www.youtube.com/watch?v=wMU39mVTmOg

REFERENCES

Abdolrazzagh-Nezhad, M. and Abdullah, S. (2017). Job shop scheduling: Classification, constraints and objective functions. *International Journal of Computer and Information Engineering*, 11(4), 429–434.

Alcácer, V. and Cruz-Machado, V. (2019). Scanning the industry 4.0: A literature review on technologies for manufacturing systems. *Engineering science and technology, an international journal*, 22(3), 899–919.

Huang, H. and Li, B. (2009). Development of motion type reconfigurable modular robot for multi-task application. In *2009 International Conference on Information and Automation*, 1386–1391. IEEE.

Julien, F., Magazine, M.J., Hall, N.G., et al. (1997). Generalized preemption models for single-machine dynamic scheduling problems. *IIE transactions*, 29(5), 359–372.

Larsen, R. and Pranzo, M. (2019). A framework for dynamic rescheduling problems. *International Journal of Production Research*, 57(1), 16–33.

Manier, M.A. and Bloch, C. (2003). A classification for hoist scheduling problems. *International Journal of Flexible Manufacturing Systems*, 15, 37–55.

Nouri, H.E., Driss, O.B., and Ghédira, K. (2016). A classification schema for the job shop scheduling problem with transportation resources: state-of-the-art review. In *Artificial Intelligence Perspectives in Intelligent Systems: Proceedings of the 5th Computer Science On-line Conference 2016 (CSOC2016), Vol 1*, 1–11. Springer.

Valente, A. (2016). Reconfigurable industrial robots: A stochastic programming approach for designing and assembling robotic arms. *Robotics and Computer-Integrated Manufacturing*, 41, 115–126.