Automatisation et amélioration de performances d'application pick & place multi-robots

Gaël Humbert

N°d'ordre NNT : 2016LYSEI036

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'INSA de LYON

Ecole Doctorale N° 160
Électronique Électrotechnique et Automatique (EEA)

Spécialité de doctorat : Automatique

Soutenue publiquement le 25/04/2016, par :
Gaël Humbert

Automatisation et amélioration de performances d'applications pick & place multi-robots

---

Devant le jury composé de :

P. LUTZ	Professeur des Universités	FEMTO - Besançon	Président Rapporteur
P. CASTAGNA	Professeur des Universités	IRCCyN - IUT Nantes	Rapporteur
K. KOUISS	Maître de Conférences	Institut Pascal - SIGMA	Examinateur
M. GUILLEMOT	Maître de Conférences	LaMCoS - INSA-Lyon	Examinatrice
X. BRUN	Professeur des Universités	Ampère - INSA-Lyon	Directeur de thèse
M.T. PHAM	Maître de Conférences / HDR	Ampère - INSA-Lyon	Co-directeur de thèse
B. JACQUEMIN	Vice Président Innovation	SCHNEIDER ELECTRIC	Invité
C. LAPERIERE	Responsable ADE France	SCHNEIDER ELECTRIC	Invité
D. NOTERMAN	Maître de Conférences	DISP - INSA-Lyon	Invité

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://www.edchimie-lyon.fr** Sec : Renée EL MELHEM Bat Blaise Pascal 3e etage secretariat@edchimie-lyon.fr Insa : R. GOURDON	M. Stéphane DANIELE Institut de Recherches sur la Catalyse et l'Environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 avenue Albert Einstein 69626 Villeurbanne cedex directeur@edchimie-lyon.fr
E.E.A.	**ELECTRONIQUE, ELECTROTECHNIQUE, AUTOMATIQUE** **http://edeea.ec-lyon.fr** Sec : M.C. HAVGOUDOUKIAN Ecole-Doctorale.eea@ec-lyon.fr	M. Gérard SCORLETTI Ecole Centrale de Lyon 36 avenue Guy de Collongue 69134 ECULLY Tél : 04.72.18 60.97 Fax : 04 78 43 37 17 Gerard.scorletti@ec-lyon.fr
E2M2	**EVOLUTION, ECOSYSTEME, MICROBIOLOGIE, MODELISATION** **http://e2m2.universite-lyon.fr** Sec : Safia AIT CHALAL Bat Darwin - UCB Lyon 1 04.72.43.28.91 Insa : H. CHARLES Safia.ait-chalal@univ-lyon1.fr	Mme Gudrun BORNETTE CNRS UMR 5023 LEHNA Université Claude Bernard Lyon 1 Bât Forel 43 bd du 11 novembre 1918 69622 VILLEURBANNE Cédex Tél : 06.07.53.89.13 e2m2@ univ-lyon1.fr
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTE** **http://www.ediss-lyon.fr** Sec : Safia AIT CHALAL Hôpital Louis Pradel - Bron 04 72 68 49 09 Insa : M. LAGARDE Safia.ait-chalal@univ-lyon1.fr	Mme Emmanuelle CANET-SOULAS INSERM U1060, CarMeN lab, Univ. Lyon 1 Bâtiment IMBL 11 avenue Jean Capelle INSA de Lyon 696621 Villeurbanne Tél : 04.72.68.49.09 Fax :04 72 68 49 16 Emmanuelle.canet@univ-lyon1.fr
INFOMATHS	**INFORMATIQUE ET MATHEMATIQUES** **http://infomaths.univ-lyon1.fr** Sec :Renée EL MELHEM Bat Blaise Pascal 3e etage infomaths@univ-lyon1.fr	Mme Sylvie CALABRETTO LIRIS – INSA de Lyon Bat Blaise Pascal 7 avenue Jean Capelle 69622 VILLEURBANNE Cedex Tél : 04.72. 43. 80. 46 Fax 04 72 43 16 87 Sylvie.calabretto@insa-lyon.fr
Matériaux	**MATERIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec : M. LABOUNE PM : 71.70 –Fax : 87.12 Bat. Saint Exupéry Ed.materiaux@insa-lyon.fr	M. Jean-Yves BUFFIERE INSA de Lyon MATEIS Bâtiment Saint Exupéry 7 avenue Jean Capelle 69621 VILLEURBANNE Cedex Tél : 04.72.43 71.70 Fax 04 72 43 85 28 Ed.materiaux@insa-lyon.fr
MEGA	**MECANIQUE, ENERGETIQUE, GENIE CIVIL, ACOUSTIQUE** **http://mega.universite-lyon.fr** Sec : M. LABOUNE PM : 71.70 –Fax : 87.12 Bat. Saint Exupéry mega@insa-lyon.fr	M. Philippe BOISSE INSA de Lyon Laboratoire LAMCOS Bâtiment Jacquard 25 bis avenue Jean Capelle 69621 VILLEURBANNE Cedex Tél : 04.72 .43.71.70 Fax : 04 72 43 72 37 Philippe.boisse@insa-lyon.fr
ScSo	**ScSo*** **http://recherche.univ-lyon2.fr/scso/** Sec : Viviane POLSINELLI Brigitte DUBOIS Insa : J.Y. TOUSSAINT viviane.polsinelli@univ-lyon2.fr	Mme Isabelle VON BUELTZINGLOEWEN Université Lyon 2 86 rue Pasteur 69365 LYON Cedex 07 Tél : 04.78.77.23.86 Fax : 04.37.28.04.48

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Remerciements

Je tiens tout d'abord à remercier Xavier Brun et Minh-Tu Pham, mes directeurs de thèse pour leur encadrement, pour leur accompagnement dans ce monde qui est la recherche et spécialement pour leur aide dans la rédaction de ce mémoire et des articles.

Je tiens à remercier vivement M. Pierre Castagna et M. Philippe Lutz d'avoir bien voulu être rapporteurs de cette thèse, et ainsi que M. Khalid Kouiss qui a bien voulu participer au jury, aux côtés de Xavier Brun, de Minh-Tu Pham et de Mady Guillemot.

Merci à Claude Lapérière, pour m'avoir donné l'opportunité de réaliser ce sujet de thèse et pour m'avoir fait confiance.

Merci à Yves Le-Campion et à Alexandre Girardot, sans qui cette thèse n'aurait pas pu être réalisée.

Merci aussi à Merouan Benkeder, pour son aide précieuse et son transfert de savoir durant la première moitié de ma thèse.

Merci à l'équipe ADE France pour son accueil, pour sa sympathie et pour son expérience.

Merci à Rainer Ritschel et à son équipe, pour m'avoir accueilli durant mes déplacements en Allemagne. Les échanges entre la culture allemande et la culture française ont été très intéressants et très enrichissants.

Merci à Eric Daymier et à Mohammed Ghezal, pour leur apport à l'avancée de la thèse.

Je tiens à remercier toute l'équipe du "Labo", Mady Guillemot, Pascal Brossard et Didier Noterman, pour l'ambiance conviviale qu'ils savent entretenir au quotidien. Cela était un réel plaisir de travailler à côté de leur bureau.

Merci aux ODD, pour m'avoir permis de me changer les idées, afin de mieux réattaquer mes travaux.

Et enfin, je tiens à remercier mes parents qui ont toujours cru en moi, à Mélina pour m'avoir soutenu durant ces trois années, qui n'ont pas été de tout repos et à mes amis qui ont toujours pensé que j'étais fait pour la recherche.

Automatisation et amélioration de performances d'applications pick & place multi-robots

Résumé *Cette thèse s'inscrit dans le domaine de l'emballage et du conditionnement. À notre connaissance, dans le monde industriel et académique, il n'existe pas d'outils de simulation qui prennent en compte toutes les étapes du cycle de développement afin d'aider au dimensionnement et à l'amélioration des performances d'application pick & place multi-robots. Il existe des outils performants répondant à un seul besoin mais aucun logiciel répondant à tous. L'objectif de ces travaux est le développement d'une méthodologie utilisable, à l'aide d'un unique outil logiciel, tout au long du cycle de création d'une application pick & place. Dans un premier temps, la cinématique des différents robots ainsi que leur environnement sont abordées. Dans un second temps, les stratégies de commande, permettant le travail individuel et collaboratif des robots, sont développées. En dernier lieu, le transfert du programme de la simulation à l'expérimentation est effectué de façon automatisée. Á chaque étape, une phase de tests est initiée. Des tests comportementaux pour vérifier le fonctionnement des robots et leur interaction avec leur environnement sont effectués. Des essais en simulation des différents algorithmes sont réalisés suivant différentes configurations d'applications pick & place. Enfin, la comparaison entre les résultats obtenus en simulation et en expérimentation montre la pertinence de la démarche proposée.*

Mots clés applications pick & place, stratégies de collaboration, règles de planification individuelles, dimensionnement, outil logiciel, expérimentation.

Automation and improvement of performances of multi-robots pick & place applications

Abstract *This work belongs to the field of packaging. To best of our knowledge, in industrial and academic context, there are no digital tool that take into account all the steps of the development cycle to assist in the sizing and in the improvement of performance of multi-robots pick & place applications. The aim of these works is the development of a methodology usable, using a single software tool, throughout the creation cycle of a pick & place application. Firstly, kinematics of different robots and their environment are addressed. Secondly, control laws for individual and collaborative work of robots are developed. Finally, the transfer of the program from simulation to experimentation is carried out. At each step, a testing phase is initiated. Behavioral tests to check the robots operation and their interaction with their environment are performed. Simulation tests of the different algorithms are carried out according different configurations of pick & place application. Finally, a comparison between the result in simulation and in experimentation shows the relevance of the proposed approach.*

Key words pick & place applications, collaborative strategies, scheduling rules, sizing, software tool, experimentation.

Table des matières

Chapitre 1

Introduction générale

Contexte

Ce travail de thèse a débuté en mai 2013 et s'effectue dans le cadre d'un contrat CIFRE bilatéral entre la société Schneider Automation SAS (Schneider Electric) et le laboratoire Ampère à l'INSA de Lyon. Schneider Electric est un groupe industriel français connu pour être leader mondial dans la gestion de l'énergie, mais il possède aussi un secteur d'activités en robotique. La thèse se fait en collaboration avec la Business Unit Industry Business sous la responsabilité de M. Alexandre Girardot pour Schneider Electric, le département Méthodes pour l'Ingénierie des Systèmes sous la codirection de Xavier Brun et de Minh Tu Pham pour le laboratoire Ampère, et l'Atelier Inter-établissements de Productique (AIP) Rhône-Alpes Ouest sous la codirection de Didier Noterman et Mady Guillemot. Cette dernière entité est un centre de ressources et de compétences, doté de moyens industriels importants mis à la disposition commune de plusieurs établissements d'enseignement supérieur de la région Rhône-Alpes.

Ces travaux s'inscrivent dans le contexte de l'emballage et du conditionnement, ce secteur possède une dimension économique mondiale dont les perspectives de croissance sont estimées à 3% en moyenne pour les 3 années à venir. Deux grands segments de marché constituent ce domaine industriel : les fabricants d'équipements et les fabricants d'emballages et de contenants. Le segment des fabricants de machines présente depuis 2006 une croissance soutenue de l'ordre de 5% par an. L'Allemagne est le premier producteur exportateur, la France occupe le 6$^{\text{ème}}$ rang avec une part de marché mondial de 3% [Obs, 2012]. L'innovation méthodologique et technologique est un moyen efficace d'accroître la compétitivité des entreprises dans ce secteur, la demande des clients restant principalement motivée par une meilleure productivité et une plus grande flexibilité [OSEO, 2011] [DGCIS, 2009]. Les robots et modules robotiques de pick & place répondent parfaitement à ces exigences dès lors que l'ensemble des aspects de la mise en œuvre de ces systèmes est pris en compte [Sahin, 2005].

Problématique

Depuis quelques années, une demande forte des clients vis-à-vis de la productivité et de la flexibilité de leurs chaînes de production est constatée. Pour cette raison, les robots

et les modules robotiques de pick & place sont de plus en plus présents dans l'industrie manufacturière, en particulier dans le domaine agroalimentaire. Ce sont des applications hautes performances dont les caractéristiques des robots peuvent atteindre les valeurs suivantes : vitesse 10 m/s, accélération 100 m/s^2, précision +/-0.1 mm, répétabilité +/-0.1 mm, cycle de prise-dépose 0.40 s en moyenne. Afin d'augmenter les performances de ces applications, il est nécessaire de mieux maîtriser le dimensionnement du système de production (nombre de robots, caractéristiques des robots, capacité des stocks, vitesse des convoyeurs, etc.) tout en améliorant la gestion des flux et la gestion de la charge de travail entre les robots lorsque plusieurs sont utilisés.

Une application pick & place est composée en général de plusieurs robots installés les uns à la suite des autres prenant des produits sur un premier convoyeur et les déposant dans des cartons circulant sur un second convoyeur, voir Figure 1.1 [Schubert, 2000] [Sahin, 2005].

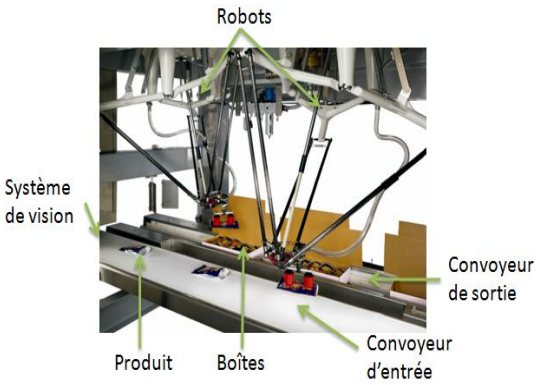

FIGURE 1.1 – Cellule robotique avec des robots Delta.

Sur une cellule d'emballage multi-robots, lorsqu'il n'existe pas de système d'amélioration de flux de production, les instructions "prendre" sont réparties équitablement entre les premiers robots. Un dernier robot est ajouté en bout de ligne afin de tenter de récupérer les produits qui n'ont pas pu être pris par les robots précédents. Des produits initialement affectés à un robot peuvent s'avérer finalement imprenables s'ils sont, par exemple, sortis de la zone de travail du robot par manque de cartons à remplir. Cette approche sans gestion de flux est largement répandue en industrie car simple d'utilisation, mais présente trois inconvénients remarqués notamment par Schneider Electric :

1. Les premiers robots ont une charge de travail individuelle importante proche de 85%, qui peut conduire à des problèmes de disponibilité.
2. À l'opposé, le dernier robot a une charge de travail de l'ordre de 25%.
3. Un grand nombre de produits ne peut être pris et est donc mis au rebut. Le taux de rebut moyen est de 1% en régime permanent mais de 5% en régime transitoire.

Dans les applications industrielles, l'expérience du terrain est couramment utilisée pour le dimensionnement des lignes de production. Pour un cahier des charges donné, incluant les performances à atteindre sur chaque flux de production, le dimensionnement de toutes les ressources et la programmation des organes de commande de tous les robots sur la chaîne de production est un problème complexe : beaucoup de paramètres sont à prendre en compte (nombre de robots, de convoyeurs, de produits, etc.). Plusieurs compétences doivent être mises à contribution : robotique, gestion de production, simulation et programmation de calculateur.

Il existe des outils théoriques comme les réseaux de Petri et des logiciels de simulation (ils seront détaillés dans la section 3.2) mais il n'existe pas encore d'outils dédiés à la fois à la simulation et à l'implémentation (c'est-à-dire des outils dont le développement en simulation peut se transposer directement sur une machine sur site). Certains outils comme

la programmation en ligne permettent le passage de la simulation à la mise en œuvre mais ne gèrent pas la programmation de haut niveau (Quels robots utiliser ? Quelles consignes à atteindre ? etc.). Ce type de fonctionnalité pourrait être intéressant pour les applications pick & place où la simulation de flux de produits et les aspects de collaboration entre plusieurs robots sont pris en compte.

Le travail de cette thèse consiste à développer une méthodologie et un outil logiciel utilisable tout au long de la création d'une application pick & place. Cette méthodologie et cet outil devront aider l'utilisateur à dimensionner un système multi-robots (nombre de robots, etc.) et à définir la stratégie de commande (algorithmes, etc.) mais aussi à la mise en œuvre sur site d'une chaîne de production en facilitant le transfert entre la simulation et l'implémentation.

Organisation du document

Ce mémoire de thèse est découpé en sept parties, la première étant cette introduction. Le chapitre 2 présente un état de l'art de manière générale puis détaillée des cellules multi-robots de pick & place. Un recensement des algorithmes existants dans la littérature est aussi présenté. Celui-ci se focalise sur des règles de planification individuelles et sur quelques heuristiques. Le chapitre 3 traite de la méthodologie développée durant ces travaux de thèse. Il recense différents outils logiciels existants pouvant être utilisés pour chaque étape de la méthodologie et il présente ensuite le nouvel outil développé durant cette thèse. Le développement d'une application pick & place faite à l'aide de l'outil logiciel est détaillé ainsi qu'une présentation de la mise en œuvre du programme développé sur deux démonstrateurs. Le chapitre 4 présente l'architecture, les blocs fonctionnels ainsi que les algorithmes développés durant les travaux de cette thèse. Le chapitre 5 traite d'une étude comparative des résultats obtenus à l'aide de l'outil logiciel suivant différentes configurations d'applications pick & place. Le chapitre 6 présente de façon plus détaillée les démonstrateurs ainsi que le passage de l'outil logiciel à l'expérimentation à l'aide d'une étude comparative. Enfinv une conclusion clôture la thèse et présente quelques perspectives pour la poursuite des travaux réalisés. Le lecteur trouvera un glossaire des termes, acronymes, et abréviations employés dans cette thèse, une liste des références bibliographiques ainsi que deux annexes.

Le travail effectué durant cette thèse a donné lieu à deux communications acceptées lors de conférences internationales [Humbert et al., 2015] et [Humbert et al., 2016b] et à deux communications en cours de révision, une dans un journal européen [Humbert et al., 2016c] et une pour une conférence internationale [Humbert et al., 2016a].

Chapitre 2

État de l'art

Sommaire

2.1 Présentation de cellules multi-robots pick & place

Nous avons indiqué en Introduction que les robots et les modules robotiques sont de plus en plus présents dans l'industrie. Cette section va présenter d'une manière générale une application pick & place générique puis détaillera chaque partie afin d'expliquer la composition et de montrer la complexité d'un système pick & place. En fin de chapitre un état de l'art sur les brevets en rapport aux systèmes pick & place sera présenté.

2.1.1 Qu'est-ce que le pick & place ?

Une application pick & place est un système qui saisit un objet afin de le déplacer d'un point A à un point B. Cette action peut être effectuée par un homme ou, dans notre contexte, par un robot. On peut séparer les applications pick & place en deux catégories principales : celles qui prennent et déposent les produits à une position fixe et celles qui prennent et déposent des objets à une position variable appelée prise et dépose en tracking (sur un convoyeur par exemple). Il existe des applications qui mixent les deux catégories (prise fixe et dépose tracking). Le travail de thèse concerne uniquement les systèmes

utilisant les prises et les déposes en tracking. Dans ces systèmes, les produits arrivent sur un convoyeur d'entrée, sont pris par un module robotisé puis sont déposés dans des boîtes se déplaçant sur un convoyeur de sortie. Le module robotisé peut être composé d'un ou plusieurs robots pick & place positionnés les uns à la suite des autres. Ce module robotisé est commandé en général par un contrôleur ou un automate programmable industriel (API). Nous allons voir maintenant les architectures matérielle et logicielle utilisées pour commander une cellule robotique.

2.1.2 Le matériel

Hardware

Une application pick & place est composée de plusieurs éléments :
— Un contrôleur.
— Des bus de terrain.
— Une Interface Homme Machine (IHM).
— Un ou plusieurs sous-modules robotisés (palettiseur, emballage etc.).
— Un ou plusieurs moteurs avec leur variateur.
— Un système de vision.

Le contrôleur est le cœur de l'architecture, il contient le programme de l'application. Il peut communiquer avec d'autres contrôleurs via des bus de terrain (Ethernet par exemple) dans de grandes chaînes de production et avec une ou plusieurs Interfaces Homme Machine. Il peut communiquer via d'autres bus de terrain (Sercos III, CANopen etc.) avec des sous-modules, comme des modules entrées/sorties, des variateurs ou un contrôleur de sécurité. Comme indiqué précédemment, il existe en général une Interface Homme Machine (IHM) qui est un pupitre permettant de piloter en ligne l'application. Le déplacement des axes des robots est géré par des servo-variateurs. Ce sont des éléments qui commandent en puissance les moteurs. Il existe des variateurs pour un seul moteur, des variateurs multiaxes et des variateurs déportés directement sur les moteurs. Le système de vision est un élément important car il fournit la position des produits sur le convoyeur. Il existe des systèmes de vision permettant de connaître certaines caractéristiques des produits comme l'orientation, la couleur ou la forme. Enfin, il peut y avoir un contrôleur de sécurité qui s'occupe uniquement de la sécurité (arrêt d'urgence via des boutons ou capteurs, des systèmes anti-pincement etc.). Un exemple d'architecture hardware est montré sur la Figure 2.1.

Software

Chaque application possède un ou plusieurs programmes implémentés dans son ou ses contrôleurs. Ce ou ces programmes ont été développés à l'aide d'un outil logiciel propre à chaque marque de contrôleurs. Ils seront détaillés dans la section 3.2.3. Ces logiciels possèdent en général plusieurs langages de programmation conforme à la norme CEI 61131-3 : le langage Ladder Diagram (LD, inspiré des schémas à relais), le langage Sequential Function Chart (SFC, inspiré du GRAFCET), le langage Function Block Diagram (FBD, langage graphique constitué de blocs), le langage Structured Text (ST, langage haut niveau comparable à du C) et le langage Instruction List (IL, langage bas niveau comparable à

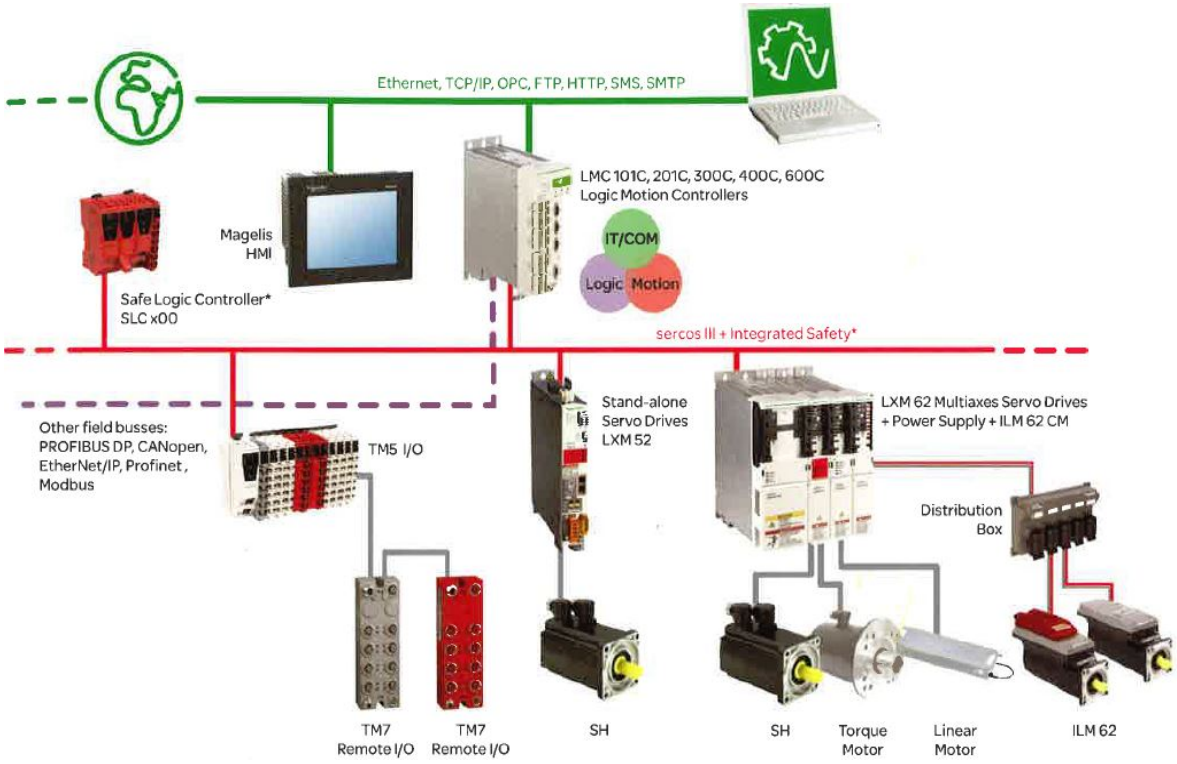

FIGURE 2.1 – Exemple d'architecture hardware.

de l'assembleur). C'est grâce à ces programmes que le contrôleur va pouvoir commander les moteurs via les variateurs pour faire fonctionner un convoyeur ou faire déplacer un robot, par exemple. Ils permettent d'effectuer des calculs afin qu'un robot puisse prendre le produit voulu, par exemple. C'est dans ces programmes que les actions sont décidées en fonction de l'évolution du contexte renseigné par les entrées. Les actions consisteront alors à activer certaines sorties ou à commander certains moteurs.

Nous avons présenté jusque-là une vue générique d'une application pick & place. La prochaine partie va détailler toutes les caractéristiques que possède un système pick & place et va montrer qu'entre deux chaînes de production, le système pick & place peut être totalement différent et donc ne peut pas être traité de la même façon.

2.1.3 Les caractéristiques d'une application pick & place

Chaque application pick & place est spécifique. Une application pick & place possède plusieurs caractéristiques qui doivent être prises en compte lors de son développement qui peuvent être regroupées en six catégories :

1. Les robots.
2. Les convoyeurs.
3. L'environnement.
4. Les conditions d'utilisation et de fonctionnement.

5. Les événements et le mode de fonctionnement.
6. Les stratégies utilisées.

Les robots

Les caractéristiques de cette catégorie peuvent être divisées en trois groupes :

1. L'espace : il définit la position du robot dans la cellule, la forme et la taille de la zone de travail.
2. La dynamique : elle spécifie la vitesse, l'accélération et le jerk des robots.
3. La cinématique du robot : elle est liée au type de robots utilisé. Il existe six familles de robots pour des applications pick & place, illustrées par la Figure 2.2.

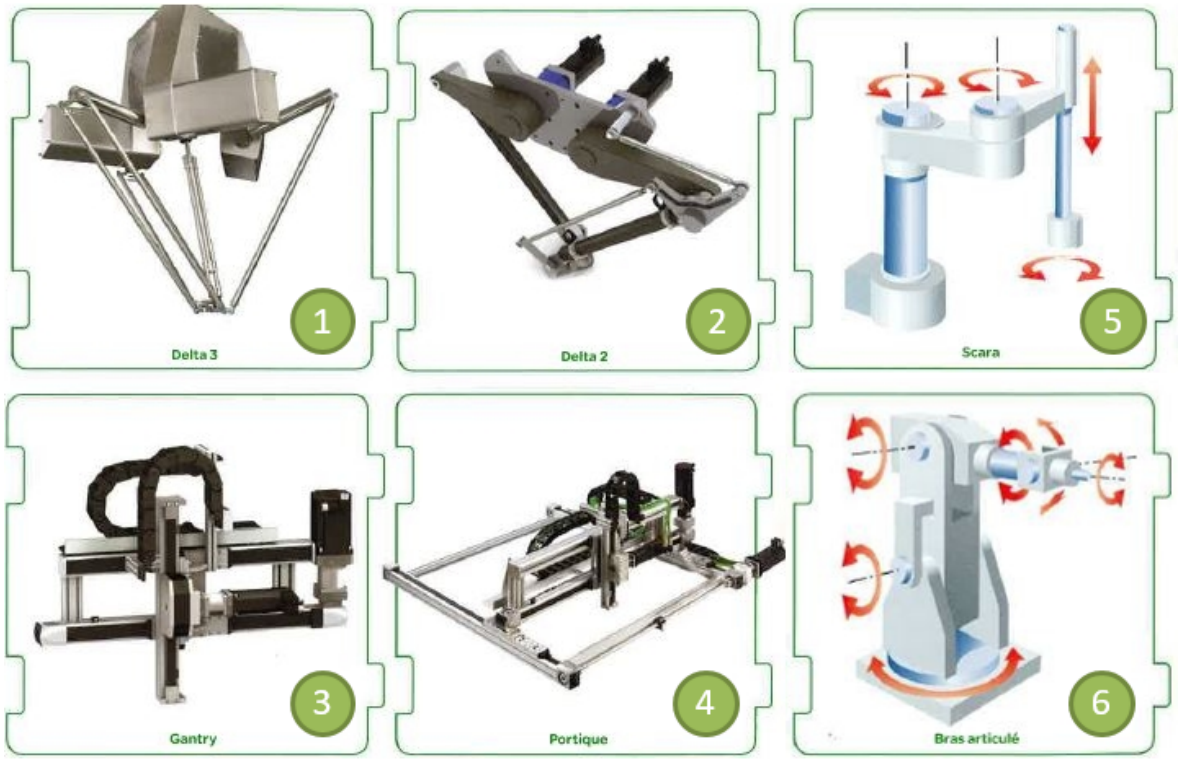

FIGURE 2.2 – Différents types de robots.

Le premier robot est le robot Delta 3, voir Figure 2.3.a. Le robot Delta fait partie de la famille des robots parallèles. C'est un robot ayant un bras de manipulation formé de 3 parallélogrammes, qui grâce à sa légèreté, lui permet d'être très rapide. Il est utilisé pour la mise en boîte dans l'agro-alimentaire (ex : confiserie) ou la pharmacie, dans l'usinage de haute précision etc. Le second robot est le robot Delta 2. Il fait partie de la même famille que le robot Delta 3 mais possède moins de degrés de liberté. Les troisième et quatrième robots sont les robots Cartésiens : Gantry et Portique. Ils ont un mouvement linéaire sur 3 axes, voir Figure 2.3.b. Le cinquième est le robot SCARA : Selective Compliance Articulated Robot Arm. Il a été développé en 1981 par les entreprises japonaises

Sankyo Seiki, Pentel et NEC. Il possède 4 axes avec trois rotations et une translation, voir Figure 2.3.c. Il est utilisé pour la manipulation de pièces dans un plan. Le dernier est le bras polyarticulé 6 axes, voir Figure 2.3.d. Il est utilisé dans l'industrie pour tous types d'applications : palettisation, peinture, usinage, chargement/déchargement etc.

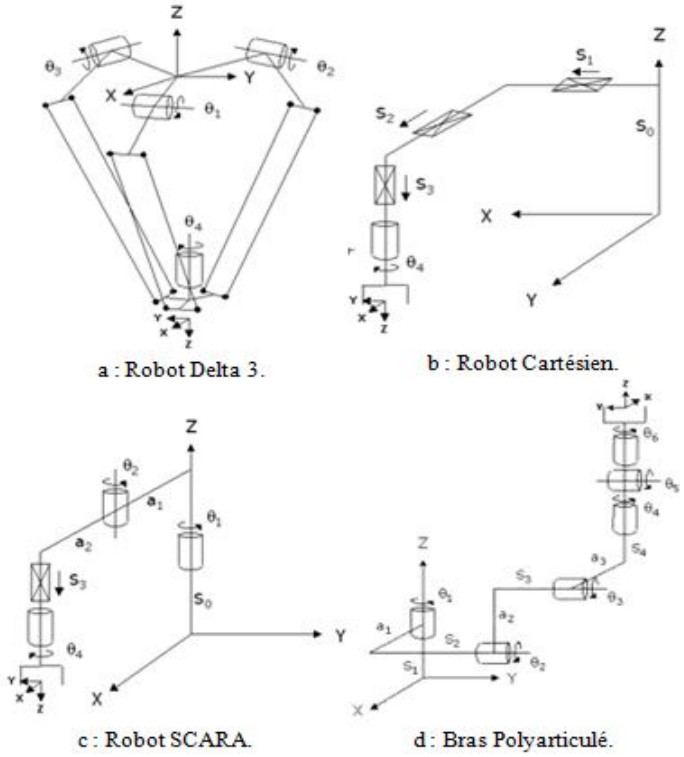

FIGURE 2.3 – Cinématiques des robots.

Les convoyeurs

Ils sont caratérisés par leur position, leur dimension, leur direction ainsi que leur sens et leur vitesse de fonctionnement, voir Figure 2.4. L'application pick & place avec des convoyeurs parallèles dans le même sens est la plus répandue.

L'environnement

Les caractéristiques de cette catégorie peuvent être séparées en trois groupes :

1. Les produits définis par leur orientation, leur type (parfum, couleur, forme etc.), leur poids, leur limite de vitesse et d'accélération etc.
2. Les boîtes caractérisées par leur nombre de places, leur orientation et/ou le type des produits à l'intérieur, leur poids maximum etc.
3. Les places (les produits sont directement déposés sur le convoyeur de sortie) spécifiées par leur orientation, leur type, leur poids etc.

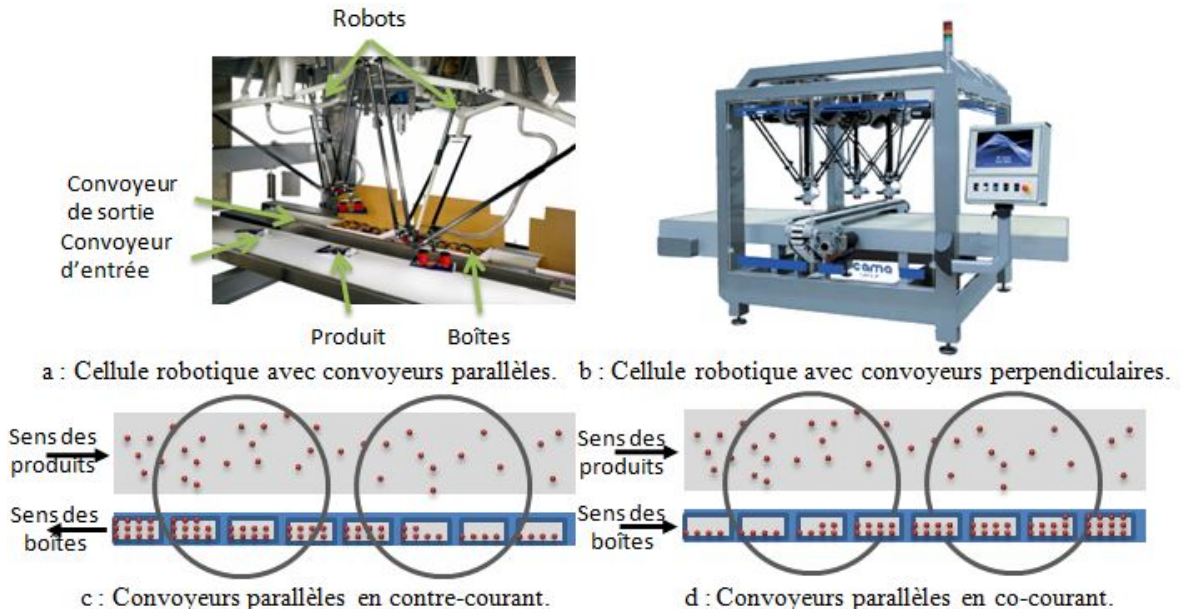

FIGURE 2.4 – Différentes configurations de convoyeurs.

Les conditions d'utilisation et de fonctionnement

Les spécificités fonctionnelles sont liées à quatre groupes :

1. La prise et la dépose : elles peuvent se faire en tracking (le robot se synchronise avec le convoyeur, permettant une prise ou une dépose de produits sans arrêt de celui-ci) ou fixe, simple ou multiple et avec des temps de préhension différents.
2. Le convoyage : il peut comporter un arrêt ou un changement de vitesse et un flux de produits positionnés aléatoirement ou en ligne.
3. La configuration : elle caractérise le nombre de robots et de convoyeurs.
4. La vision : elle a comme paramètres l'emplacement, le nombre de caméras ainsi que la taille de la zone observée.

Les événements et le mode de fonctionnement

Les événements influant sur les applications pick & place peuvent être une sur-cadence, une sous-cadence ou la panne d'un robot. Les différents modes de fonctionnement sont le démarrage, le régime permanent et l'arrêt de production.

Les stratégies utilisées.

Les stratégies sont des algorithmes implémentés dans le contrôleur de l'application et qui permettent de faire fonctionner le système et d'améliorer ses performances. Elles peuvent être appliquées à plusieurs niveaux :

1. La prise et la dépose.
2. La volonté d'avoir un panachage.

3. L'équilibre de la charge de travail entre les robots.
4. La gestion des convoyeurs.
5. L'utilisation ou non de buffers physiques (zones tampon où sont déposés les produits en cas de saturation des places).
6. La gestion de l'anticollision.

2.1.4 Les brevets d'application pick & place

Plusieurs brevets ont été déposés concernant les robots de pick & place. Le robot Delta motorisé a été breveté en 1985 par Reymond Clavel [Clavel, 1990], voir Figure 2.5. Ce type de robots est largement utilisé dans des applications pick & place hautes performances.

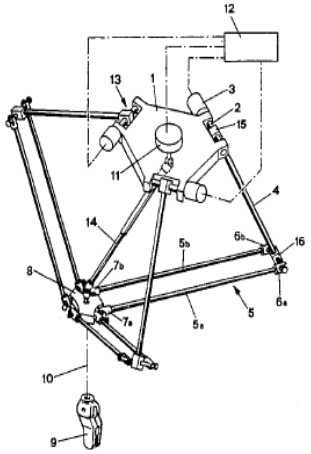

FIGURE 2.5 – Robot Delta 3 par [Clavel, 1990].

Le robot SCARA a été breveté en 1982 par Hiroshi Makino [Makino, 1982], voir Figure 2.6.

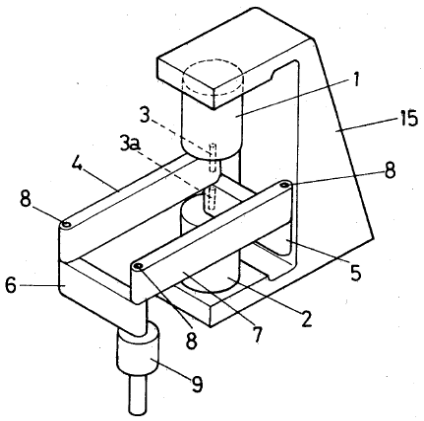

FIGURE 2.6 – Robot SCARA par [Makino, 1982].

Une des architectures connue, est l'utilisation des convoyeurs à contre-courant. C'est Schubert [Schubert, 2000] qui déposa le brevet de cette architecture en 2000. En plus de

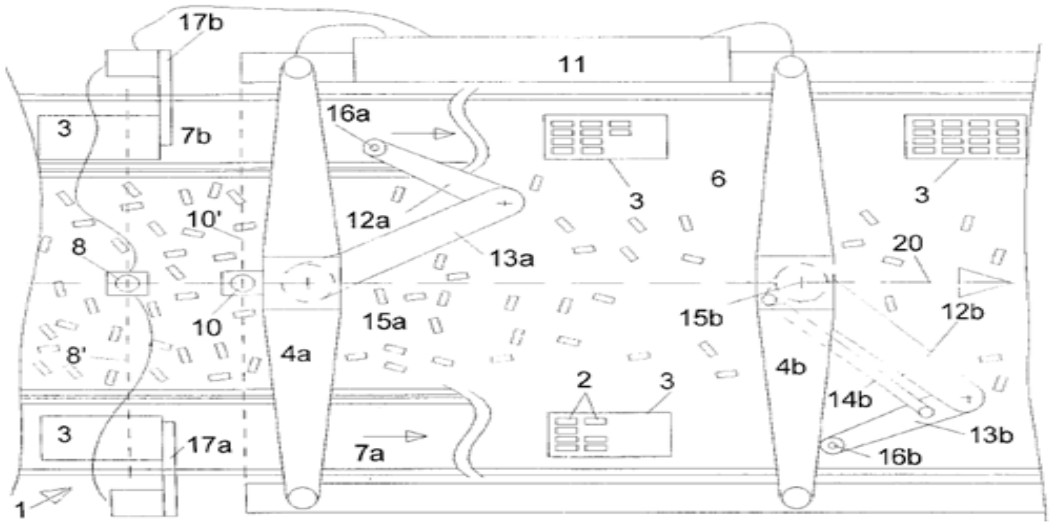

FIGURE 2.7 – Convoyeurs à contre-courant par [Schubert, 2000].

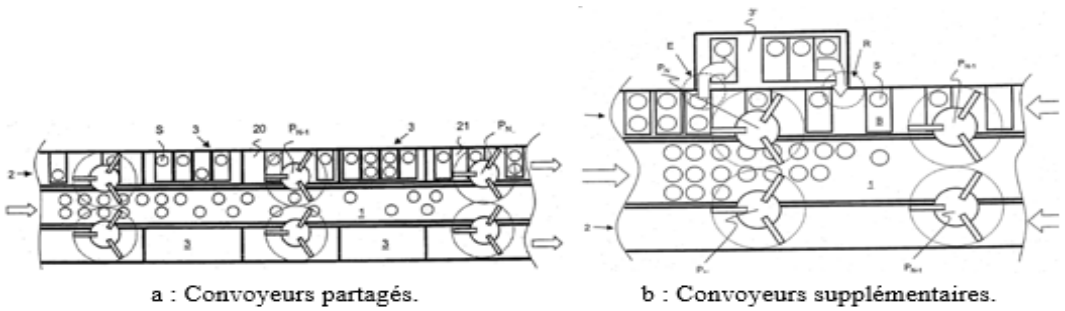

FIGURE 2.8 – Architectures de convoyeurs par [Huppi et al., 2003].

cela, il ajouta une architecture spécifique pour les robots voir Figure 2.7. L'intérêt de cette configuration de convoyeurs sera détaillé ultérieurement.

Huppi *et al.* [Huppi et al., 2003] ont déposé un brevet concernant différentes architectures mettant en place des éléments de stockage avec une logique FIFO (First In First Out) ou FILO (First In Last Out). Ces éléments sont créés en disposant des convoyeurs les uns à la suite des autres permettant d'avoir différentes vitesses de convoyage le long de la chaîne de production ou en rajoutant des convoyeurs supplémentaires situées à l'extérieur, voir Figures 2.8.a et 2.8.b.

Rutschmann [Rutschmann, 2010], quant à lui, a breveté une solution pour détecter les données de qualité des produits afin de différencier la cause d'une mise aux rebuts d'un produit : manque de qualité, sur-cadence, système mal dimensionné, etc. Cette solution utilise le principe de redondance d'informations sur la qualité et la position des produits à l'aide de plusieurs systèmes de vision situés avant chaque zone de travail des robots.

2.1.5 Conclusion sur les cellules robotisées

Durant cette section nous avons vu l'éventail des possibilités et la difficulté que peut avoir un constructeur de machines pour développer une application pick & place. Il devra, en collaboration avec le client, mettre en place un cahier des charges détaillé. En effet, au niveau hardware il devra préciser le nombre d'entrées/sorties ou de moteurs voulus et le type de bus de terrain souhaité, par exemple. Il devra aussi choisir les langages de programmation avec lesquelles il travaillera. Le plus important est de bien exprimer son besoin. En effet cela permettra de choisir le type de convoyeurs et de robots (Delta, SCARA, etc.) et de bien les configurer (vitesses, sens, multiprise, etc.). Une fois l'architecture et les caractéristiques de l'application choisies, il devra ensuite passer à la partie commande et donc programmer les algorithmes que doivent utiliser les robots afin de prendre et déposer les produits.

2.2 Algorithmes existants

Après avoir détaillé la partie dimensionnement des applications pick & place en explicitant différentes architectures et caractéristiques d'une application pick & place multi-robots, nous allons montrer la partie stratégie de commande. Dans cette section, seront présentés plusieurs algorithmes développés et utilisés dans la littérature. Ces algorithmes peuvent être séparés en deux catégories : les règles de planifications individuelles (algorithmes simples et applicables que pour un seul robot) et les heuristiques (algorithmes plus complexes permettant la collaboration entre plusieurs robots). Des algorithmes brevetés par l'industrie seront aussi présentés.

2.2.1 Règles de planification individuelles

Lorsqu'un seul robot est utilisé, un algorithme d'ordonnancement tel que la file d'attente est suffisant. Mattone *et al.* [Mattone et al., 1998] ont proposé des règles innovantes de planification en ligne basées sur les files d'attente, voir Figure 2.9. Leur contribution consiste à décomposer la zone de travail des robots en trois parties ayant une priorité de prise et de dépose. La première zone est dite "zone optimale" où les robots vont prendre et déposer les produits en priorité en utilisant une file d'attente. Cette zone se trouve au milieu de la zone de travail des robots. Lorsqu'il n'y a plus de produits ou de places dans cette zone, les robots vont prendre ou déposer dans la deuxième zone qui est en amont de la zone optimale. La dernière zone est situé en aval de la zone optimale. Les règles de gestion de files d'attente utilisées sont les suivantes :

- *FIFO* : First In First Out. Le robot prend le premier produit entré dans sa zone de travail.
- *LIFO* : Last In First Out. Le robot prend le dernier produit entré dans sa zone de travail.
- *SPT* : Shortest Processing Time. Le robot prend le produit le plus proche de son outil.

L'université Polytechnique de Turin [Comba et al., 2013] a comparé, au moyen d'un simulateur créé sous Matlab, deux structures pick & place avec deux convoyeurs dans le

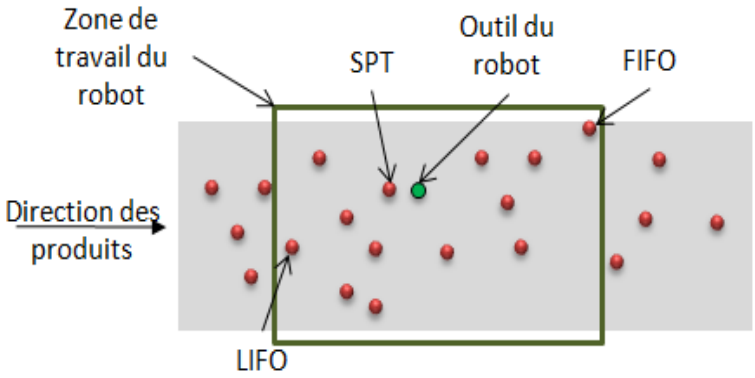

FIGURE 2.9 – Identification des produits suivant les règles de planification individuelles.

même sens puis dans le sens opposé suivant différentes hypothèses : présence ou non d'une logique de prise et de dépose et utilisation ou non de buffers physiques.

Les résultats obtenus montrent qu'un système avec des convoyeurs en co-courant possède un débit maximum légèrement plus élevé qu'un système avec des convoyeurs en contre-courant. En effet, il y a moins de distance à parcourir pour les robots lorsque les convoyeurs fonctionnent dans le même sens. De plus il y a un effet boule de neige avec des convoyeurs en contre-courant : les distances se rallongent de plus en plus jusqu'à être en dehors des limites de la zone de travail des robots. Alors que pour un système en co-courant les distances resteront globalement les mêmes, voir Figure 2.10. Cependant en cas de perturbation comme une sur-cadence, le contre-courant permet une perte de produits plus faible. Lorsqu'il y a une sous-cadence ou une sur-cadence temporaire, un système avec des convoyeurs en co-courant a tendance à perdre davantage de produits.

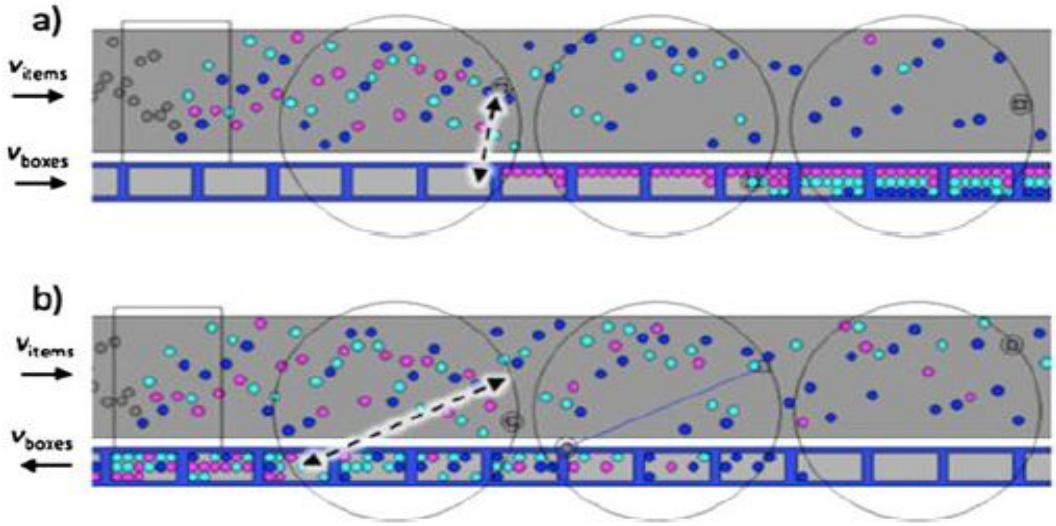

FIGURE 2.10 – Comparaison des structures co-courant/contre-courant [Comba et al., 2013].

Suivant les hypothèses, la répartition de la charge de travail entre les robots est diffé-

rente. Sans logique de prise et de dépose, la charge de travail est très déséquilibrée pour un système en co-courant. En effet le premier robot est saturé tandis que le dernier robot est inactif. Pour des convoyeurs en contre-courant, il y a un meilleur équilibre mais la charge du premier robot varie fortement. Avec une logique de prise, il y a une nette amélioration de la répartition de la charge pour les deux configurations. Lors de l'utilisation de buffer, seul le dernier buffer est utilisé.

Avec l'ajout d'une logique de dépose, il y a une amélioration de l'équilibrage des zones tampons, mais la charge de travail des robots devient anormalement élevée. En effet, les robots effectuent des allers-retours dans les buffers (le robot met un produit dans le buffer puis le récupère presque tout de suite pour le mettre dans une boîte). Avec une restriction sur l'utilisation des buffers (utilisation de la zone tampon quand il n'y a plus de place dans une boîte et quand le produit est dans la seconde moitié de la zone de travail du robot. De même, le produit est repris s'il y a une place vide dans une boîte présente dans la seconde moitié de la zone de travail.), cette charge de travail retrouve une valeur normale.

Au département Génie Industriel et Management de l'Université de Ben-Gourion du Néguev en Israël, Edan *et al.*[Edan et al., 2004] ont, quant à eux, travaillé sur des robots qui assemblent deux pièces pour comparer différents algorithmes afin de maximiser des assemblages et minimiser le déséquilibre de la charge de travail :

- Un algorithme "Prends-ce-que-tu-peux". Les robots assemblent tous les produits qu'ils peuvent.
- Un algorithme "Moyenne" égalisant le nombre d'assemblages entre les robots. Si le premier robot assemble plus de produits, il attendra que le deuxième robot dépasse son nombre d'assemblage, pour continuer à fonctionner.
- Un algorithme "Occupation" égalisant le temps d'attente entre les robots. Si le temps d'attente d'un robot pour assembler des produits est supérieur à l'autre robot, ce dernier temporisera ses assemblages jusqu'à avoir un temps d'attente identique entre les robots.
- Un algorithme à base de logique floue.

Il est à noter qu'ils comparent différents emplacements des robots le long du convoyeur : tous les robots d'un même coté, alternance droite gauche etc.

Les résultats obtenus sont regroupés dans le tableau 2.1 et montrent que l'algorithme "Prends-ce-que-tu-peux" maximise les assemblages mais maximise aussi la différence du nombre d'assemblages entre les robots. Deuxièmement, l'algorithme de moyenne répartit bien la charge de travail entre les robots mais le nombre d'assemblages est réduit.

TABLE 2.1 – Résultats obtenus par l'université de Ben-Gourion [Edan et al., 2004].

Algorithmes	"Prends-ce-que-tu-peux"	Moyenne	Occupation	Logique floue
Nombre d'assemblage	++	−	-	+
Équilibre de la charge de travail	−	++	+	-

De plus l'emplacement des robots le long du convoyeur influence peu sur le nombre d'assemblages et sur l'équilibre de la charge de travail entre les robots.

Durant cette section trois règles de planification individuelles basées sur les files d'attente ont été présentées (*FIFO*, *LIFO* et *SPT*). Ces règles étant des algorithmes de base, elles seront exploitées durant les études comparatives réalisées dans les chapitres 5 et 6 de ce mémoire de thèse. De plus un début d'étude comparative entre plusieurs configurations a été effectué (convoyeur dans le même sens ou en sens contraire). L'étude fournissant des résultats prometteurs, ils seront repris lors de nos études comparatives. Finalement, une comparaison entre différents algorithmes a été faite. Le principe de l'algorithme de moyenne (répartition du nombre d'assemblages pour un équilibre de la charge de travail) est très important et inspirera donc de futurs algorithmes développés au cours de cette thèse. Les travaux effectués dans cette section ont pour base des règles individuelles et des algorithmes simples. Nous allons voir à présent des algorithmes plus complexes permettant de rajouter une couche haut-niveau de programmation afin de permettre la collaboration entre les robots dans le but d'améliorer les performances du système pick & place.

2.2.2 Heuristiques et métaheuristiques

Une heuristique est une méthode de calcul qui fournit rapidement une solution réalisable, pas nécessairement optimale, pour un problème d'optimisation. Une métaheuristique est un algorithme d'optimisation visant à résoudre des problèmes d'optimisation difficile pour lesquels on ne connaît pas de méthode classique plus efficace. Les métaheuristiques sont généralement des algorithmes stochastiques itératifs, qui progressent vers un optimum global, par une exploration de la fonction coût sur l'ensemble de l'espace des variables de décision. Ces méthodes utilisent parfois un haut niveau d'abstraction, leur permettant d'être adaptées à une large gamme de problèmes différents.

Ces méthodes peuvent être utilisées pour une application pick & place. Elles ont pour objectif de gérer les robots en tenant compte des produits, des cartons et des convoyeurs. Des travaux de recherche liés à ces méthodes, utilisées dans des applications robotiques, sont nombreux. Voici une présentation non-exhaustive de quelques-unes de ces heuristiques et métaheuristiques ainsi que de leur utilisation dans un système pick & place. Cet état de l'art est focalisé seulement sur certains algorithmes car ce travail de thèse n'a pas pour principal objectif le développement d'algorithmes d'optimisation.

Présentation d'heuristiques et de métaheuristiques

L'optimisation par colonie de fourmis (ACO) a été initialement proposée par Marco Dorigo *et al.* [Dorigo et al., 1991] [Dorigo, 1992] dans les années 1990. Les algorithmes de colonies de fourmis sont des algorithmes inspirés du comportement des fourmis, recherchant un chemin entre leur colonie et une source de nourriture, qui utilisent le principe des phéromones. Il s'appuie sur quatre étapes : le codage de la solution, la construction des chemins des fourmis, la mise à jour locale des phéromones et la mise à jour globale des phéromones.

Le principe de l'optimisation par colonie de fourmis est illustré par la figure 2.11 :

1. La première fourmi trouve la source de nourriture (F), via un chemin quelconque (a), puis revient au nid (N) en laissant derrière elle une piste de phéromone (b).
2. Les fourmis empruntent indifféremment les quatre chemins possibles, mais le renforcement de la piste rend plus attractif le chemin le plus court.

3. Les fourmis empruntent le chemin le plus court, les portions longues des autres chemins perdent leur piste de phéromones.

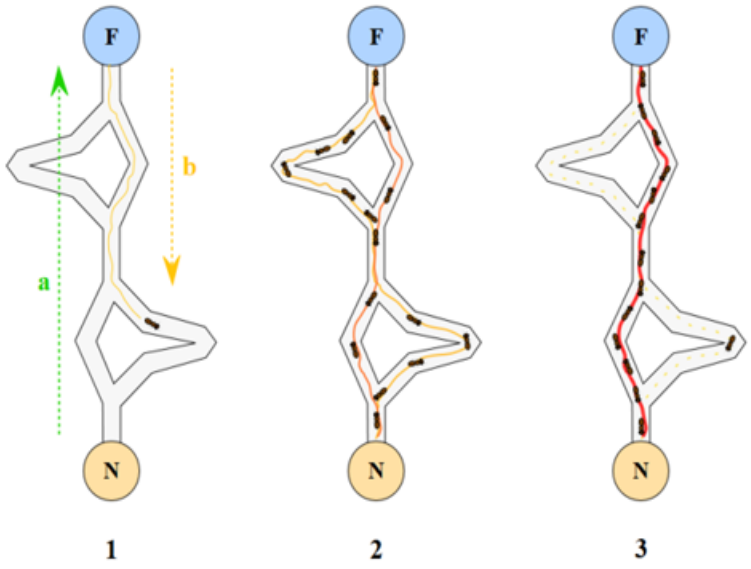

FIGURE 2.11 – Principe de l'optimisation par colonie de fourmis.

Les algorithmes génétiques (GA) ont été initiés dans les années 1970 par John Holland [Holland, 1975]. Ils appartiennent à la famille des algorithmes évolutionnistes. Leur but est d'obtenir une solution approchée à un problème d'optimisation, lorsqu'il n'existe pas de méthode exacte (ou que la solution est inconnue) pour le résoudre en un temps raisonnable. Les algorithmes génétiques utilisent la notion de sélection naturelle et l'appliquent à une population de solutions potentielles au problème donné. La solution est approchée par "bonds" successifs. Ce sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et des mécanismes d'évolution de la nature : croisement, mutation, sélection, voir Figures 2.13 et 2.12. Un point important est le temps de calcul de la fonction d'évaluation. En effet, celui-ci doit être raisonnablement court car la fonction est évaluée de nombreuses fois.

Eberhart et Kennedy [Kennedy and Eberhart, 1995] ont inventé l'optimisation par essaim de particules (PSO). Russel Eberhart est un ingénieur en électricité et James Kennedy, un socio-psychologue. Cet algorithme est inspiré du monde des vivants comme les groupes d'oiseaux ou les bancs de poissons et repose sur la collaboration des individus entre eux. En effet, on peut observer chez ces animaux des dynamiques de déplacement relativement complexes, alors qu'individuellement chaque individu a une "intelligence" limitée, et ne dispose que d'une connaissance locale de sa situation dans l'essaim. L'information locale et la mémoire de chaque individu sont utilisées pour décider de son déplacement. Des règles simples, telles que "rester proche des autres individus", "aller dans une même direction" ou "aller à la même vitesse", suffisent pour maintenir la cohésion de l'essaim, et permettent la mise en œuvre de comportements collectifs complexes et adaptatifs. Les particules bougent à chaque itération en fonction de plusieurs éléments : leur vitesse, leur

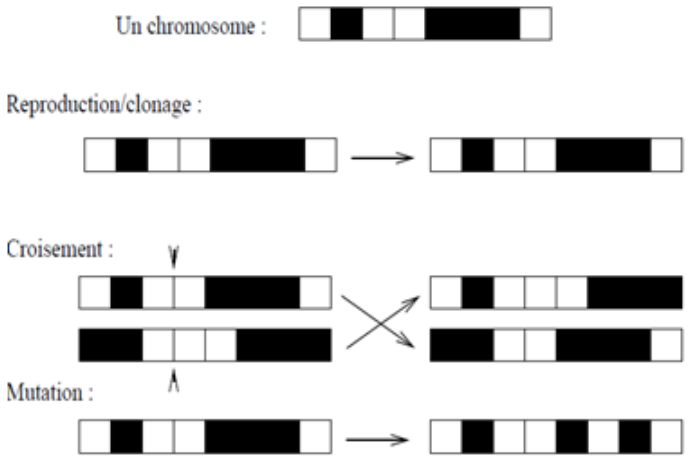

FIGURE 2.12 – Exemple de mécanisme d'évolution de la nature.

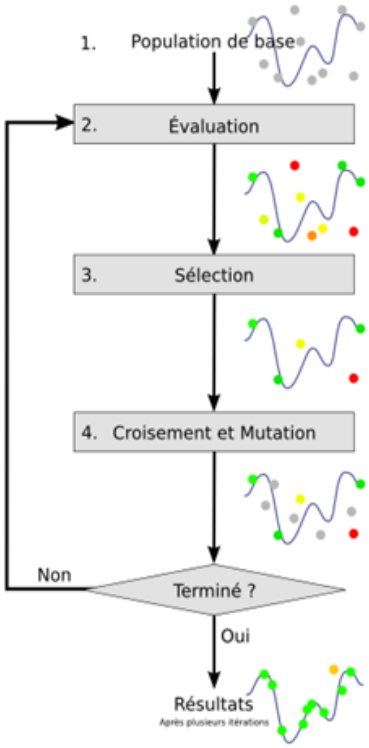

FIGURE 2.13 – Principe de l'algorithme génétique .

meilleure solution individuelle et la meilleure solution dans leur voisinage, voir Figure 2.14.

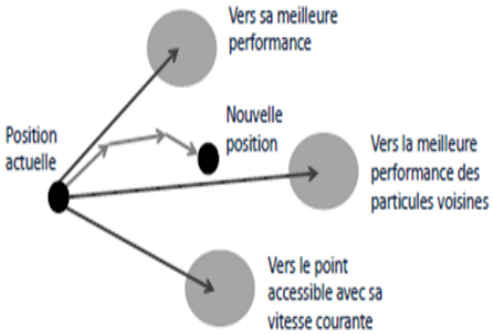

FIGURE 2.14 – Principe de l'optimisation par essaim de particules.

L'algorithme d'évolution différentielle (DE) est une méthode métaheuristique stochastique d'optimisation. Il est inspiré par les algorithmes génétiques et les stratégies évolutionnistes combinées avec une technique géométrique de recherche. Les algorithmes génétiques changent la structure des individus en utilisant la mutation et le croisement, alors que les stratégies évolutionnaires réalisent l'auto-adaptation par une manipulation géométrique des individus. Ces idées ont été mises en œuvre grâce à une opération simple, mais puissante, de mutation de vecteurs, proposée en 1995 par K. Price et R. Storn [Storn and Price, 1997].

La logique floue est une extension de la logique booléenne créée par Lotfi Zadeh en 1965 [Zadeh, 1964] en se basant sur sa théorie mathématique des ensembles flous, qui est une généralisation de la théorie des ensembles classiques. En introduisant la notion de degré dans la vérification d'une condition, elle permet à une condition d'être dans un autre état que vrai ou faux. La logique floue confère ainsi une flexibilité très appréciable aux raisonnements qui l'utilisent, ce qui rend possible la prise en compte des imprécisions et des incertitudes. Un des intérêts de la logique floue pour formaliser le raisonnement humain est que les règles sont énoncées en langage naturel. Voici par exemple quelques règles de conduite qu'un conducteur suit :

TABLE 2.2 – Exemple de règles de la logique floue.

Si le feu est	si ma vitesse est	et si le feu est	alors
rouge	élevée	proche	je freine fort.
rouge	faible	loin	je maintiens ma vitesse.
orange	moyenne	loin	je freine doucement.
vert	faible	proche	j'accélère.

La théorie des jeux est un ensemble d'outils pour analyser les situations dans lesquelles l'action optimale pour un agent dépend des anticipations qu'il forme sur la décision d'un autre agent. L'objectif de la théorie des jeux est de modéliser ces situations, de déterminer

une stratégie optimale pour chacun des agents, de prédire l'équilibre du jeu et de trouver comment aboutir à une situation optimale. Les fondements de la théorie moderne des jeux sont décrits pour la première fois en 1928 par John von Neumann [von Neumann, 1928]. Les idées de la théorie des jeux sont ensuite développées par Oskar Morgenstern et John von Neumann en 1944 [von Neumann and Morgenstern, 1944].

Exemple d'utilisation des heuristiques et des métaheuristiques

Ces différents algorithmes peuvent être utilisés pour optimiser différents types de paramètres dans des applications pick & place. La recherche de la meilleure combinaison de règles de planification individuelles est une voie d'exploration. Le temps de calcul pour trouver cette combinaison est une donnée importante que l'on cherche souvent à minimiser. Huang *et al.* [Huang et al., 2012] ont utilisé la *Greedy Randomized Adaptive Search Procedure* pour rechercher la combinaison optimale des règles de planification. Cet algorithme permet de réduire le temps de calcul de 56% par rapport à une méthode de recherche exhaustive. Ceci, car il n'évalue pas toutes les combinaisons. De plus il montre que lorsqu'un système pick & place multi-robots ne fonctionne pas à son maximum, le dernier robot doit utiliser la règle individuelle de planification *FIFO*. Tandis que lorsque le système est sous-dimensionné, le dernier robot doit utiliser la règle *SPT*.

Fujimoto *et al.* [Fujimoto et al., 1995] ont, quant à eux, utilisé un algorithme génétique pour un atelier flexible. L'utilisation de cet algorithme permet d'obtenir des résultats plus rapidement qu'en simulant toutes les possibilités.

Daoud *et al.* [Daoud et al., 2014b] [Daoud et al., 2014a] ont comparé trois méthodes : l'optimisation par colonie de fourmis, les algorithmes génétiques et l'optimisation par essaim de particules. L'objectif est identique : trouver la meilleure combinaison de règles de planification pour maximiser le débit de prise d'un système robotique pick & place en prenant en compte le temps d'exécution. Ils ont montré que l'optimisation par colonie de fourmis permet d'obtenir les meilleurs résultats avec le minimum de temps de calcul.

Zhu, *et al.* [Zhu and Chen, 2009] ont étudié un algorithme d'évolution différentielle pour un problème de machines pick & place utilisant un préhenseur multi-têtes pour une application de placements de composants sur un circuit imprimé en le comparant à un algorithme génétique. L'algorithme différentiel est plus efficace et obtient des meilleurs résultats que l'algorithme génétique. Il permet de déposer les composants plus rapidement sur le circuit imprimé tout en réduisant le temps de calcul nécessaire.

On retrouve ce contexte de placement de produits sur un circuit imprimé dans plusieurs travaux. Bonert *et al.* [Bonert et al., 2010] ont recherché une solution au problème du voyageur de commerce avec un algorithme génétique. Ils la mettent en œuvre pour la collaboration de plusieurs robots afin de réduire la durée de leurs mouvements.

Najera *et al.* [Najera and Brizuela, 2005], Chang *et al.* [Chang et al., 2012] et Peng *et al.* [Peng and Zeng, 2013] ont proposé un algorithme génétique afin de rechercher les séquences optimales de prise et de dépose de composants pour l'assemblage de circuits imprimés. Les premiers ont amélioré le temps d'assemblage d'un circuit imprimé par rapport aux méthodes utilisées dans l'industrie de 13.93% en moyenne. Pour cela ils ont rendu leur algorithme "élitiste". Lors d'un croisement ils ne gardent que le meilleur chromosome de la paire. Cependant, ils utilisent deux fois plus de parents que pour un croisement standard.

Les seconds utilisent un algorithme génétique avec un paramètre appelé *ESMA* (*External Self-evolving Multiple Archives*) qui permet d'améliorer la convergence de l'algorithme d'environ 15%. De plus, cet algorithme permet d'éviter les optimums locaux. Tandis que les troisièmes ont utilisé en plus de l'algorithme génétique une méthode ad-hoc appelée *DSWS* (*Distance Score with Weights Selection*). Ils ont obtenu des meilleurs temps de déplacements pour placer les composants qu'avec un simple algorithme génétique ou avec un algorithme d'optimisation par essaim de particules.

La charge de travail est aussi un point crucial dans des applications pick & place multi-robots. Rubinovitz *et al.* [Rubinovitz et al., 1998] ont utilisé une heuristique pour équilibrer la charge de travail entre les robots dans une chaîne de production en allouant une quantité de travail égale à tous les robots de la ligne. Elle s'appuie sur un algorithme par séparation et évaluation (*Branch and Bound*). Cet algorithme a été utilisé pour résoudre des problèmes NP complexes tels que l'équilibrage d'une ligne d'assemblage [Johnson, 1988] [Johnson, 1956].

Mendelson, *et al.* [Mendelson et al., 2002] ont mis au point un système multi-robots de palettisation décentralisée utilisant la logique floue en le comparant à un algorithme simple "Prends-ce-que-tu-peux". L'algorithme repose sur la logique floue et prend en compte le nombre de pièces assemblées, la position des robots et le taux d'utilisation des robots. Les résultats obtenus sont similaires à ceux montrés dans la section 2.2.1. L'algorithme simple possède de meilleures performances, cependant la logique floue permet une meilleure répartition du taux d'utilisation entre les robots.

D'autres voies de recherche ont été explorées. Huang *et al.* [Huang et al., 2013] ont utilisé l'algorithme d'optimisation par essaim de particules pour réaliser rapidement la conception d'un système multi-robots pick & place en prenant en compte le bras des robots et la position de leur base.

Bozma, *et al.* [Bozma and Kalalioglu, 2012] ont proposé une approche basée sur la théorie des jeux non coopératifs pour décider des actions des robots dans une application pick & place multi-robots. Chaque robot prend en compte le convoyeur et les actions faites par les robots voisins afin de décider de ses propres actions.

Dans le même contexte de coordination entre une équipe robots mobiles afin d'éviter toutes collisions, Harmatia *et al.* [Harmatia and Skrzypczykb, 2009] ont développé un nouveau concept de théorie des jeux utilisant le point d'équilibre semi-coopératif de Stackelberg. De plus la logique floue est utilisée pour renforcer la robustesse de la méthode d'optimisation.

Finalement, Alaya *et al.* [Alaya et al., 2006] ont résolu le problème du sac à dos multidimensionnel avec une méthode métaheuristique utilisant des colonies de fourmis. Ce problème peut être transcrit à une application pick & place lorsqu'il a besoin de prendre en compte le poids de chaque élément afin de remplir au plus juste les cartons.

Habituellement, les techniques d'optimisation ci-dessus sont très gourmandes en temps de calcul et ne sont pas appropriées pour des applications en temps réel où le cycle de calcul nécessaire est de l'ordre de la milliseconde. Or précisément, dans les applications pick & place hautes performances, le temps de décision doit être le plus faible possible afin de ne pas laisser passer de produits. De plus, certains algorithmes utilisent comme critère, le nombre de produits pris ou le nombre de boîtes remplies. Or ces critères ne peuvent

être obtenus qu'après une simulation ou une expérimentation et non en temps réel durant le fonctionnement d'une chaîne de production. Une autre alternative est d'utiliser ce type d'algorithmes hors ligne dans la simulation afin de trouver la meilleure combinaison de règles de planification individuelles parmi les combinaisons testées, puis d'appliquer le résultat dans la pratique.

Cependant, ce genre de simulation demande beaucoup d'itérations pour chaque configuration d'application. Par exemple, on souhaite avoir la meilleure combinaison de règles de planification individuelles pour chaque robot dans un système avec quatre robots. Il y a trois règles possibles : *FIFO*, *LIFO* et *SPT*. On peut affecter une règle différente pour la prise et pour la dépose. Il y a donc au total $3 \wedge 2 \wedge 4 = 6561$ possibilités. Le critère principal est le nombre de produits perdus. Pour pouvoir évaluer ce critère, il faut effectuer une simulation durant minimum 10 minutes pour avoir suffisamment de données exploitables et ceci pour chaque possibilité. À partir de 10 minutes de simulation, l'écart type entre plusieurs simulations ayant le même cahier des charges, est suffisamment faible pour pouvoir exploiter les données. Les heuristiques permettent de réduire le nombre de possibilités, mais le temps de simulation reste cependant élevé. Finalement, dans tous ces travaux, pour valider les résultats, seule la simulation est utilisée, la traduction des stratégies de commandes de la simulation à l'expérimentation n'est pas abordée. C'est pour ces différentes raisons que les heuristiques présentées ne seront pas retenues pour cette thèse.

Nous avons vu depuis le début de cette section des travaux effectués d'un point de vue académique. Il existe beaucoup d'autres travaux académiques qui n'ont pas été abordés au cours de cette thèse, faute de temps. En parallèle, il existe des travaux visant à améliorer les performances des applications pick & place à l'aide d'algorithmes, effectués par des industriels et faisant l'objet de brevets. Il convient donc de se pencher sur ces travaux afin de confondre les deux points de vue : académique et industriel.

2.2.3 Brevets liés aux algorithmes de pick & place

Comme indiqué en Introduction, l'amélioration des performances des machines est un enjeu important pour les industriels. Plusieurs d'entre eux ont déposé des brevets afin de répondre à cet objectif en proposant des algorithmes et des méthodes pour améliorer différents critères de performances d'applications pick & place dans l'emballage.

Erlandsson-Warvelin *et al.* [Erlandsson-Warvelin and Knobel, 2012] donnent des indications sur une méthode de tri. Elle prend en compte le sens de tri voulu (perpendiculaire ou parallèle au convoyeur par exemple) et le partage du convoyeur en zones : début de tri, limite de tri et fin de tri, voir Figure 2.15. Cette méthode utilise le même principe que Mattone *et al.* [Mattone et al., 1998].

Izumi *et al.* [Izumi et al., 2013] ont déposé un brevet sur le partage de convoyeurs afin de répartir la charge de travail entre les robots. Ils ont proposé cinq principes :

1. Le convoyeur est partagé en plusieurs zones, voir Figure 2.16. Par exemple pour un système à deux robots, le premier robot prend les produits sur la partie haute du convoyeur tandis que le deuxième prend les produits sur la partie basse. La taille des zones peut varier si la vitesse des robots est différente.
2. Le convoyeur est partagé suivant le principe 1. Cependant, il prend en compte le

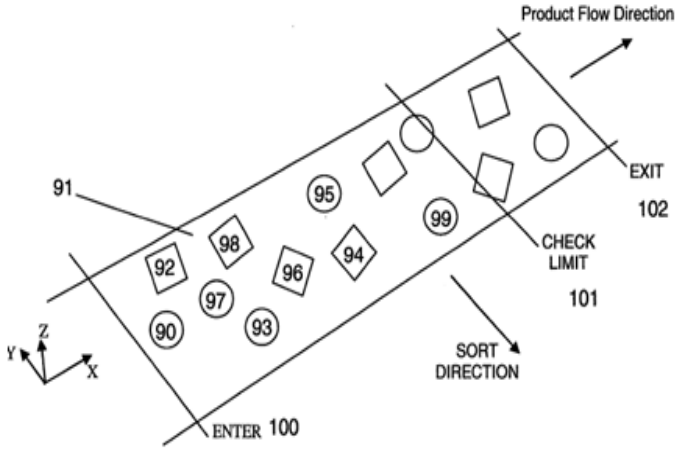

FIGURE 2.15 – Méthode de tri par [Erlandsson-Warvelin and Knobel, 2012].

nombre de produits pris par chaque robot afin d'adapter la taille des zones. Si le premier robot prend plus de produits que le deuxième, la taille de sa zone sera réduite dynamiquement.

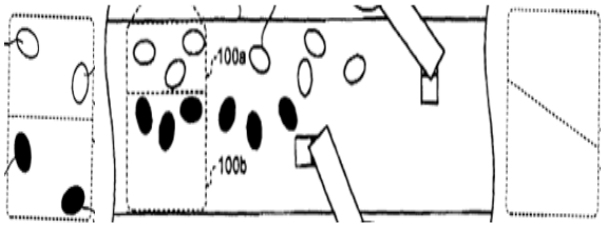

FIGURE 2.16 – Exemples de partage du convoyeur pour deux robots par [Izumi et al., 2013].

3. Les produits sont affectés aux robots suivant leur orientation, voir Figure 2.17. Le premier robot va prendre les produits entre -90 ° et 0 ° tandis que le deuxième va prendre les produits entre 0 ° et 90 ° , par exemple.
4. Les produits sont affectés suivant le principe 3. Cependant, la distribution prend en compte le nombre de produits pris par chaque robot afin d'adapter dynamiquement la taille des zones. Si le premier robot prend plus de produits que le deuxième, la taille de sa zone sera réduite (le premier robot ne prend plus que les produits entre -90 ° et -30 ° et le deuxième va prendre les produits entre -30 ° et 90 °).
5. Le convoyeur est partagé en plusieurs zones mais en utilisant des caméras pour partitionner le convoyeur, voir Figure 2.18. Avec cette configuration, les zones sont attribuées les unes à la suite des autres.

L'intérêt de ces travaux est le partage de la charge de travail entre les robots qui est un procédé important dans une application pick & place. Ces partages de convoyeurs seront utilisés durant l'étude comparative du chapitre 5.

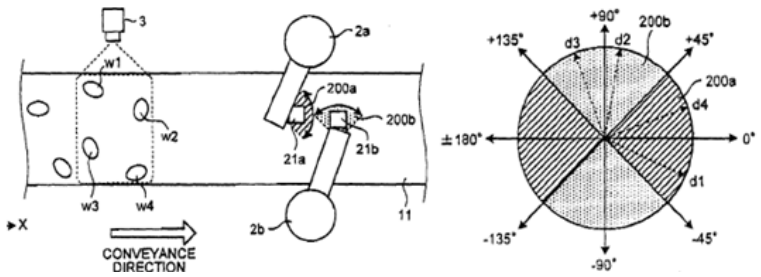

FIGURE 2.17 – Exemple d'attribution des produits aux robots suivant l'orientation par [Izumi et al., 2013].

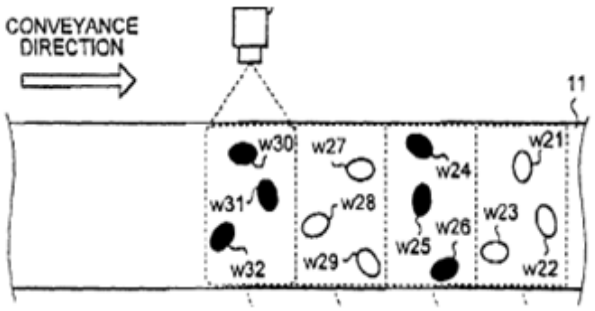

FIGURE 2.18 – Exemple de partage de convoyeur avec caméras par [Izumi et al., 2013].

Herzog [Herzog, 2003] a déposé un brevet sur une méthode de remplissage des conteneurs utilisant un système à évènements discrets. L'objectif de ce brevet est d'améliorer la répartition de la charge de travail entre les robots afin d'augmenter leur durée de vie. Il a utilisé la programmation quadratique pour résoudre le problème. Cette méthode est composée de cinq étapes :

1. Acquisition des données nécessaires (charges des robots, nombres de pièces, etc.).
2. Génération de plusieurs solutions à l'aide d'un système d'équations discrètes.
3. Suppression des solutions ne remplissant pas les conditions.
4. Evaluation et classement des solutions selon les conditions.
5. Application de la meilleure solution.

Ehrat et Renner [Ehrat and Renner, 2010] ont breveté une méthode pour ajuster les performances des robots ainsi que la vitesse des convoyeurs pour avoir une augmentation linéaire ou régressive du remplissage des conteneurs assurée, indépendamment du nombre de produits approvisionnés. Ce brevet repose sur une application avec des convoyeurs en contre-courant. Le principe de ce brevet est la mesure de la concentration de produits tout au long du convoyeur. Cette concentration permet d'ajuster le niveau de remplissage des robots. De plus, le convoyeur est contrôlé par un des robots de la chaîne de production. Suivant la concentration de produits au niveau de ce robot, le convoyeur va être accéléré ou ralenti pour permettre de maintenir le taux de remplissage souhaité. Le contrôle du convoyeur peut être transféré à un autre robot si la concentration en produits, au niveau du robot contrôleur, est nulle.

Wappling *et al.* [Wappling and Murphy, 2012] ont déposé un brevet sur une méthode de planification pour des applications pick & place prenant en compte les temps de prise et de dépose pour chaque produit afin de minimiser le temps total de déplacement. Les positions des produits et des places sont tout d'abord déterminées. Ensuite, le temps de prise et de dépose pour chaque produit et place est ensuite calculé prenant en compte la vitesse des robots et des convoyeurs (pour anticiper leur déplacement). Pour finir, les robots vont prendre et déposer les produits dans un ordre déterminé en respectant les temps de prise et de dépose voulus.

2.2.4 Conclusion sur les algorithmes existants

Nous avons vu durant cette section qu'il existe plusieurs algorithmes développés dans la littérature et qui ont été séparés en deux groupes. Le premier groupe contient les règles de planification individuelles. Ces algorithmes ne sont utilisés que pour un seul robot et reposent sur le principe des files d'attente. On peut retenir trois règles : *FIFO*, *LIFO* et *SPT* qui seront exploitées durant la phase d'étude comparative. Le deuxième groupe contient les heuristiques. Ce sont des algorithmes complexes qui permettent de prendre en compte la collaboration entre les robots. Ces algorithmes demandent cependant un temps de calcul ou d'exécution trop important pour être utilisés dans une application pick & place haute performance car il est très difficile de résoudre ces problèmes d'optimisation en temps réel. De plus, pour cet état de l'art, ces algorithmes n'ont été testés et validés qu'en simulation. Il n'y a pas d'essai sur des démonstrateurs et la transition entre simulation et expérimentation n'est pas abordée, ce qui est dommageable. Finalement, quelques brevets issus des travaux effectués par des industriels ont été présentés. Ces travaux prennent en compte l'expérience du terrain et garantissent la validation par l'expérience. Certains de ces travaux seront donc utilisés durant la phase d'étude comparative.

2.3 Conclusion sur l'état de l'art

Dans ce chapitre, nous avons présenté un état de l'art global lié aux applications pick & place. La première section était concentrée sur la partie dimensionnement des applications tandis que la seconde section était orientée sur la partie stratégie de commande. Nous avons vu que l'utilisateur d'application pick & place doit fournir aux constructeurs un cahier des charges détaillé et expliquer correctement son besoin. En effet, d'une chaîne de production à une autre, le système pick & place peut être totalement différent. Toutes les parties du système peuvent être configurées à différents niveaux (nombre de robots, vitesse, prise multiple, convoyeurs, produits etc.). Nous avons aussi vu que, pour avoir des performances souhaitées ou les améliorer, l'utilisateur doit mettre en place des algorithmes. Ces algorithmes sont à deux niveaux : les règles de planification individuelles pour chaque robot et les stratégies de collaboration entre les robots. Cependant, il doit prendre en compte un critère essentiel pour les applications temps réel qui est le temps de calcul. Celui-ci doit être le plus petit possible pour garantir un fonctionnement en temps réel.

Nous nous sommes attachés jusqu'à présent aux éléments distinctifs d'une application pick & place. Le prochain chapitre présentera une méthodologie conduisant à la configu-

ration et à la mise en place d'une application pick & place en prenant en compte tous les éléments vus durant ce chapitre (les caractéristiques d'une application et les algorithmes de pick & place).

Chapitre 3

Méthodologie pour la mise en place d'une application pick & place

Sommaire

3.1 Introduction

Lorsqu'un utilisateur ou un constructeur de machines souhaite mettre en place une cellule robotisée pour réaliser du pick & place, il devra passer par plusieurs étapes : le développement mécanique du robot, la programmation du robot et les tests pour vérifier le fonctionnement de l'application. En industrie, l'implémentation d'un système pick & place sur une chaîne de production repose essentiellement sur l'expérience du terrain de la ou les personnes en charge du projet et de son équipe. Ils utilisent peu d'outils pour aider au dimensionnement et à l'implémentation. C'est en partant de ce constat que nous

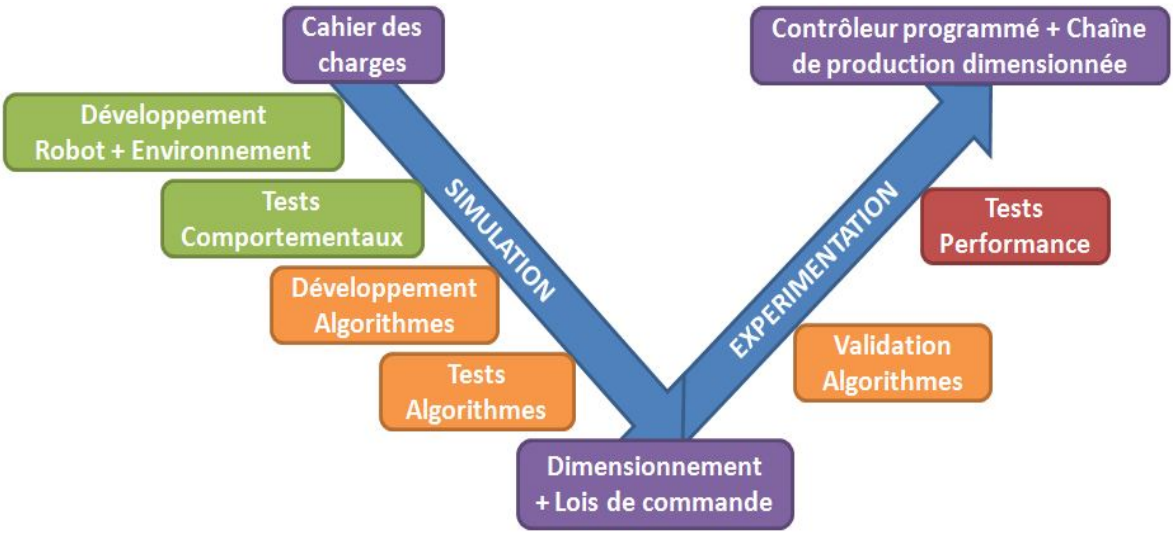

FIGURE 3.1 – Cycle de développement d'une application pick & place.

avons été conduits à développer une méthodologie pour la mise en place d'une application pick & place qui peut être résumée au moyen de la Figure 3.1.

Le développement d'une application pick & place peut se décomposer en deux parties : la simulation et l'expérimentation. La simulation permet, à l'aide du cahier des charges fourni par l'utilisateur (qui doit être le plus détaillé et précis possible, voir section 2.1), de réaliser le dimensionnement de l'application (boîtes vertes) et de développer les lois de commande permettant de contrôler le système (boîtes orange côté simulation). Le dimensionnement de l'application consiste dans un premier temps à développer la cinématique et la dynamique du robot à utiliser puis, dans un second temps, à développer son environnement (produits, cartons etc.). Une fois les robots fonctionnant et interagissant correctement avec leur environnement, la phase réalisation des lois de commande peut être effectuée. Elle a pour objectif de développer des algorithmes permettant la commande des robots afin de prendre les produits et les déposer dans les cartons. Une fois le dimensionnement et le développement des lois de commande réalisés et leur validation effectuée en simulation, la phase d'expérimentation débute. Cette partie consiste à programmer le contrôleur commandant l'application afin de tester sur le procédé physique les différentes lois de commande développées (boîte orange côté expérimentation) et afin d'obtenir les performances finales du système pick & place mis en place (boîte rouge).

À notre connaissance, dans le monde industriel et académique, il n'existe pas d'outil logiciel qui prenne en compte les quatre aspects suivants : une simulation du comportement des robots, une simulation de l'environnement de travail (flux de produits, flux de boîtes), le travail collaboratif de plusieurs robots et la possibilité d'aller de la simulation à l'expérimentation de manière automatisée. Il existe des outils performants qui répondent à un seul besoin mais aucun logiciel ne répond entièrement aux quatre fonctionnalités énoncées ci-dessus. Il est nécessaire qu'à chaque étape une prise en main et un développement doivent être de nouveau effectués. Ces outils étant complexes, les industriels préfèrent utiliser l'expérience acquise sur le terrain.

La volonté de la méthode développée durant cette thèse est l'utilisation d'un nouvel outil logiciel tout au long du cycle de développement d'une application pick & place. Depuis le développement d'une cellule robotique, en passant par le développement et les tests d'algorithmes pour finir avec le passage de la simulation à l'expérimentation (génération du programme pour les contrôleurs industriels des robots), le même outil logiciel sera utilisé. Il s'agit d'une contrainte forte mais aussi originale de notre travail qui est justifiée par la finalité des travaux de cette thèse dont la vocation est une utilisation industrielle.

Dans ce chapitre nous allons présenter dans la section 3.2 des outils logiciels permettant de réaliser les étapes du cycle de développement d'une application pick & place. La section 3.3 présentera l'outil logiciel choisi pour réaliser la méthodologie développée. Avec la section 3.4, nous montrerons l'utilisation de l'outil logiciel choisi pour le développement d'une application pick & place. Finalement, la section 3.5 présentera la transition entre la simulation et l'expérimentation ainsi qu'une vue globale des démonstrateurs utilisés.

3.2 Outils logiciels

L'intérêt essentiel de la simulation est qu'elle permet de travailler à moindre coup et sans le recours à une partie opérative réelle évitant ainsi les contraintes de délai et de sécurité. Les algorithmes de tests peuvent être répétés plusieurs fois, en continu, sans contraintes mécaniques et de sécurité. Les résultats peuvent être obtenus plus rapidement car le temps de simulation peut être souvent accéléré. De plus, il y a un intérêt stratégique, plusieurs solutions peuvent être montrées aux clients avant une prise de décision et d'achat. Elle permet d'accompagner le client dans la définition et la mise en place d'une solution. Lorsque la simulation est faite en parallèle de la construction d'une machine, suivant son stade de développement, elle peut aider à la formation des opérateurs avant qu'ils ne soient postés sur une machine réelle. De plus, elle permet de développer, de tester et de valider des processus (gestion de flux, dimensionnement etc.) avant la mise en service sur site et en parallèle de la construction de la machine, ce qui permet de gagner du temps. Elle autorise aussi l'amélioration d'applications existantes sans avoir à arrêter une chaîne de production.

Comme indiqué en introduction il existe plusieurs logiciels et outils logiciels répondant à chaque étape du cycle de développement de la Figure 3.1. Nous allons voir, dans un premier temps, les logiciels utilisés pour l'étape *Développement Robot et Environnement* du cycle. Cette étape se décompose en deux parties. La première partie est le développement cinématique et dynamique du robot. On utilise des logiciels de visualisation 3D afin de vérifier les mouvements cinématiques et dynamiques d'un robot. La seconde partie est la visualisation et la gestion du flux dans l'application. On utilise pour cela des logiciels de gestion de flux qui vont visualiser l'ensemble de l'application : robots, convoyeurs, produits et boîtes afin de vérifier et d'améliorer le fonctionnement, au niveau flux de produits et de boîtes, de la chaîne de production. L'étape *Développement Algorithmes* peut être aussi effectuée à l'aide des logiciels de gestion de flux. Dans un second temps nous présenterons un aperçu des logiciels utilisés afin de réaliser la programmation des contrôleurs commandant les applications pick & place. Dans un troisième temps, nous présenterons

un état de l'art des logiciels développés par les concurrents directs de Schneider Electric. Finalement, des outils logiciels "universels", réalisant plusieurs étapes du cycle de développement, seront présentés.

3.2.1 Logiciels de CAO et de visualisation 3D

Dans la littérature, il existe plusieurs travaux dédiés à la simulation de la robotique pick & place. Cependant, ces travaux sont seulement utilisés pour de la visualisation, afin de vérifier la cinématique et la dynamique. La simulation est aussi utilisée pour le développement du design des robots afin de valider leur comportement, leurs déplacements ou leur interaction avec l'environnement (détection d'obstacles).

Johari *et al.* [Johari et al., 2007] ont utilisé *Workspace 5* pour visualiser une application robotique dans le but de détecter les collisions entre les robots et leur environnement. Le système étudié est un robot qui palettise des cartons à la fin d'une chaîne de production. Le projet utilise une programmation hors ligne et aucun langage de robot n'est généré. Sam *et al.* [Sam et al., 2012] ont conçu un système robotique pick & place utilisant le logiciel *SolidWorks SoftMotion*. Ce système est composé d'un robot cartésien, de plusieurs robots articulés avec différentes pinces et de plusieurs convoyeurs. Le logiciel a mis en évidence rapidement des erreurs de conception. L'étude de ce système à l'aide de la simulation a permis de réduire le temps de conception et d'améliorer les performances du système. De plus, le logiciel autorise une étude du système soumis à des contraintes pour s'assurer de la sécurité.

Workspace 5 est un logiciel de simulation 3D de robot de la société Watson Automation Technical Solutions Ltd, voir Figure 3.2.a. Certaines de ses fonctionnalités sont la détection de collisions, la vérification du positionnement des robots et la programmation hors-ligne.

SoftMotion est un module de simulation du logiciel *LabVIEW* de la société National Instruments pouvant interagir avec le logiciel *SolidWorks* de la société Dassault Système, voir Figure 3.2.b. Il permet de simuler des systèmes avec des profils de mouvements réels, en prenant en compte la dynamique et la cinématique ainsi que les temps de cycle. Il permet aussi de visualiser et d'optimiser la conception et d'évaluer les différents concepts de design avant d'utiliser un prototype physique.

Un autre logiciel pouvant réaliser la simulation de robots pick & place est le logiciel *Visual Components*. C'est un logiciel de simulation 3D de ligne de production, voir Figure 3.2.c. Il peut être utilisé pour l'optimisation d'usine et la configuration de chaînes de production. Il permet de simuler et visualiser en 3D différentes lignes de production.

ADEFID (Advanced Engineering platForm for Industrial Development) a été créé par Dr. Max A. Gonzalez Palacios, voir Figure 3.2.d [González-Palacios, 2012]. C'est un kit de développement de logiciel écrit en MS Visual Studio®C++. Il est consacré aux applications industrielles, à la recherche et au développement dans le domaine des systèmes mécaniques. Il est composé de plusieurs outils et applications.

— *ADRS (Architecture Design and Robot Simulation)* : un outil auxiliaire pour aider à la conception de manipulateurs. Cette application fournit à l'utilisateur des outils pour concevoir de manière interactive une architecture en spécifiant des paramètres

de conception comme la longueur, la torsion, le décalage et l'angle entre chaque lien. Ils sont ensuite modifiés en ligne.

— *ADRS-PP* : une application pour simuler un processus de pick & place.

— *RoboString* : une application dérivée d'ADRS permettant la synchronisation de trois robots.

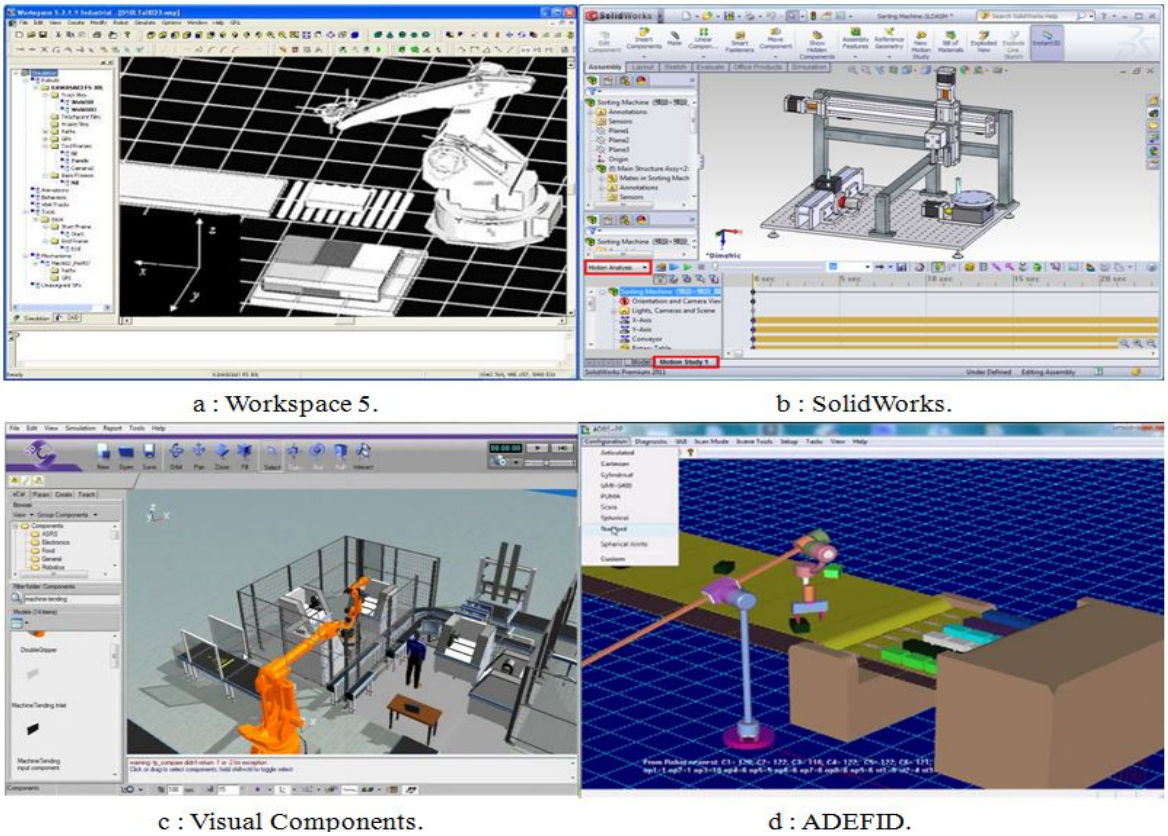

FIGURE 3.2 – Logiciel de simulation comportementale.

Le coût d'investissement pour l'achat de ce genre de logiciel et le temps nécessaire à la prise en main étant important, seul un descriptif global a pu être effectué. De plus, ces logiciels se limitant à une seule étape du cycle de développement, ils ne sont pas conformes au cahier des charges fixé.

3.2.2 Logiciels de gestion de flux

Afin d'améliorer les performances d'une application pick & place multi-robots, la gestion de flux doit être prise en compte et optimisée. Il existe de nombreux logiciels dédiés à la simulation de flux. Nous avons pu évaluer plusieurs d'entre eux.

Witness est un logiciel de simulation de flux de la société Lanner, voir Figure 3.3.a. C'est un outil d'aide à la décision utilisé pour modéliser des scénarii allant de la conception d'installations aux problèmes de ressources humaines et de logistique. Il possède plusieurs spécificités.

— Le logiciel possède une interface intuitive, la prise en main se fait rapidement.
— Il permet de développer des modèles de très grande taille ainsi que des sous-modèles duplicables.
— Il possède un module d'optimisation, Witness Optimizer, recherchant automatiquement la solution optimale de plusieurs paramètres : temps de cycle, nombre d'opérateurs, taille des stocks pour un objectif donné.
— Il permet d'importer des fichiers CAD et XML, d'échanger avec Excel, Oracle ou SGL Server et les simulations peuvent être pilotées via VBA et C#.

Néanmoins, il possède quelques limitations.

— C'est un logiciel de gestion de flux, ce qui entraîne une animation très basique.
— Il n'y a pas d'applications pick & place à proprement parlé, les produits sont assemblés avec les cartons au lieu d'être déposés à une certaine place.
— Il ne gère pas plusieurs flux sur les convoyeurs.
— Il faut ajouter autant de convoyeurs qu'il y a besoin de lignes de flux (donc incapacité de faire un réel flux aléatoire).
— La communication avec l'extérieur est limitée, il n'y a pas de gestion de bus de terrain.

Simul8 est un logiciel de la société SimCore, voir Figure 3.3.b. Il s'agit d'un outil pour la planification, la conception, l'optimisation et le réarrangement de production réelle, la fabrication et la logistique. Simul8 permet d'évaluer des changements sur une ligne de production et de réduire les coûts avant d'investir, de tester des nouvelles idées avant implantation, d'identifier les dysfonctionnements et ainsi devenir plus performant tout en augmentant la productivité et de comparer différents scénarii prenant en compte toutes les alternatives pour de meilleures décisions.

Arena est un logiciel de simulation de flux de la société Rockwell, voir Figure 3.3.c. *Arena* est un logiciel de simulation d'événements discrets et d'automatisation. Il utilise le processeur et le langage de simulation SIMAN (SIMulation Analysis).

Simio est un logiciel conçu et mis en œuvre par la même équipe qui a développé et mis en œuvre Arena, voir Figure 3.3.d. *Simio* a été implémenté avec la technologie .NET. Les objets de Simio se définissent en utilisant des flux de processus qui peuvent être visualisés et modifiés par l'utilisateur. Il a été conçu comme un produit totalement orienté objets. Il possède plusieurs spécificités majeures.

— Le logiciel est intuitif, toutefois moins que le logiciel Witness.
— Il utilise des objets intelligents. On peut définir différents comportements, propriétés, états, déclenchements d'actions, etc. Il est donc flexible et personnalisable.
— Il permet de générer plusieurs scénarii simultanément (simulation de différentes valeurs de paramètres).
— Il possède de nombreux rapports et indicateurs (taux d'utilisation, évolution du nombre de produits, détection de blocage etc.).
— Il possède aussi un module d'optimisation, OptQuest, permettant la création automatique de scénarii pour la recherche itérative de la meilleure solution.

Cependant il possède les mêmes limitations que le logiciel *Witness*.

Meiyun *et al.* [Meiyun and Heguo, 2011] ont utilisé le logiciel de simulation *Witness* pour analyser l'ensemble d'un système de production afin de trouver les goulots d'étranglement dans ce système. En analysant ces goulots, les défauts du système ma-

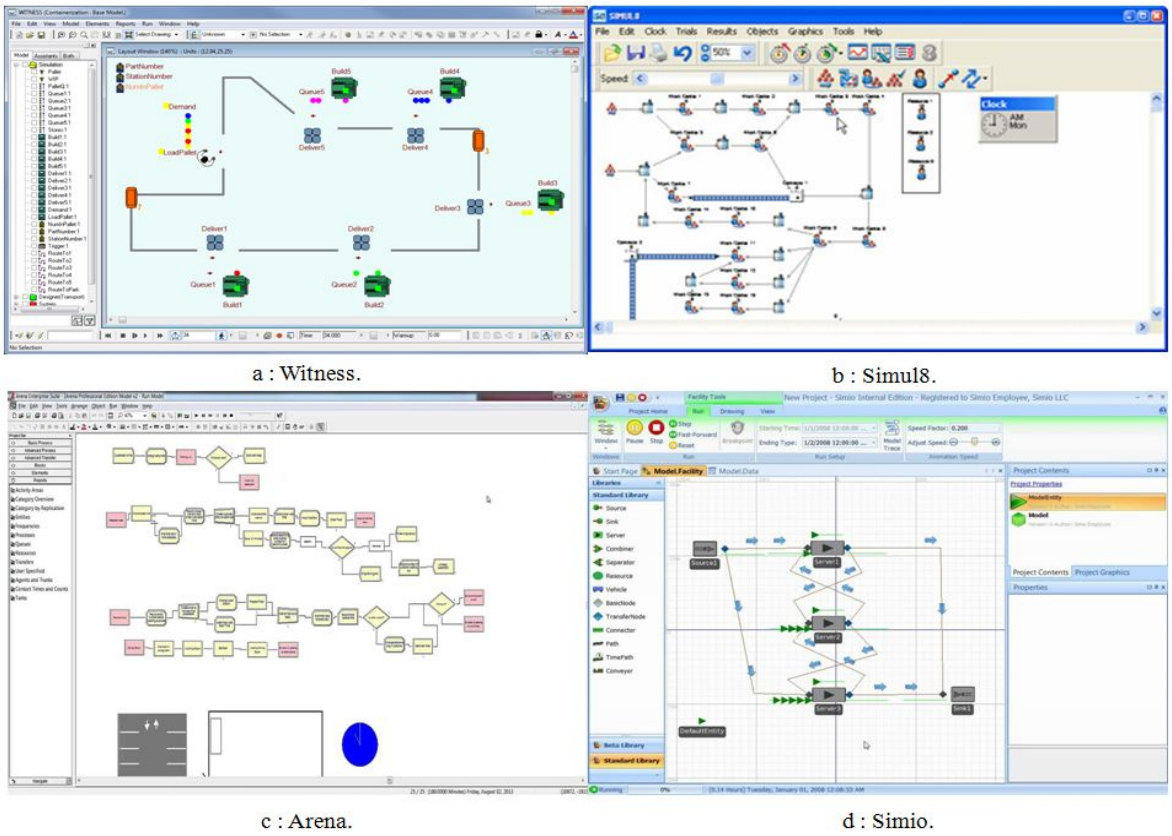

FIGURE 3.3 – Logiciel de simulation de flux.

nufacturier, qui influent directement sur l'efficacité de la production, peuvent être supprimés (meilleure répartition des produits entre les machines etc.). Mirzapourrezaei *et al.* [Mirzapourrezaei et al., 2011] ont aussi utilisé le logiciel *Witness* pour évaluer différents aspects d'un système manufacturier. L'objectif était d'augmenter la productivité et l'efficacité d'une ligne manufacturière. Pegden *et al.* [Pegden and Sturrock, 2009] [Pegden, 2007] ont proposé *Simio*, un logiciel basé sur des objets intelligents. Sturrock *et al.* [Sturrock and Pegden, 2011] ont décrit quelques récentes innovations et améliorations apportées à *Simio*. Concannon *et al.* [Concannon et al., 2003] ont décrit certaines des exigences, avantages et caractéristiques de *Simul8*, un logiciel qui utilise des techniques de planification de production et d'ordonnancement. Hindle *et al.* [Hindle and Duffin, 2006] ont examiné une étude de cas où l'outil *Simul8* a été utilisé pour répondre à un problème d'ordonnancement complexe à travers un centre de fabrication de composites. Nikakhtar *et al.* [Nikakhtar et al., 2011] ont comparé deux outils de simulation parmi les plus connus : *Arena* et *Witness*.

Comme indiqué, tous ces programmes sont principalement dédiés à la simulation de flux. Les visualisations sont très basiques, et principalement axées sur les flux. Le déplacement des produits ou des boîtes sur des convoyeurs peut être visualisé avec *Simio 3D* ou *Witness Quick3D*^{TM}, mais il ne peut pas prendre en compte plusieurs produits sur la largeur d'un convoyeur et seule la durée moyenne d'exécution d'un robot est simulée et

non, la durée des mouvements de l'effecteur. Cet aspect 3D est très important pour un logiciel de développement d'applications pick & place car il permet de vérifier facilement, par exemple, que les produits sont assignés aux bons robots, que les robots prennent les bons produits puis les déposent dans les bonnes places ou encore que les robots prennent correctement les produits à leur bonne position.

3.2.3 Logiciels de programmation des contrôleurs

Comme indiqué dans la section 2.1.2, chaque fabriquant de contrôleur possède son propre logiciel de programmation. Tous ces logiciels utilisent les langages de la norme CEI 61131-3. Les points pouvant différencier un logiciel d'un autre sont l'interface utilisateur, le type de compilation et la gestion des variables. Ces logiciels représentant un coût d'achat important, seule une liste des principaux logiciels a été établie.

— *CoDeSys* de 3S-Smart Software Solutions GmbH.
— *SoMachine Motion* pour Schneider Electric.
— *Sinumerik* pour Siemens.
— *Val3* pour Staübli.
— *Automation Builder* pour ABB.
— *Mapp* pour B&R.

Par ailleurs, ce travail étant réalisé pour Schneider Electic sur des robots Schneider Electric, c'est l'atelier de programmation *SoMachine Motion* qui sera utilisé.

3.2.4 Logiciels des concurrents de Schneider Electric

Certaines entreprises citées précédemment développent d'autres logiciels afin d'acquérir un savoir-faire autre que la programmation des contrôleurs. Ces logiciels peuvent avoir des vocations multiples : permettre l'amélioration des performances des applications pick & place en temps réel, représenter en simulation une cellule robotique etc. La présentation de ces logiciels est faite de façon générale pour plusieurs raisons. Le coût d'acquisition de ces logiciels est important. De plus, le temps afin de réaliser une étude comparative entre ces logiciels est aussi important. Finalement, l'accès aux informations est très restreint. Peu d'informations sont divulguées pour ne pas aider la concurrence. Cela ne permet donc pas d'évaluer les performances des logiciels, mais seulement de donner un aperçu des fonctionnalités. Dans la concurrence, nous pouvons trouver cinq grandes entreprises qui proposent ce genre de logiciels :

— Staübli : le logiciel *LineManager* gère dynamiquement la charge de travail entre les robots [Staubli, 2013].
— ABB : les logiciels *RobotStudio* et *PickMaster* avec l'add-on *PowerPac Picking* offre un environnement configurable pour tester différentes applications [ABB, 2014]. Stumm *et al.* [Stumm et al., 2014] ont utilisé ce logiciel pour montrer une représentation réaliste des flux de produits et les changements dynamiques dans les flux de produits dans des applications pick & place.
— Keba : le logiciel *Real World Simulation Software Package* peut tester des stratégies de pick & place [Keba, 2014].
— Bosch Rexroth : le logiciel *IndraWorks* gère dynamiquement la charge de travail entre les robots [Bosch, 2011].

— Fanuc : le logiciel *Genkotsu* améliore la vitesse et les performances de leurs robots par apprentissage [Fanuc, 2013].

3.2.5 Outils logiciels "universels"

Nous avons vu précédemment que les industriels développent d'autres logiciels afin de balayer un spectre plus large du cycle de développement d'une application pick & place. Dans la littérature, nous pouvons trouver d'autres logiciels qui ont la même optique.

Récemment, *V-REP (Virtual Robot Experimentation Platform)*, un logiciel de la société Suisse Coppelia Robotics, prend en compte plusieurs étapes du cycle de développement d'une application. *V-REP* est un simulateur de robots orienté objets : chaque objet peut être contrôlé individuellement via un script intégré ou un plugin. Les contrôleurs peuvent être écrits en C++, Python ou Matlab. Ce logiciel est utilisé pour le développement rapide d'algorithmes, pour du prototypage rapide, de la robotique pour l'éducation ou pour du contrôle de sécurité. Il contient plusieurs fonctionnalités : détection de collision, solveur de cinématique direct et inverse, planification de mouvement, capteur de vision et de proximité etc.

Un autre logiciel, *Gazebo* [Aguero et al., 2015], a été initialement développé par Dr. Andrew Horward et Nate Koenig [Koenig and Howard, 2004]. *Gazebo* est un outil de simulation 3D de robots permettant de tester rapidement des algorithmes, la conception des robots et d'effectuer des tests régressifs utilisant des scénarii réalistes. Il donne la possibilité de simuler plusieurs robots dans leur environnement (intérieur ou extérieur). Il possède plusieurs fonctionnalités : l'accès à des moteurs physiques hautes performances, un rendu réaliste de l'environnement (lumière, texture etc.), la génération de capteurs (laser, contact, Kinect etc.), le développement de plugins pour le contrôle des robots, des capteurs et de l'environnement etc. Depuis quelques années, il est utilisé par le projet ROS (Robot Operating System) qui est un ensemble d'outils, de bibliothèques et de conventions visant à simplifier la création de comportements pour des robots complexes et robustes à travers une grande variété de plates-formes robotiques.

3.2.6 Conclusion sur les outils logiciels

Nous avons vu dans cette section qu'il existe des logiciels performants répondant parfaitement à un seul besoin du cycle de développement de la Figure 3.1. Il est donc nécessaire qu'à chaque étape, une prise en main et un développement soient de nouveau effectués. Pour cette raison et à cause de la complexité de leur utilisation, les industriels préfèrent continuer à se baser sur l'expérience acquise sur le terrain. Cependant cette tendance change. En effet, les industriels commencent à développer leur propre logiciel afin de réaliser d'autres étapes du cycle de développement autres que la programmation de contrôleur. De plus dans la littérature, il émerge des outils logiciels permettant de balayer le cycle. Cela appuie la démarche de cette thèse : l'utilisation d'un outil logiciel tout au long du cycle de développement d'une application pick & place qui prend en compte les quatre aspects suivants : une simulation du comportement des robots, une simulation de l'environnement de travail, le travail collaboratif de plusieurs robots et la possibilité d'aller de la simulation à l'expérimentation de manière plus automatisée et avec méthodologie. Dans la section suivante, nous allons détailler l'outil logiciel choisi afin de réaliser

les étapes du cycle de développement de la Figure 3.1.

3.3 Présentation *SOSi*^{TM}

Durant la phase état de l'art concernant les logiciels, une opportunité s'est présentée dans le cadre d'une collaboration entre Schneider Electric, l'INSA de Lyon et Solystic. Solystic est une société française spécialisée dans le tri postal qui conçoit et fabrique différents équipements et systèmes de tri postal dans le monde, voir Figure 3.4. Cette collaboration a permis l'utilisation de *SOSi*^{TM}, un logiciel de simulation interne à la société Solystic, afin de développer une application pick & place.

FIGURE 3.4 – Exemple de machine de tri postal conçue par Solystic.

SOSi^{TM} (SOlystic Simulation) a été conçu intégralement par Solystic devant la nécessité de virtualiser leurs systèmes pour la conception. En effet, le développement et l'intégration afin d'améliorer en continu leurs programmes se faisaient directement sur les machines. De plus, Solystic ne trouvait sur le marché aucun outil de simulation satisfaisant leurs besoins d'une solution non intrusive mariant temps réel et production rapide. *SOSi*^{TM} est un outil logiciel de simulation/émulation pourvu d'une visualisation 3D. Sa conception est telle qu'il se comporte de la même manière et avec la même vélocité qu'un ensemble électromécanique aussi complexe et exigeant en terme de vitesse que les machines de tri Solystic. Il fournit un mécanisme logiciel garantissant une exécution temps réel (temps de cycle de 1 ms pour 300 axes) dans un contexte 3D, une gestion de bus de terrain temps réel, l'émulation d'un grand nombre de capteurs/actionneurs (cellule, moteurs, variateurs, etc.). Une machine virtuelle développée avec *SOSi*^{TM} utilise une grande partie des données techniques d'une machine réelle. Cet outil logiciel permet de dérouler des fiches de tests sur une machine sans aléas électromécaniques, étape *Développement Robot et Environnement* du cycle de développement de la Figure 3.1. Ceci permet de monter en fonctionnalité, d'avoir une intégration continue pendant le développement et d'avoir un capital de tests de non régression. De plus, ce logiciel est utilisé pour la validation de processus et d'ergonomie. *SOSI*^{TM} intègre des agents programmés capables

d'interagir avec le monde virtuel. Il permet d'avoir une optimisation de l'exploitation en considérant la pénibilité des tâches (charges portées, distances parcourues, etc.) et la formation. Finalement, ce logiciel permet de réinjecter des semaines entières d'exploitation réelle de leurs machines et de les "rejouer" dans un contexte virtuel en vitesse accélérée, voir Figure 3.5. Ceci leur permet de développer et de prévoir de nouveaux processus.

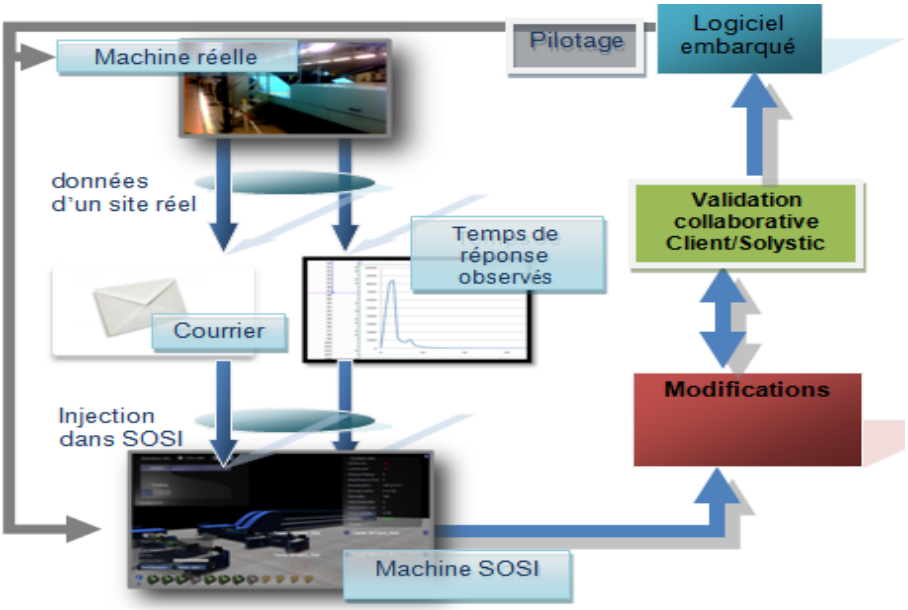

FIGURE 3.5 – Schéma d'optimisation In-Situ.

Les caractéristiques de SOSi^{TM} sont donc :
- Une reproduction au plus proche du comportement d'une solution réelle.
- Une observation et une analyse des impacts du changement avant mise en exploitation dans l'environnement opérationnel.
- Une simulation qui peut aller du simple équipement jusqu'au centre de tri complet.
- La non-mobilisation des équipements déjà installés lors des tests d'intégration.
- La validation du dimensionnement des outils industriels.
- La sécurisation de la mise en exploitation des changements et évolutions.
- La validation du calcul du Retour sur Investissement pour les évolutions envisagées.

3.4 Développement d'une application pick & place avec *SOSi*^{TM}

3.4.1 Utilisation de *SOSi*^{TM} pour une application pick & place

À travers cet outil, une nouvelle méthode pour la conception d'applications pick & place a été développée dans les travaux de cette thèse. Les différentes étapes sont liées afin d'éviter d'avoir un nouveau développement à réaliser à chaque fois, ce qui permet de gagner du temps. Cette méthode permet la validation d'une grande partie de l'application avant sa mise en service. Elle permet aussi d'obtenir une évaluation prévisionnelle des

performances. Le développement de la partie physique peut se faire en parallèle de la partie logicielle. Une fois l'application mise en fonctionnement, il y a la possibilité d'améliorer la stratégie de pilotage sans avoir recours à un arrêt prolongé de la production. Le programme est d'abord modifié, puis validé en simulation et est ensuite implémenté sur site. Cela permet donc de réduire les coûts liés à l'arrêt de la production.

Contrairement à d'autres travaux, l'environnement logiciel développé dans notre travail permet de créer une cellule virtuelle, fonctionnant en 3D et en temps réel, reposant sur le modèle CAO de la cellule réelle. Cette cellule virtuelle possède la même cinématique et la même dynamique que la cellule réelle, de plus son environnement peut aussi être simulé : produits arrivant sur un convoyeur, etc. Des scénarii peuvent être mis en place afin de vérifier son comportement. Une couche de programmation haut niveau peut être utilisée afin de mettre en place une stratégie de prise et de dépose des produits ainsi qu'une stratégie de collaboration entre plusieurs robots, si besoin est. Ce logiciel est de plus modulaire, il permet de configurer le système de production (robots, convoyeurs, etc.), son environnement (produits, boîtes, etc.) et les différents scénarii.

Le simulateur possède aussi une gestion des bus de terrain qui sont généralement en place entre la commande et la partie opérative et un noyau temps réel. Les utilisateurs peuvent ainsi vérifier le comportement de différentes stratégies de pick & place sur leur ordinateur avec une simulation fine où le comportement d'un contrôleur industriel réel est inclus. L'idée est d'exécuter les simulations, soit sur un simulateur avec un comportement proche du système de production réel, soit sur un hardware cible.

$SOSi^{TM}$ étant initialement développé pour la conception et l'évolution de systèmes de tri postal, qui n'a que peu de similarités avec une application pick & place, un développement depuis zéro a dû être fait. Afin d'anticiper l'intégration dans le contrôleur, l'architecture Figure 3.6 a été conçue. Cette architecture permet de séparer une application pick & place en fonctionnalités (ce sont les différentes couleurs de boîtes). L'outil développé, à travers cette architecture, intègre deux niveaux de stratégies. Une règle de planification individuelle simple (boîtes rouges claire), pour un seul robot, qui donne au robot (boîtes violettes) la position du produit ou de la place. Les robots peuvent ensuite prendre ou déposer les produits sur les convoyeurs (boîtes grises). Ces derniers déplacent les produits et les boîtes selon une certaine vitesse. Des stratégies de collaboration peuvent aussi être utilisées (boîte rouge foncé). Ce sont des algorithmes avec une couche supérieure. Les produits peuvent être assignés aux robots avant qu'ils n'arrivent dans leur zone de travail par exemple. De plus, cette architecture permet de gérer la génération (boîte bleue) de produits et de boîtes (boîtes orange) sur les convoyeurs d'entrée et de sortie. Les produits et les boîtes sont considérés comme des fonctionnalités à part entière afin de prendre en compte toutes leurs caractéristiques (poids, type, quantité maximum, etc.).

3.4.2 Création d'une application pick & place

Afin d'illustrer la création d'une application pick & place, nous allons utiliser l'exemple de la figure 3.7. Il est composé de quatre robots et de deux convoyeurs. Chaque robot est composé d'une base, et d'un effecteur. Cet exemple sera la base de l'application pick & place utilisée dans cette thèse.

La création d'une application pick & place à l'aide de $SOSi^{TM}$ est composée de deux

FIGURE 3.6 – Architecture de simulation avec deux niveaux de stratégies (règles de planification individuelles et stratégies collaboratives).

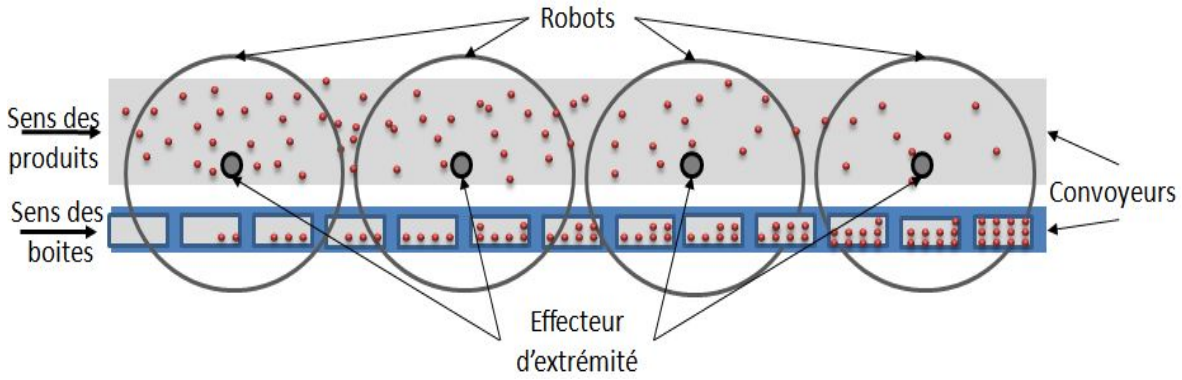

FIGURE 3.7 – Schéma d'une application pick & place.

étapes : le prototypage et le développement de plugins.

Le prototypage a pour mission de fournir une description détaillée tant au niveau fonctionnel qu'au niveau géométrique (étape *Développement Robot et Environnement* de la Figure 3.1). Son support est le fichier S3D. La définition du modèle de simulation est effectuée avec un logiciel de CAO 3D, *Autodesk 3ds Max* par exemple. Le modèle peut être soit importé via un fichier CAO (venant par exemple de *CATIA*) ou créé directement à l'aide des fonctionnalités d'*Autodesk 3ds Max*, voir Figure 3.8. Chaque convoyeur est associé directement à son propre moteur. Chaque robot est associé à deux moteurs, un pour la translation de l'effecteur via un variateur et un pour la rotation de l'effecteur. Différents éléments ont été ajoutés afin de simuler des interfaces homme-machine.

Le développement de plugins fournit le fond nécessaire à une simulation ou à une émulation (étape *Développement Robot et Environnement* du cycle de développement, voir Figure 3.1). Il permet de définir et de fournir des objets virtuels actifs au sein de l'application : moteurs, variateurs, capteurs, etc. Il permet d'animer le comportement (interactions, animations, etc.) des ensembles électromécaniques. Les comportements ci-

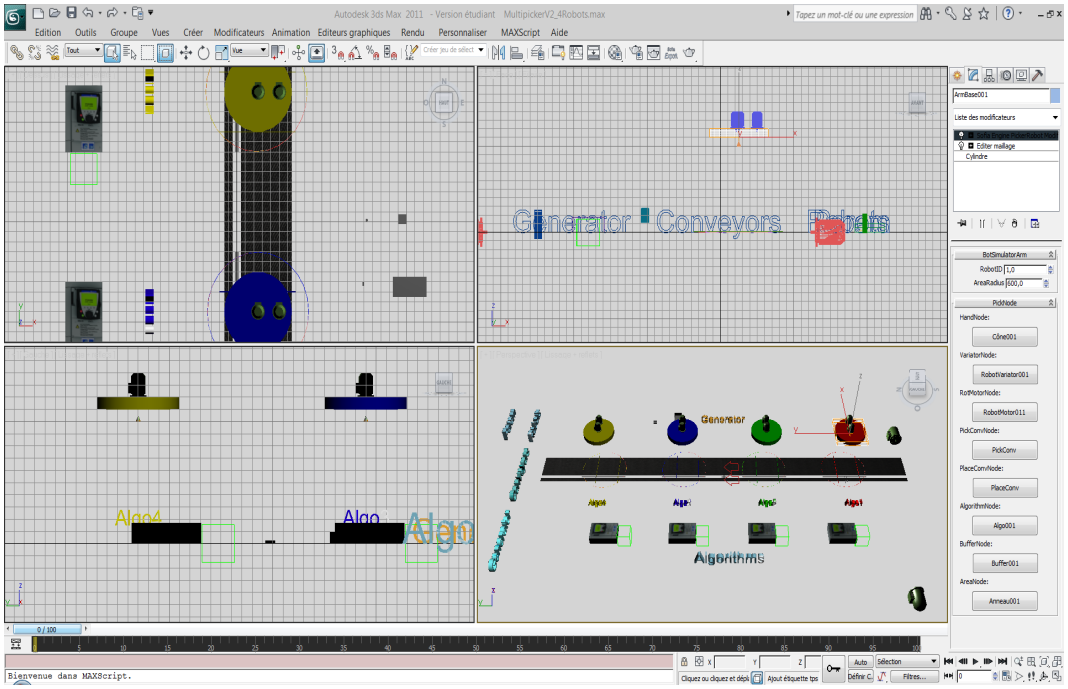

FIGURE 3.8 – Application pick & place sur Autodesk 3ds Max.

nématiques des objets de l'application sont définis en C++ avec *Visual C++* selon les comportements suivants :

- Les robots ont le fonctionnement décrit sur la Figure 3.9.
  - Lors des étapes *Attendre_produit* et *Attendre_place*, les robots attendent l'information de position d'un produit ou d'une place qui est fournie par les blocs stratégies.
  - Lors de la présence d'un produit ou d'une place, les robots se déplacent vers cette position avec un mouvement défini (linéaire par exemple).
  - Ce mouvement est calculé tous les temps de cycle de l'outil logiciel, ce qui permet de suivre un produit si celui-ci se déplace sur un convoyeur.
  - La position de l'effecteur est mise à jour à chaque temps de cycle.
  - Le mouvement de translation dépend des caractéristiques du moteur de translation associé au robot (vitesse maximum de translation, accélération de translation).
  - S'il est nécessaire d'effectuer une rotation (pour repositionner un produit par exemple), le mouvement de rotation dépend des caractéristiques du moteur de rotation associé au robot (vitesse maximum de rotation, accélération de rotation).
  - Les robots doivent être capables de prendre et déposer plusieurs produits les uns à la suite des autres (multi-picking/multi-placing).
- Les convoyeurs déplacent les produits (ou places) qui leur sont affectés suivant leur vitesse, leur direction et leur sens. La position des produits (ou places) est mise à jour à chaque temps de cycle.
- Le générateur d'objets crée des produits (ou des places) sur les convoyeurs selon des

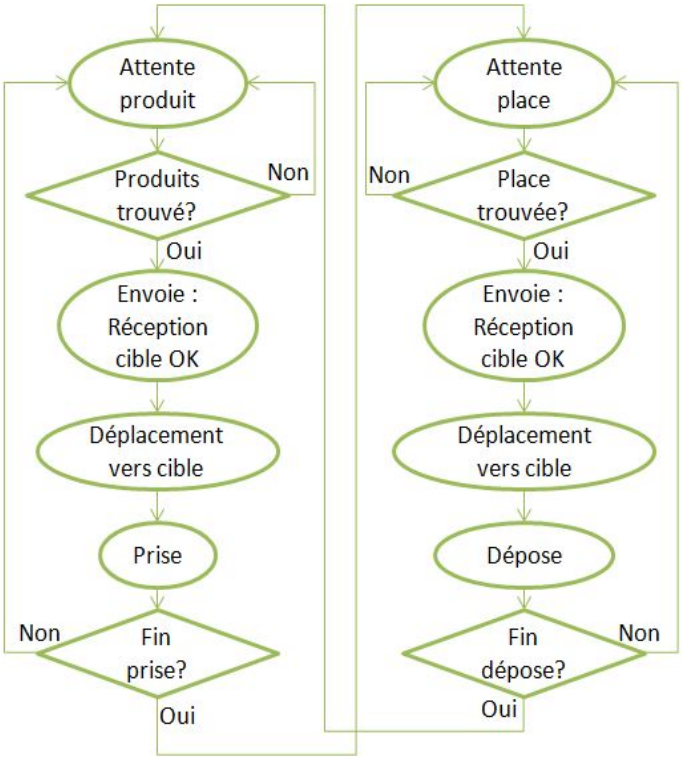

FIGURE 3.9 – Diagramme de fonctionnement d'un robot.

paramètres donnés (position, espacement, orientation, produits disposés en ligne ou aléatoirement etc.). Il doit être capable de faire du panachage (différents types de produits sur le même convoyeur). C'est lui qui donne les caractéristiques aux produits ou aux places (poids, couleur etc.).

— Un autre objet, qui n'est pas représenté sur l'architecture de simulation mais qui peut être utilisé dans des applications pick & place, est le buffer physique. Le buffer stocke des produits en attente d'être repris pour être déposé dans une boîte. Il n'a pas de comportement cinématique, mais doit être considéré comme un élément à part entière car c'est un élément sur lequel les robots agissent et possédant ses propres caractéristiques (position, taille du buffer, position des places dans le buffer, etc.).

En plus de développer le comportement des composants de l'application, les plugins permettent de mettre en place différentes règles de planification individuelles pour chaque robot ou stratégies de collaboration entre les robots (étape *Développement Algorithmes* du cycle de développement). Les règles de planification individuelles donnent aux robots la position ou l'identifiant du produit ou de la place à atteindre qui se situe dans leur zone de travail. Quant aux stratégies de collaboration, elles gèrent la collaboration entre les robots, le partage de la charge de travail entre les robots (en assignant les produits aux robots avant qu'ils arrivent dans leur zone de travail ou en donnant des seuils de remplissage des boîtes par exemple). Elles autorisent l'utilisation des règles de planification individuelles. De plus, elles peuvent gérer dynamiquement la vitesse des convoyeurs. Enfin, elles gèrent la prise en compte de l'arrêt d'un robot (lié à une panne par exemple). Les règles et les stratégies développées seront détaillées dans le chapitre 4.

Une fois les plugins créés, ceux-ci doivent être liés à leur composant. Cela doit être effectué dans *Autodesk 3ds Max*. Une des fonctionnalités d'*Autodesk 3ds Max* est la possibilité d'ajouter des attributs aux différents composants créés. Les plugins ont été conçus de telle sorte qu'ils créent un attribut, pour chaque plugin développé, afin de les lier à leur composant. Ceci permet aux composants d'avoir leur comportement, mais également aux plugins d'avoir accès aux caractéristiques de leur composant (taille, position etc.). De plus, ces attributs permettent d'ajouter d'autres caractéristiques aux composants comme une vitesse nominale, un identifiant, une zone de travail, etc.

Dès que le modèle et les plugins pour le comportement ont été développés (robots, convoyeurs, générateur d'objets, produits et boîtes) la simulation peut être effectuée avec l'outil logiciel $SOSi^{TM}$, voir Figure 3.10, pour tester le comportement du modèle (étape *Tests comportementaux* du cycle en développement) :

— Génération de l'environnement (génération des produits ou des boîtes avec les caractéristiques souhaitées et placés aux endroits voulus etc.).
— Déplacement des objets sur les convoyeurs avec la bonne vitesse.
— Déplacement de l'effecteur d'extrémité des robots avec un mouvement correct.
— Prise et dépose des produits au bon endroit.

Si le modèle correspond aux attentes et que les plugins pour les stratégies sont développés, la simulation peut de nouveau être effectuée afin de tester le comportement des algorithmes (les règles individuelles donnent le bon produit ou la bonne place et les stratégies de collaboration fonctionnent correctement) et d'analyser les résultats obtenus avec les différentes stratégies de pick & place (étape *Tests Algorithmes* du cycle en développement). Cette partie sera détaillée dans le chapitre 5. Une fois les paramètres de la simulation optimisés, des tests expérimentaux peuvent débuter pour vérifier le fonctionnement des algorithmes et tester leurs performances (étapes *Validation Algorithmes* et *Tests Performance* du cycle en développement). Cependant, tout ne peut pas être simulé : le temps de communication avec la vision et entre les algorithmes et les robots, la baisse de précision due aux vibrations, etc. Ce sont ces paramètres qui vont induire des écarts entre la simulation et l'expérimentation.

3.4.3 Conclusion sur l'utilisation de l'outil logiciel $SOSi^{TM}$

Nous avons effectué dans cette section, la simulation d'une application pick & place (partie gauche du cycle de développement de la Figure 3.1). À ce niveau, le développement et les tests cinématiques des robots, le développement et l'interaction des robots avec l'environnement ont été effectués. De plus, nous avons mis en place une architecture de simulation intégrant deux niveaux de stratégies. Les différentes stratégies utilisées seront détaillées dans le chapitre 4. Les prochaines étapes sont la transition entre la simulation et l'expérimentation, puis les tests expérimentaux où les algorithmes et les performances de l'application seront validés.

3.5 Du virtuel au réel

Comme indiqué précédemment, après la simulation, la traduction des algorithmes et des stratégies en langage automate est faite (passage entre la simulation et l'expérimen-

FIGURE 3.10 – Exemple de simulation avec quatre robots et deux convoyeurs co-courant.

tation du cycle en développement de la Figure 3.1). Afin de faciliter cette traduction, les algorithmes sont écrits avec des fonctions les plus simples possible, ce qui permet aussi de réduire le temps d'exécution. De plus, le logiciel automate utilise un langage orienté objet tout comme le logiciel de simulation. L'architecture de la Figure 3.11, mise en place dans le contrôleur, est donc la mise en œuvre de celle de la Figure 3.6. Elle est composée des mêmes blocs de stratégies individuelles (boîtes rouges claires) et collaboratives (boîte rouge foncée) que pour la simulation. On retrouve les robots (boîtes violettes) et les convoyeurs (boîtes grises) avec les produits et les boîtes (boîtes orange), qui sont réels et non simulés, mais possédant les mêmes caractéristiques et les mêmes comportements. Il y a de plus la partie contrôleur avec la logique de commande (boîtes vertes). C'est cette partie qui contrôle les robots et les convoyeurs, et qui structure le programme. Cette similarité entre la simulation et l'expérimentation permet de prendre en compte, dès le début de la création d'une application pick & place, le programme final qui sera implémenté dans le contrôleur. Ce résultat ne peut être obtenu avec un logiciel dédié au comportement du robot ou à la simulation de flux, car cela nécessite un développement logiciel pour aboutir à un code compréhensible pour les robots.

Durant cette thèse, deux démonstrateurs ont été utilisés. Le premier est situé à l'INSA de Lyon et fourni par Schneider Electric, voir Figure 3.12. Le second est situé sur le site de Schneider Electric à Marktheidenfeld en Allemagne, voir Figure 3.13. Les démonstrateurs et leur utilisation seront détaillés dans le chapitre 6.1.

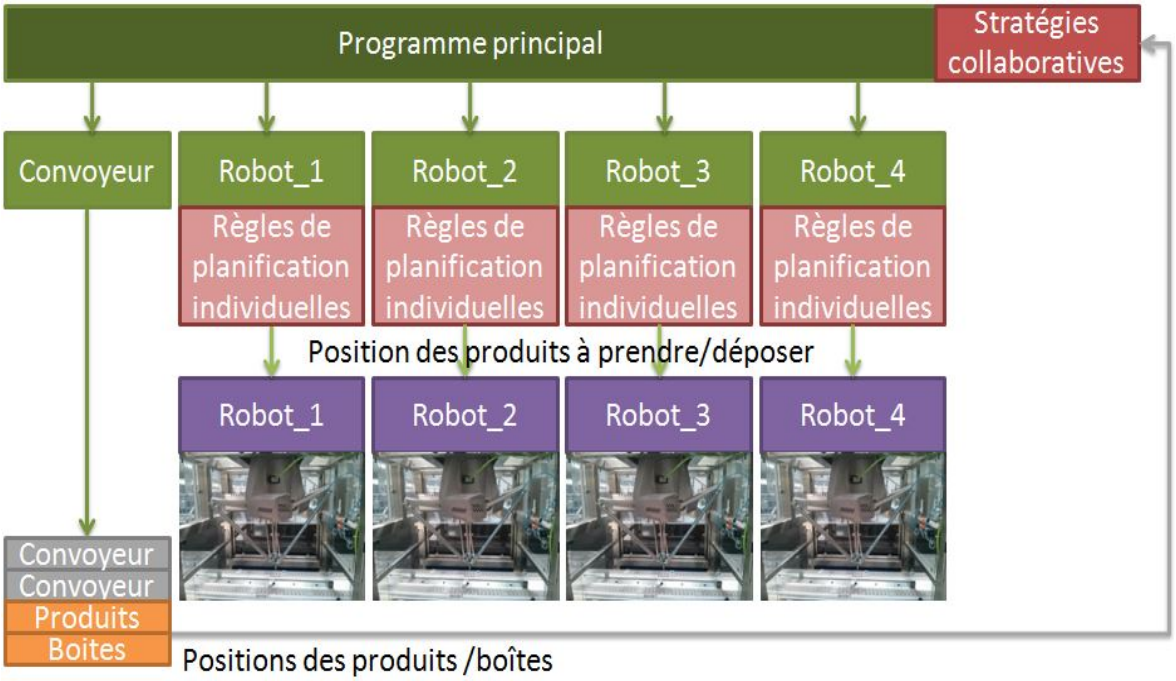

FIGURE 3.11 – Architecture du contrôleur avec deux niveaux de stratégies (règles de planification individuelles et stratégies collaboratives).

FIGURE 3.12 – Démonstrateur avec un robot et un convoyeur en anneau et un contrôleur Schneider Electric LMC400C.

FIGURE 3.13 – Démonstrateur avec trois robots et deux convoyeurs parallèles.

3.6 Conclusion sur la méthodologie pour la mise en place d'une application pick & place

Dans ce chapitre, nous avons présenté une méthodologie pour la mise en place d'une application pick & place. La volonté de cette méthodologie est l'utilisation tout au long du cycle de développement d'une application pick & place d'un outil logiciel unique afin d'aider l'utilisateur. Nous avons montré, à travers un état de l'art, qu'il existe des logiciels performants mais complexes répondant à un seul besoin du cycle de développement. Ceci entraînait les industriels à n'utiliser que leur expérience acquise sur le terrain. Cependant, nous avons indiqué que les industriels ont commencé à acquérir de nouveaux savoir-faire afin de réaliser d'autres étapes du cycle de développement que la programmation de contrôleur. Cette tendance est accompagnée par l'émergence de nouveaux outils logiciels englobant plusieurs étapes du cycle. Cela montre l'intérêt de la méthodologie développée durant cette thèse.

Dans un second temps, nous avons présenté l'outil logiciel retenu afin de développer cette méthodologie : $SOSi^{TM}$. Nous avons ensuite détaillé l'utilisation de $SOSi^{TM}$ pour la création d'une application pick & place en simulation. Le développement de cette simulation comprend le comportement des robots, l'environnement de travail, les lois de commande individuelles pour chaque robot ainsi que le travail collaboratif entre plusieurs robots.

Finalement, nous avons montré la possibilité de passer de la simulation à l'expérimentation de manière simplifiée et rapide en indiquant les similarités entres les architectures de simulation et d'expérimentation. Une présentation globale des démonstrateurs utilisés

pour la partie expérimentation a été faite.

Dans ce chapitre nous avons donc mis en place l'architecture de l'application pick & place, voir Figure 3.6. Le comportement des robots ainsi que de l'environnement a été développé et testé. La prochaine étape est donc le développement des algorithmes permettant d'établir des lois de commande.

Chapitre 4

Algorithmes développés

Sommaire

Dans la section 2.2, nous avons vu qu'il existait beaucoup de travaux sur des algorithmes de pick & place. Ces algorithmes peuvent être séparés en deux catégories : les règles individuelles (règles prenant en compte uniquement un seul robot) et les stratégies de collaboration entre plusieurs robots. Nous avons dit que les règles de planification individuelles développées pouvaient être utilisées pour le type d'applications pick & place traitées dans cette thèse, tandis que, les heuristiques permettant la collaboration entre les robots ne pouvaient pas être exploitées car elles demandaient de trop grandes ressources en terme de temps de calcul. Ne trouvant pas d'autres algorithmes compatibles avec le temps réel nécessaire à nos applications pick & place, il a été décidé d'en développer de nouveaux dans l'optique d'améliorer les performances des systèmes pick & place multi-robots tout en gardant les contraintes de temps réel (étape *Développement Algorithmes* du cycle de développement d'une application pick & place, voir Figure 4.1). Ce chapitre va présenter les règles déjà développées dans la littérature et utilisées dans cette thèse ainsi que les nouveaux algorithmes développés. Avant de commencer, nous allons expliquer l'architecture et les blocs fonctionnels où les différents algorithmes fonctionneront.

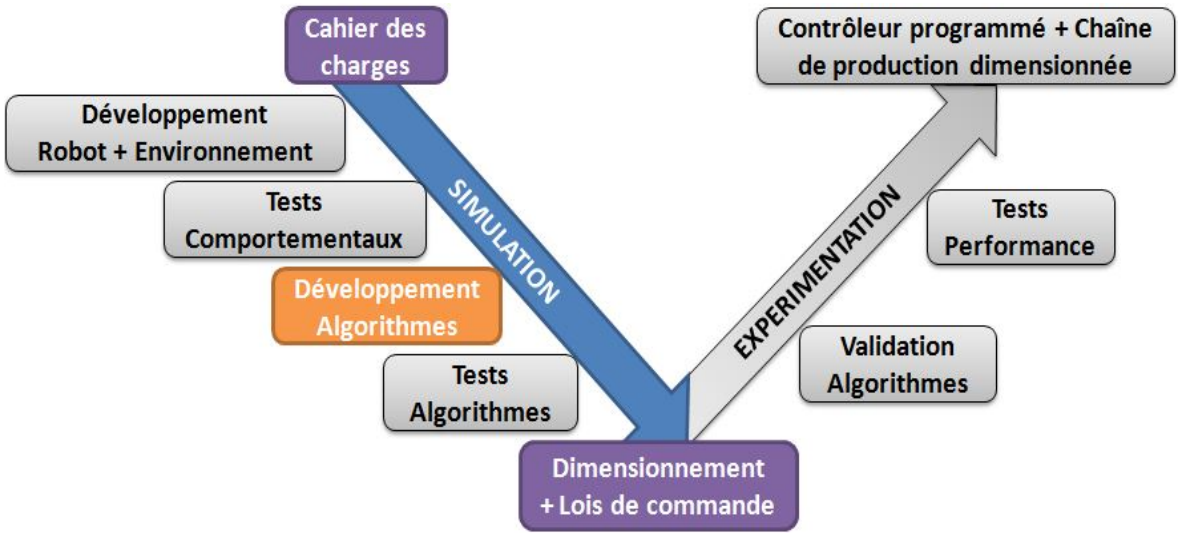

FIGURE 4.1 – Étape *Développement Algorithmes* du cycle de développement d'une application pick & place.

4.1 Architecture et blocs fonctionnels

Comme expliqué dans les sections 3.4 et 3.5, la simulation et l'expérimentation utilisent les mêmes types de blocs fonctionnels pour les règles de planification individuelles et les stratégies de collaboration. Dans les deux cas, leur fonctionnement est identique, seul le langage utilisé change légèrement.

4.1.1 Blocs fonctionnels de règles de planification individuelles

L'objectif des blocs de règles de planification individuelles est de fournir la position ou l'identifiant du produit ou de la place à atteindre par le robot associé. La recherche d'un produit ou d'une place s'effectue exactement de la même façon et suit le diagramme de la Figure 4.2. Dès qu'un robot attend un produit (ou une place), un algorithme de recherche est exécuté. Celui-ci analyse tous les produits (ou places) pour trouver l'identifiant du produit (ou de la place) qui correspond le mieux suivant la règle souhaitée. Une fois le produit (ou la place) trouvé, son identifiant (noté *ID* sur la Figure 4.2) et sa position sont donnés au robot. La synchronisation entre les blocs de règles et de robots se fait à l'aide de drapeaux (*Attente_Produit*, *Produit_Trouve* etc.).

Les blocs ont besoin comme informations :

— De l'identifiant, de la position, des caractéristiques et du nombre des produits et des boîtes.
— De l'identifiant, de la position et de la zone de travail des robots.
— De la position de l'effecteur du robot.
— De l'identifiant de la règle souhaitée.

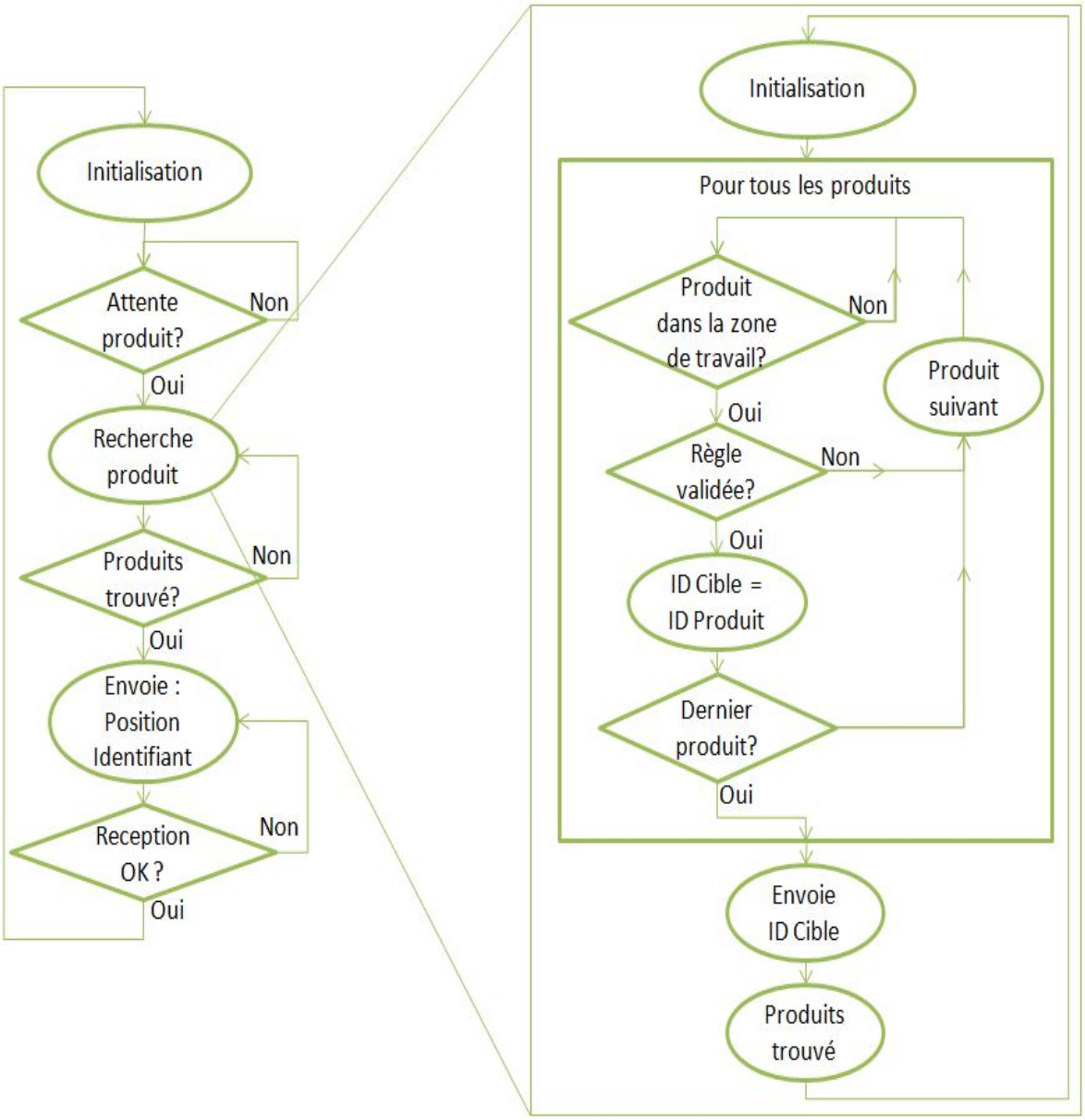

FIGURE 4.2 – Diagramme de fonctionnement du bloc règle de planification individuelle.

4.1.2 Bloc fonctionnel de stratégies de collaboration

L'objectif du bloc de stratégies de collaboration est de gérer les algorithmes de prise et de dépose permettant la collaboration entre les robots et le partage de la charge de travail entre les robots. Il autorise l'utilisation des règles de planification individuelles. De plus, il peut gérer dynamiquement la vitesse des convoyeurs. Enfin, il gère la prise en compte de l'arrêt d'un robot (dû à une panne par exemple). Toutes ces fonctions sont gérées parallèlement. Le bloc a besoin comme informations :

- De l'identifiant, de la position, des caractéristiques et du nombre des produits et des boîtes.
- De l'identifiant, de la position et de la zone de travail et du nombre des robots.
- De la vitesse des convoyeurs.
- De l'identifiant de la stratégie souhaitée.

4.1.3 Conclusion sur l'architecture et les blocs fonctionnels

L'objectif de ces blocs fonctionnels est de simplifier la mise en place d'algorithmes et leur test. De plus, ils ont été développés pour avoir un faible impact sur un programme existant : seuls des drapeaux doivent être ajoutés. Ces blocs contiennent tous les algorithmes que l'on souhaite utiliser et tester. Le changement d'un algorithme par un autre est effectué simplement en modifiant l'identifiant de l'algorithme souhaité, aussi bien pour les règles que pour les stratégies. Il est, de plus, facile de rajouter un algorithme, si nécessaire, dans le bloc fonctionnel. En effet, les algorithmes sont assimilés à des fonctions, il faut associer cette fonction à un identifiant puis sélectionner cet identifiant. Ce qui permet de tester facilement et rapidement différents algorithmes sans avoir à compiler de nouveau le programme.

Les blocs fonctionnels étant créés, les algorithmes allant à l'intérieur peuvent être développés. Ces algorithmes dépendent des caractéristiques des produits qui peuvent être classés en quatre catégories : les produits simples, les produits orientés, les produits avec poids et les produits avec type.

4.2 Produits simples

Un produit simple est un produit sans caractéristique spéciale. On n'a pas besoin de connaître et de prendre en compte son orientation, son poids ou son type afin de le prendre et le déposer correctement. Nous allons voir dans un premier temps, les règles de planification individuelles pouvant être utilisées pour chaque robot. Puis, dans un deuxième temps, différentes stratégies permettant la collaboration entre les robots seront présentées.

4.2.1 Règles de planification individuelles

Comme indiqué dans la section 2.2.1, il existe plusieurs règles de planification individuelles qui ont été proposées dans la littérature pouvant être utilisées pour la prise et la dépose des produits [Mattone et al., 1998] :

— *FIFO* : First In First Out. Le robot prend le premier produit entré dans sa zone de travail. Le robot dépose le produit dans la première place dans sa zone de travail.
— *LIFO* : Last In First Out. Le robot prend le dernier produit entré dans sa zone de travail. Le robot dépose le produit dans la dernière place dans sa zone de travail.
— *SPT* : Shortest Processing Time. Le robot prend le produit le plus proche de son outil dans sa zone de travail. Le robot dépose le produit dans la place la plus proche de son outil dans sa zone de travail.

D'autres règles ont été développées afin de modifier le comportement des robots. Ces règles ont pour objectif de gérer le remplissage des boîtes, et donc, ne peuvent être utilisées que pour la dépose de produits :

— *MinMax* : Le robot remplit la boîte la moins ou la plus remplie dans sa zone de travail. Le choix est fait par une variable entière : 0 pour la moins remplie et 1 pour la plus remplie. Un seuil peut être aussi appliqué. Par exemple, le robot remplit les boîtes remplies à plus de 50% de leur capacité maximum.
— *XPlaces* : Le robot remplit toujours la même boîte avec X produits dans sa zone de travail. Il faut faire attention à ne pas déposer le premier produit dans une boîte trop en aval de la zone de travail du robot, dans le sens du convoyeur de sortie, sinon cette boîte risque de sortir de la zone sans avoir reçue tous les produits voulus.

4.2.2 Stratégies de collaboration

Pour une application de pick & place avec des convoyeurs parallèles (application la plus répandue en industrie), un seuil de performance ne pouvant pas être dépassé peut être identifié. Ce seuil peut être atteint lorsque tous les robots fonctionnent à 100% de leur performance avec une charge de travail identique (il n'y a pas de temps d'attente de produits ou de places). De plus les robots doivent prendre et déposer les pièces en moyenne au milieu des convoyeurs d'entrée et de sortie en suivant une trajectoire perpendiculaire aux convoyeurs, voir Figure 4.3. Lorsque ces conditions sont réunies, les performances de l'application ne peuvent plus être améliorées sans modification des performances mécaniques des robots ou des trajectoires que peuvent effectuer les robots.

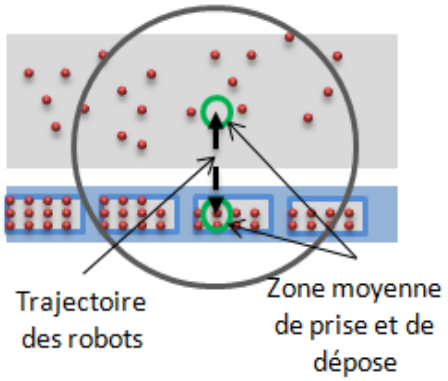

FIGURE 4.3 – Seuil de performance d'une application pick & place avec convoyeurs parallèles.

Afin d'orienter le comportement de prise et de dépose des robots vers le comportement permettant d'atteindre le seuil de performances, des stratégies de collaboration ont été

développées. Elles peuvent être utilisées afin d'assigner les produits aux robots avant qu'ils n'arrivent dans leur zone de travail. Les différentes stratégies de collaboration sont illustrées en utilisant une application pick & place composée de quatre robots. Les produits rouges sont attribués au robot 1, les produits verts au robot 2, les produits bleus au robot 3, les produits jaunes au robot 4. Cependant, les robots peuvent prendre les produits assignés aux robots en amont d'eux dans le sens du convoyeur d'entrée. Par exemple, le robot 3 peut prendre les produits du robot 2 et du robot 1. Cela permet aux autres robots de récupérer les produits ne pouvant pas être pris par les robots en amont. S'il y a une assignation par groupes, ces derniers sont traités séquentiellement. Le groupe le plus clair est traité en premier et le groupe le plus foncé est traité en dernier. Les premières stratégies développées sont à utiliser lorsque les robots prennent et déposent un seul produit à la fois. Elles sont illustrées à l'aide de la Figure 4.4 et sont les suivantes :

— *DownToUp* : Assigne les produits aux robots, un par un, d'aval en amont du convoyeur.
— *IntToExt* : Assigne les produits aux robots, un par un, de l'intérieur à l'extérieur du convoyeur.
— *IntToExt Cyclic* : Assigne les produits aux robots, un par un, de l'intérieur à l'extérieur du convoyeur pour un nombre de produits égal au nombre de robots. Puis l'ordre des robots est changé cycliquement.
— *IntToExt Alt* : Assigne les produits aux robots, un par un, de l'intérieur à l'extérieur du convoyeur pour un nombre de produits égal au nombre de robots. Puis on assigne de l'extérieur à l'intérieur et l'assignation est alternée.
— *Horizontal* : Assigne aux robots une zone horizontale du convoyeur correspondant à son numéro. Cette stratégie a été inspirée par [Izumi et al., 2013].
— *Vertical* : Assigne aux robots une zone verticale du convoyeur correspondant à son numéro. Cette stratégie a été inspirée par [Izumi et al., 2013].

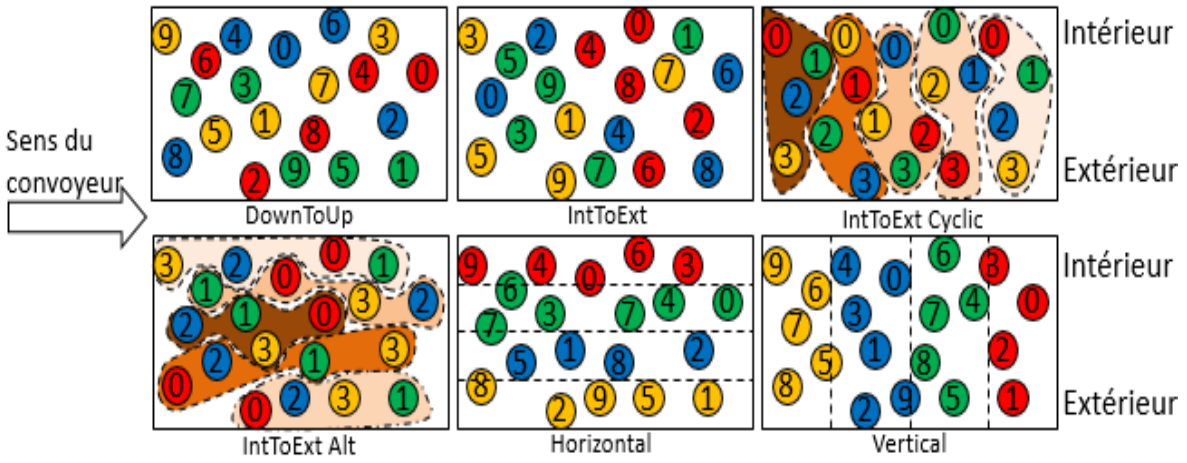

FIGURE 4.4 – Exemple de différentes stratégies de collaboration avec une prise simple.

Ces stratégies peuvent être étendues lorsque les robots prennent et déposent plus d'un produit à la suite. Elles sont illustrées avec l'exemple de la Figure 4.5 qui est la mise en œuvre des stratégies vues précédemment avec quatre robots prenant trois produits les uns à la suite des autres. Les stratégies ont le même algorithme mais au lieu d'assigner les

produits aux robots un par un, les produits seront assignés X par X (ici X est égal à 3).

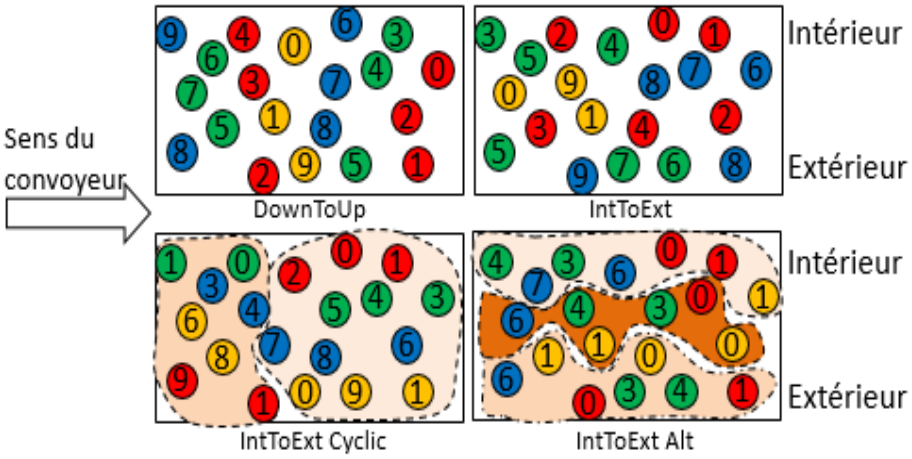

FIGURE 4.5 – Exemple de différentes stratégies de collaboration avec une prise multiple.

Une variante des stratégies pour la prise multiple a été développée. Son principe est l'utilisation de la règle *SPT*, vue dans la section 4.2.1. Le premier produit assigné est trouvé de la même façon qu'avec une stratégie classique. Quand aux produits suivants, ils sont trouvés avec la règle *SPT*. Le produit assigné suivant est le produit le plus proche par rapport au produit précédemment assigné, voir Figure 4.6. Ces stratégies ont pour but de réduire le temps de déplacement entre deux produits.

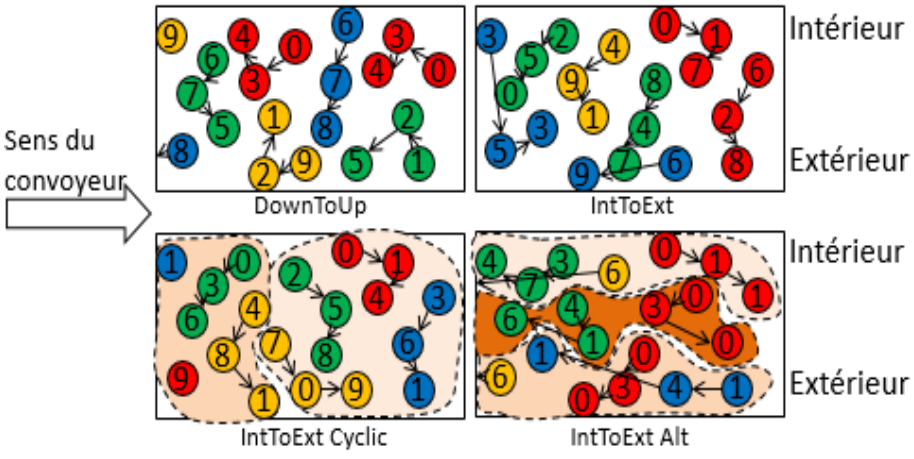

FIGURE 4.6 – Exemple de différentes stratégies de collaboration avec une prise multiple utilisant une règle SPT.

Les stratégies présentées jusqu'à maintenant sont utilisées lorsque les produits sont positionnés aléatoirement sur le convoyeur d'entrée. Pour des produits positionnés en ligne sur le convoyeur, une autre stratégie a été proposée.

— *DownToUp Cyclic* : Assigne les produits aux robots, un par un, de l'amont à l'aval puis de l'intérieur à l'extérieur du convoyeur pour un nombre de produits égal au nombre de robots. Puis l'ordre des robots est changé cycliquement, voir Figure 4.7.

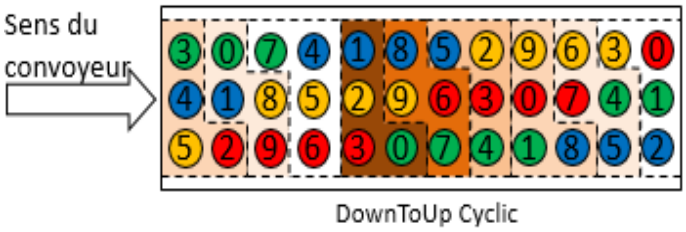

FIGURE 4.7 – Exemple d'une stratégie de collaboration pour des produits en ligne.

Une stratégie de collaboration peut aussi être appliquée pour la dépose des produits dans les cartons. La stratégie *DownToUp Cyclic* peut être utilisée, mais ce sont les places dans les cartons qui sont attribuées aux robots. En poursuivant avec notre exemple : le robot 1 dépose les produits dans les places rouges, le robot 2 dans les places vertes, le robot 3 dans les places bleus et le robot 4 dans les places jaunes, voir Figure 4.8.

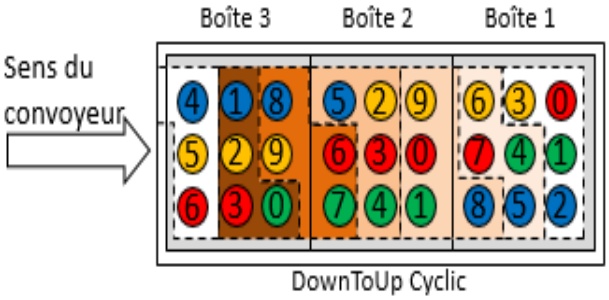

FIGURE 4.8 – Exemple d'une stratégie de collaboration pour des places.

Un autre type de stratégie de dépose est la gestion du remplissage des boîtes. Ces stratégies permettent de donner un seuil de remplissage des boîtes pour chaque robot. Un exemple de ce type de stratégies est le remplissage *Linéaire*, voir Figure 4.9. Pour une application avec quatre robots, le robot 1 remplit les boîtes jusqu'à 25%, le robot 2 jusqu'à 50%, le robot 3 jusqu'à 75% et le dernier robot remplit jusqu'au maximum. Un autre exemple de stratégie, reposant sur les seuils de remplissage, est le remplissage *Régressif*. Le robot 1 remplit les boîtes jusqu'à 40%, le robot 2 jusqu'à 70%, le robot 3 jusqu'à 90% et le dernier robot remplit jusqu'au maximum. D'autres seuils peuvent bien évidemment être proposés.

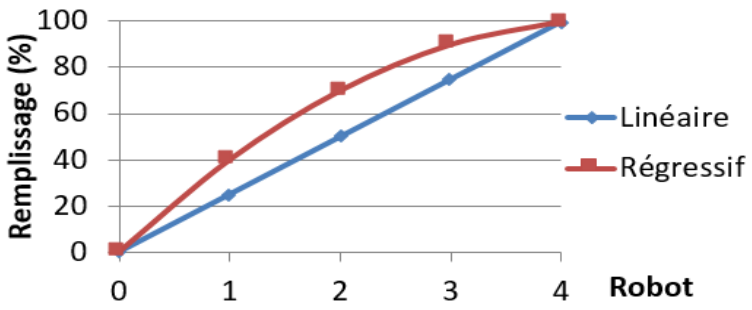

FIGURE 4.9 – Exemple de remplissage linéaire et régressif.

4.3 Produits orientés

L'orientation des produits peut influencer fortement les performances globales d'une application pick & place. En effet, si elle n'est pas prise en compte dans les algorithmes de pick & place, les robots peuvent "perdre" du temps avant de prendre un produit. En général, l'effecteur des robots, s'oriente en même temps qu'il se déplace. Par exemple si un robot doit prendre plusieurs produits les uns à la suite des autres en utilisant la stratégie *IntToExt Cyclic* avec la règle *SPT*, lorsque le robot va prendre le deuxième produit, il va choisir celui qui est le plus proche en distance du premier produit, donc en temps de translation. Cependant ce produit peut ne pas être le plus proche si on prend en compte la durée de rotation de l'effecteur. Un autre produit qui peut être plus éloigné en distance du premier produit, peut cependant être pris plus rapidement car le temps de rotation que doit effectuer le robot est plus court. Un exemple est illustré avec la Figure 4.10 où le produit 1 est le premier produit pris, le produit 2 est le produit le plus proche en distance, mais une grande rotation doit être effectuée et le produit 3 est un produit légèrement plus éloigné que le produit 2, mais le robot ne devra effectuer qu'une légère rotation.

Il est donc important de prendre en compte aussi bien le temps de translation et le temps de rotation dans les algorithmes afin de ne pas dégrader les performances de l'application. De plus, l'impact de l'orientation des produits est plus important lorsqu'un robot effectue de la prise multiple que lorsqu'un robot effectue de la prise simple. En effet, un robot se déplace sur moins de distance entre produits qu'entre produits et boîtes. Ce qui entraîne donc des temps de déplacement moyens plus réduits. Le robot aura donc moins de temps pour effectuer sa rotation.

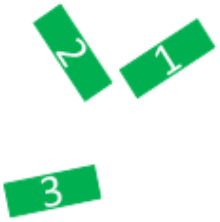

FIGURE 4.10 – Exemple d'orientation des produits.

Les algorithmes prenant en compte la rotation reprennent les algorithmes utilisés avec des produits simples. Plus précisément, ce sont les algorithmes utilisant la règle *SPT* qui vont être modifiés. Pour chaque algorithme, en plus de calculer le temps de translation pour aller prendre un produit et donc prendre le plus proche au niveau distance, le temps de rotation est aussi calculé. Ensuite, l'algorithme va garder le temps le plus grand entre les deux. Puis, il va choisir le produit avec le temps le plus petit. Dans un premier temps, cette modification peut s'effectuer avec la règle de planification individuelle *SPT*. Dans un second temps, la modification peut être apportée à toutes les stratégies de collaboration pouvant utiliser la règle *SPT* : *DownToUp*, *IntToExt*, *IntToExt Cyclic*, *IntToExt Alt* et *DownToUp Cyclic*. Un exemple comparant la stratégie *DownToUp* utilisant la règle *STP* avec ou non la prise en compte de l'orientation est donné avec la Figure 4.11 pour des robots prenant quatre produits les uns à la suite des autres. On remarque bien que l'assignation est totalement différente.

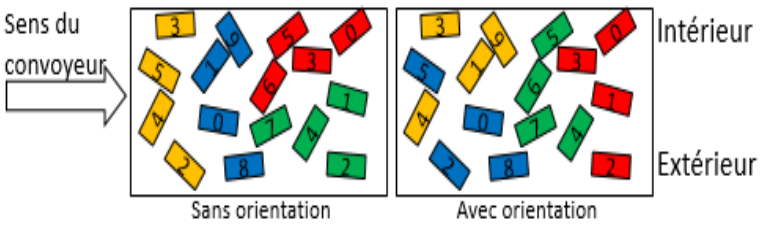

FIGURE 4.11 – Comparaison de la stratégie *DownToUp* prenant en compte ou non de l'orientation.

4.4 Produits avec poids

Le poids des produits a un impact important sur le temps de cycle moyen des robots. Plus les produits seront lourds, plus le temps de cycle moyen sera grand, ce qui entraînera une réduction de la cadence. Cependant, dans une chaîne de production pour l'emballage, le poids des produits ne varie pas assez pour avoir un impact sur le temps de cycle moyen au cours de la production. Pour ce genre d'application, le poids est à prendre en compte au niveau du remplissage des boîtes. C'est le poids actuel et le poids souhaité des boîtes qui sont importants. Les boîtes sortant de la chaîne de production doivent avoir un poids le plus proche du poids souhaité. Pour cela, les robots doivent choisir les bons produits et les déposer dans les bonnes boîtes. Le nombre de produits à l'intérieur d'une boîte peut être fixe ou non.

Avant de développer des algorithmes qui donnent aux robots les bons produits à prendre, on peut se pencher sur la question de la configuration de l'application. Est-ce que la configuration d'une application peut permettre aux robots d'avoir plus de choix de produits et donc la possibilité d'avoir de meilleurs produits ? Le sens des convoyeurs a une grande influence sur le choix des produits. En effet, il y a beaucoup plus de produits (ou de places) en amont d'un convoyeur qu'en aval, voir Figure 4.12. Pour des convoyeurs en co-courant, il y a beaucoup de produits pour beaucoup de places, mais peu de produits pour peu de places. Le dernier robot en aval des convoyeurs a donc très peu de choix pour finir de remplir les boîtes, les boîtes ont donc du mal à atteindre au plus juste le poids souhaité. Alors que pour des convoyeurs en contre-courant, d'un côté il y a beaucoup de produits pour peu de places et de l'autre il y a beaucoup de places pour peu de produits. Cela permet donc aux robots en aval du convoyeur de sortie d'avoir le maximum de possibilités pour remplir correctement les boîtes et aux robots en amont de trouver obligatoirement une place pour leur produit.

La configuration de l'application étant fixée, les algorithmes peuvent être conçus. Seules des règles de planification individuelles ont été développées. L'utilisation de stratégies de collaboration peut poser des problèmes si un produit initialement assigné à un robot ne peut pas être pris. Trois règles sont proposées :

— *WeightMean* : X produits sont trouvés. Le poids moyen de l'ensemble doit être le plus proche du poids moyen d'un produit. Les boîtes sont remplies par palier de X produits. Par exemple, si le poids moyen d'un produit est de 10g, le robot prend en premier un produit qui pèse 9.5g, il va chercher un second produit le plus proche de 10.5g. Cette règle doit être utilisée avec la règle *XPlaces* pour que les robots remplissent bien les mêmes boîtes.

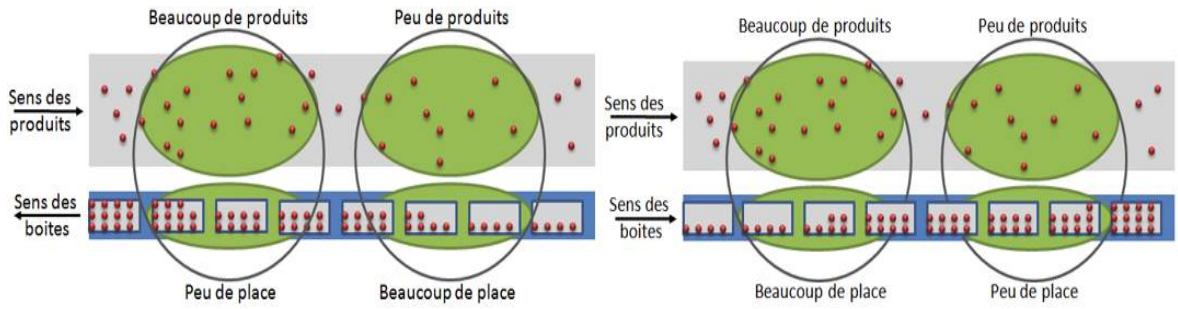

FIGURE 4.12 – Densité de produits et de place le long d'un convoyeur.

— *WeightAdjust* : La règle essaie d'avoir un remplissage des boîtes le plus linéaire par rapport au poids. Par exemple, si le poids moyen d'un produit est de 10g, la boîte contient déjà deux produits de 9.8g et 10.5g, il va chercher un produit le plus proche de (Nombre de produits dans la boîte +1)*Poids moyen d'un produit - Poids actuel de la boîte= $(2+1)*10-(9.8+10.5)=9.7$g. Si le plus proche produits pèse 9.8g le suivant sera donc le plus proche de $(3+1)*10-(9.8+10.5+9.8)=9.9$g. Cette règle doit être utilisée avec la règle *TheBox* car la boîte est déjà identifiée.
— *WeightAdjustLast* : Les robots remplissent les boîtes avec une règle de planification individuelle classique (*FIFO*, *LIFO*, etc.) jusqu'à l'avant dernier produit. Pour le dernier ils prennent le produit avec un poids qui permet de remplir au mieux la boîte. Cette règle doit être utilisée avec la règle *TheBox* car la boîte est déjà identifiée.

4.5 Produits avec types

Les produits avec types sont des produits qui ont des caractéristiques qui peuvent être exprimées avec un entier (forme, couleur, parfum etc.). La problématique que posent les produits avec types est identique à celle des produits avec poids. Les robots doivent choisir les bons produits pour remplir correctement les boîtes. Comme pour le poids, des convoyeurs en contre-courant permettent d'augmenter le nombre de choix en termes de places, pour un produit et en termes de produits, pour une place. Seules des règles de planification individuelles ont été développées :

— *SearchProductType* : Le robot prend le produit correspondant au type de la place où le robot va le déposer.
— *SearchPlaceType* : Le robot dépose un produit à la place correspondant au type du produit.

Comme vu avec la Figure 4.12, les robots en aval du convoyeur de sortie ont beaucoup de produits pour peu de places, la règle *SearchProductType* sera donc privilégiée pour ces robots. Tandis que les robots en amont du convoyeur de sortie ont beaucoup de places pour peu de produits, la règle *SearchPlaceType* sera donc privilégiée pour ces derniers.

4.6 Algorithmes liés aux convoyeurs

Le bloc de stratégies de collaboration permet, en plus d'assigner les produits aux robots ou de gérer le taux de remplissage des boîtes, de gérer dynamiquement la vitesse des convoyeurs. Une des stratégies utilisées durant la thèse est la stratégie *PlacingPriority*. Elle autorise le convoyeur de sortie à s'arrêter lorsqu'une boîte non remplie atteint la deuxième partie de la zone de travail du dernier robot. Cela permet au dernier robot de finir de remplir la dernière boîte. Avec cette règle, toutes les boîtes seront remplies. Cependant le convoyeur de sortie doit pouvoir être arrêté durant la production. On peut établir le même type de stratégie pour les produits. La stratégie *PickingPriority* autorise le convoyeur d'entrée à s'arrêter lorsqu'un produit atteint la deuxième partie de la zone de travail du dernier robot. Cela permet au dernier robot de prendre ce dernier produit. Avec cette règle, tous les produits sont pris. Cependant le convoyeur d'entrée doit pouvoir être arrêté durant la production, ce qui peut ne pas être le cas lorsque celui-ci est précédé d'un four par exemple.

4.7 Conclusions sur les algorithmes développés

Dans ce chapitre, nous avons détaillé les niveaux de stratégies des architectures de simulation et du contrôleur vues dans les sections 3.4 et 3.5. Ces niveaux sont contenus dans deux types de blocs fonctionnels. Le premier bloc permet de gérer les règles de planification individuelles. Chaque robot doit être affilié à son bloc. Ce bloc permet de fournir la position ou l'identifiant du produit ou de la place à atteindre par le robot associé. Le deuxième bloc permet de gérer principalement les stratégies de collaboration entre les robots. Les avantages de ces blocs sont la possibilité de contenir tous les algorithmes souhaités et de pouvoir changer l'algorithme utilisé en ligne en modifiant simplement l'identifiant de l'algorithme. Ces blocs ont été développés afin d'avoir une mise en place rapide et de permettre l'ajout ou la suppression d'algorithmes simplement.

De plus, nous avons présenté les différents algorithmes développés durant cette thèse. Cela correspond à l'étape *Développement Algorithmes* du cycle de développement de la Figure 4.1. Certains algorithmes, dont les règles de planification individuelles, sont inspirés de travaux réalisés dans la littérature. Les stratégies de collaboration ont été pensées de telle sorte à s'approcher du seuil de performance d'une application pick & place avec convoyeurs parallèles. Pour cela, elles assignent les produits aux robots avant qu'ils n'arrivent dans la zone de travail du premier robot. Selon les caractéristiques des produits utilisés dans l'application (simples, avec orientation, avec poids etc.), différents algorithmes ont été développés. Un exemple de programmation de règles individuelles et de stratégie de collaboration est donné dans l'annexe A.

L'architecture pick & place étant mise en place, les blocs fonctionnels gérant les algorithmes, les règles de planification individuelles et les stratégies de collaboration étant développés, le prochain chapitre se consacrera à l'étape *Tests Algorithmes* du cycle de développement d'une application pick & place de la Figure 4.1.

Chapitre 5

Étude comparative en simulation

Sommaire

Nous avons vu jusqu'à maintenant la méthodologie pour mettre en place une application pick & place multi-robots en utilisant un nouvel outil logiciel : $SOSi^{TM}$. Ceci correspond aux étapes *Développement Robot et Environnement* et *Tests Comportementaux* du cycle de développement de la Figure 3.1. Nous avons aussi détaillé plusieurs algorithmes permettant à chaque robot de choisir les produits à prendre et les boîtes où déposer les produits et permettant d'avoir un travail collaboratif entre les robots tout en prenant en compte les caractéristiques des produits, étape *Développement Algorithmes* du cycle de développement. Maintenant, nous allons présenter l'étape *Tests Algorithmes* de la Figure 5.1 afin de tester les différents algorithmes développés dans le chapitre 4 suivant différentes configurations d'applications pick & place toujours en utilisant l'outil logiciel $SOSi^{TM}$.

Ce chapitre est un récapitulatif détaillé de tous les tests effectués en simulation. Le nombre de configurations principales (et donc de tableaux associés de résultats) d'une application pick & place multi-robot étant important, ce chapitre est volumineux et peut être difficile à lire pour une personne ne cherchant pas une information dans ce domaine. Cependant, ce chapitre est un point important des travaux de cette thèse car il présente l'étendue du travail effectué en simulation (environ 52 heures en temps réel équivalent à 312h en temps simulé) ainsi que l'expertise que ce travail m'a permis de développer. Tous ces tests ont dû être effectués car les différents cahiers des charges des simulations proposés sont présents actuellement en industrie. Pour le lecteur qui ne souhaite pas s'attarder sur les chiffres de toutes les simulations, un résumé des résultats sera présenté dans la conclusion de ce chapitre.

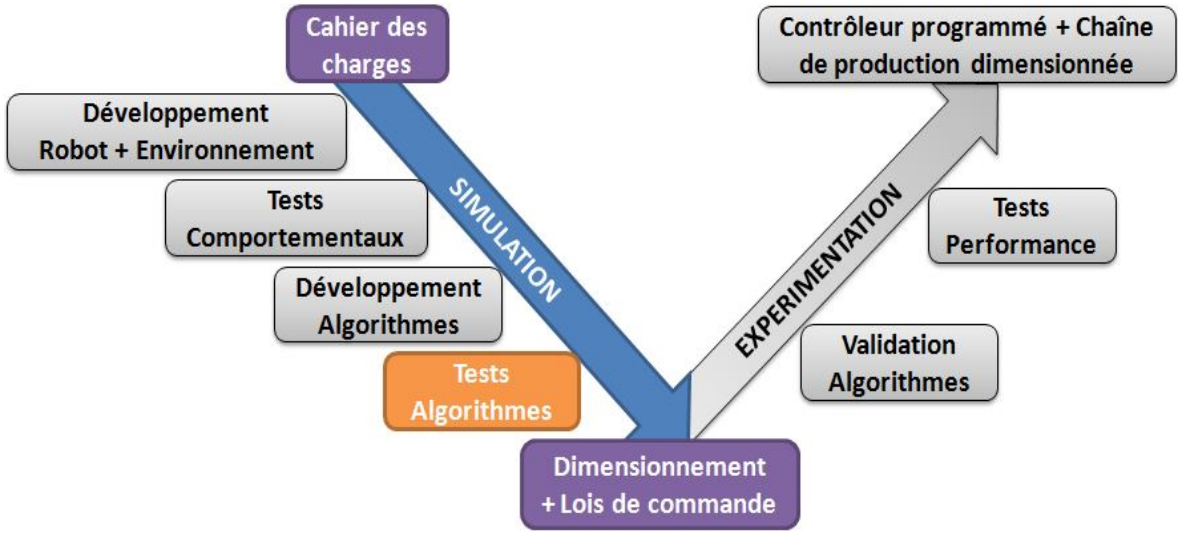

FIGURE 5.1 – Étape *Tests Algorithmes* du cycle de développement d'une application pick & place.

Avant de commencer l'étude comparative, une explication du dimensionnement réalisé dans l'industrie est nécessaire. Ce dimensionnement sera utilisé tout au long de l'étude.

5.1 Explication du dimensionnement industriel actuel

Comme indiqué dans le chapitre 1, le dimensionnement des systèmes robotisés pick & place est fait habituellement de façon empirique. Cependant, cette démarche empirique suit généralement une séquence de décision, qui peut être résumée de la façon suivante :

1. Un débit de produits en entrée est donné par le client, en termes de produits par seconde, noté FPs.
2. L'utilisateur connaît généralement la durée moyenne d'un robot pour effectuer une tâche de pick & place. Cette information donne une estimation de la cadence en produits par seconde, $PPas$.
3. Le système est surdimensionné utilisant deux possibilités :
   — Soit un ou deux robots supplémentaires sont ajoutés au nombre minimum de robots. Le nombre de robots minimum $NRmin$ est calculé à l'aide de la partie entière par excès :
   $$NRmin = \lceil FPs/PPas \rceil$$
   Le nombre final de robots est $NRf = NRmin + X$ où X peut être 1 ou 2 (X dépend de la taille de l'application).
   — Soit une marge de sécurité Mfp est ajoutée au débit de produits (10-15%). Le nombre de robots minimum $NRmin$ est calculé à l'aide de la partie entière par excès :
   $$NRmin = \lceil FPs * (1 + Mfp)/PPas \rceil$$
   Le nombre final de robots est $NRf = NRmin$.

4. Le débit de boîtes en sortie FBs est calculé en fonction du nombre de produits NPb dans une boîte : $FBs = FPs/NPb$.
5. Les vitesses des convoyeurs d'entrée Vi et de sortie Vo sont soit fixées par le client soit fixées arbitrairement, cependant elles dépendent du débit et de l'espacement entre le milieu des produits EP ou des boîtes EB : $FPs = Vi/EP$ et $FBs = Vo/EB$. Il n'y a pas de gestion dynamique de la vitesse des convoyeurs.
6. Il n'y a pas de stratégies de collaboration entre les robots.
7. Les règles de planification individuelles sont les mêmes pour tous les robots et consistent à prendre le maximum de pièces qui arrivent.

Les étapes 1 à 5 de cette démarche permettent de dimensionner les applications. Ces étapes seront reprises à chaque calcul de dimensionnement lors de l'étude comparative. Les étapes 6 et 7 indiquent qu'il n'y a aucune stratégie de commande utilisée. L'usage montre que cette absence de dimensionnement peut être améliorée. En effet, les premiers robots sont utilisés beaucoup plus que les derniers robots, ce qui peut entraîner leur usure prématurée. De plus, des produits peuvent être perdus car non pris par les robots. C'est sur ce point que les algorithmes développés dans le chapitre 4 et comparés dans ce chapitre vont essayer d'apporter une amélioration.

5.2 Cahier des charges

Une étude comparative a été effectuée en utilisant le nouvel outil logiciel présenté dans la section 3.3, $SOSi^{TM}$. Cette étude compare différentes stratégies de collaboration entre les robots ainsi que différentes règles de planification individuelles pour chaque robot présentées dans le chapitre 4.

Dans la section 5.1, le dimensionnement industriel d'une application pick & place a été expliqué. Un des principes est de surdimensionner le système à l'aide d'une marge de sécurité ajoutée au débit d'entrée. L'étude, en plus de comparer les différents algorithmes, compare l'impact que peut avoir la marge de sécurité sur les performances et le comportement d'une application. Les marges utilisées varient de telle sorte à avoir un dimensionnement du plus juste (marge de sécurité inférieure à 2.5%) à un fort surdimensionnement (marge de sécurité supérieure à 15%). En pratique, la marge de sécurité ne doit pas être égale à 0% car il faut prendre en compte le temps de calcul pour trouver un produit ou une place (si petit soit-il). Si une marge de sécurité de 0% est appliquée, le système perdra quelques produits ou quelques boîtes ne seront pas remplies, ce qui est contraire à l'objectif. Cependant, un essai avec un système largement sous-dimensionné est effectué, pour chaque configuration, afin de tester les performances maximales et les limites des stratégies de collaboration.

Tout au long de l'étude, différents paramètres sont analysés et comparés comme le pourcentage de produits perdus, le pourcentage de boîtes non remplies, l'équilibre de la charge de travail des robots, la charge de travail totale du système et la moyenne du temps de déplacement pour que les robots effectuent un aller-retour pick & place.

— La charge de travail est définie par l'équation (5.1) :

$$W = \frac{T_{Pick} + T_{Place}}{T_{Pick} + T_{Place} + T_{Wait}} * 100 \tag{5.1}$$

où T_{Pick}, T_{Place} et T_{Wait} sont respectivement le temps de prise, de dépose et d'attente en seconde.

— L'équilibre de la charge de travail est défini par la différence de la charge de travail des robots et est calculé avec l'équation (5.2) :

$$BW = \sum_{k=1}^{NR_f - 1} |W_{k+1} - W_k| \tag{5.2}$$

où W_k est la charge de travail du $k^{ième}$ robot. Plus BW est proche de 0, plus le système est équilibré.

— La charge de travail totale du système est définie par la somme de la charge de travail de tous les robots et est calculée avec l'équation (5.3) :

$$TW = \sum_{k=1}^{NR_f} W_k \tag{5.3}$$

où W_k est la charge de travail du $k^{ième}$ robot. L'index TW donne une idée du taux de fonctionnement du système entier. Plus l'index est grand et plus le système est sollicité.

— La moyenne du temps de déplacement des robots est définie par la somme des temps de prise et de dépose moyens divisée par le nombre de robots. Cette grandeur est calculée par l'équation suivante (5.4) :

$$APP = \frac{\sum_{k=1}^{NR_f} T_{Pick_k} + T_{Place_k}}{NR_f} \tag{5.4}$$

où T_{Pick_k} et T_{Place_k} sont respectivement le temps de prise et de dépose en seconde du $k^{ième}$ robot. Plus l'indicateur APP est faible, plus le système est capable de prendre de produits.

Après discussion avec Schneider Electric, le cahier des charges pour les tests des applications pick & place est le suivant :

— L'application est composée de quatre robots tous identiques.
— Les robots travaillent avec une vitesse de 10 m/s et une accélération de 100 m/s^2.
— Les robots prennent et déposent un seul produit à la fois.
— Les robots ont une zone de travail de 600 mm et décalée de 200 mm par rapport à leur centre dans le sens du convoyeur de sortie.
— Un cycle de pick & place pour un robot prend en moyenne 0.383 secondes. Le robot peut prendre $PPas = 1/0.383 = 2.61$ produits par seconde.
— Les convoyeurs sont positionnés en parallèle.
— Les convoyeurs sont en co-courant.
— Les produits dans le flux d'entrée sont positionnés aléatoirement.
— Les boîtes mesurent 0.1 m * 0.1 m et contiennent 9 produits ($NPb = 9$).
— La vitesse du convoyeur d'entrée est choisie arbitrairement égale à 0.2 m/s ($Vi = 0.2$).
— La distance entre le centre des boîtes est choisie arbitrairement égale à 0.15 m ($EB = 0.15$).

— La stratégie *PlacingPriority* est appliquée. Elle permet de ne pas avoir de boîtes non remplies à la sortie de l'application. Le pourcentage de boîtes non remplies sera donc absent des tableaux.
— Le temps de simulation est de 1h.
— L'étude est effectuée en régime permanent.

Pour un système surdimensionné, selon la méthode décrite dans la section 5.1, un débit de produits est choisi $FPs = 8.35$ produits par seconde. Pour surdimensionner le système, une marge de sécurité $Mfp = 20\%$ est ajoutée. Le nombre minimum de robots est $NRmin = FPs * (1 + Mfp)/PPas = 8.35 * (1 + 0.2)/2.61 = 3.84$ arrondi à 4. Comme une marge de sécurité a été utilisée, le nombre final de robots est $NRf = NRmin = 4$. En s'appuyant sur le nombre de produits dans une boîte, le débit de boîtes est égal à $FBs = FPs/NPb = 8.35/9 = 0.93$ boîtes par seconde. En utilisant la distance entre les boîtes, la vitesse du convoyeur de sortie doit être égale à $Vo = FBs * EB = 0.93 * 0.15 = 0.14$ m/s.

Pour un système dimensionné au plus juste, selon la méthode décrite dans la section 5.1, un débit de produits est choisi $FPs = 10.23$ produits par seconde. Pour surdimensionner le système, une marge de sécurité $Mfp = 2\%$ est ajoutée. Le nombre minimum de robots est $NRmin = FPs * (1 + Mfp)/PPas = 10.23 * (1 + 0.02)/2.61 = 4$. Comme une marge de sécurité a été utilisée, le nombre final de robots est $NRf = NRmin = 4$. En s'appuyant sur le nombre de produits dans une boîte, le débit de boîtes est égal à $FBs = FPs/NPb = 10.23/9 = 1.14$ boîtes par seconde. En utilisant la distance entre les boîtes, la vitesse du convoyeur de sortie doit être égale à $Vo = FBs * EB = 1.14 * 0.15 = 0.17$ m/s.

Pour un système sous-dimensionné, selon la méthode décrite dans la section 5.1, un débit de produits est choisi $FPs = 13.05$ produits par seconde. Pour sous-dimensionner le système, une marge de sécurité négative est ajoutée de telle sorte que l'arrondi ne rend pas la marge positive, $Mfp = -25\%$. Le nombre minimum de robots est $NRmin = FPs * (1 + Mfp)/PPas = 13.05 * (1 - 0.25)/2.61 = 3.75$ arrondi à 4. Comme une marge de sécurité a été utilisée, le nombre final de robots est $NRf = NRmin = 4$. En s'appuyant sur le nombre de produits dans une boîte, le débit de boîtes est égal à $FBs = FPs/NPb = 13.05/9 = 1.45$ boîtes par seconde. En utilisant la distance entre les boîtes, la vitesse du convoyeur de sortie doit être égale à $Vo = FBs * EB = 1.45 * 0.15 = 0.22$ m/s.

L'étude est composée de plusieurs parties. Dans un premier temps, l'influence des règles de planification individuelles est analysée. Il n'y a aucune stratégie de collaboration entre les robots utilisée afin de se focaliser uniquement sur les règles individuelles. Cette partie permet de choisir une bonne combinaison de règles à appliquer à chaque robot. Dans un second temps, l'influence des stratégies de collaboration entre les robots est étudiée. Cette partie est elle-même divisée en plusieurs parties correspondant à différentes configurations d'une application pick & place (sens convoyeurs, type de produits etc.). Cette partie permet de choisir une bonne stratégie suivant la configuration du système.

5.3 Influence des règles de planification individuelles

Dans cette section de l'étude, seules les règles de planification individuelles seront comparées. Les règles étudiées sont la règle *FIFO*, la règle *LIFO* et la règle *SPT*. Il n'y a aucune stratégie de collaboration utilisée. Le système utilisé est dimensionné au plus juste, c'est-à-dire qu'une petite marge de sécurité est ajoutée au débit d'entrée. La simulation utilise le cahier des charges initial, cependant le temps de simulation est réduit à 10 min pour chacune des possibilités.

Afin de limiter le nombre de possibilités de combinaisons de règles de planification individuelles, les logiques de prise et de dépose pour un robot sont identiques. Un robot ne pourra pas utiliser la règle *FIFO* pour la prise et la règle *LIFO* pour la dépose par exemple. S'il n'y a pas cette limite le nombre de possibilités s'élève à $3 \wedge 2 \wedge 4 = 6561$ contre $3 \wedge 4 = 81$ possibilités avec la limitation. Cette limitation est possible car le système étudié a une configuration de convoyeur en co-courant. En effet, si un robot utilise la règle *LIFO* pour la prise et la règle *FIFO* pour la dépose, ou inversement, la distance de déplacement entre la prise et la dépose est beaucoup plus importante que si le robot utilisait la règle *FIFO* ou *LIFO* pour la prise et la dépose, comportement qui n'est pas recherché. Un exemple est illustré avec la Figure 5.2, où la règle de dépose est *FIFO*.

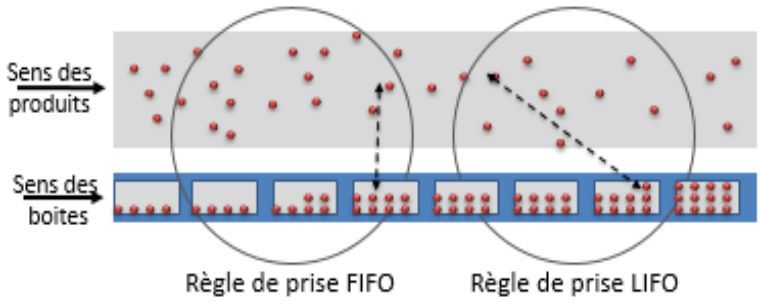

FIGURE 5.2 – Déplacement de l'effecteur des robots suivant la logique de prise pour des convoyeurs en co-courant.

Le Tableau 5.1 présente les résultats de simulation de l'effet des règles de planification individuelles de chaque robot sur un système dimensionné au plus juste. Par souci de simplicité, les règles sont désignées : 0 pour la règle *FIFO*, 1 pour la règle *LIFO* et 2 pour la règle *SPT*. Bien que toutes les possibilités aient été simulées, ce tableau ne recueille que les combinaisons qui permettent d'obtenir moins de 1% de produits perdus. Premièrement, nous notons que la pire règle est la règle *SPT* dans toutes les combinaisons. Le taux de perte est plus grand que 1%, ou très proche de cette valeur. Ceci peut être expliqué par le fait que le temps de prise et de dépose augmente du premier robot au dernier, voir Figure 5.3. Il est à noter que la règle *FIFO* doit être utilisée par le dernier robot. Lorsque plusieurs produits arrivent dans l'espace de travail du dernier robot, si la règle de planification individuelle de ce robot est la règle *LIFO* ou la règle *SPT*, le dernier ou le plus proche produit est pris alors que les autres produits continuent. Si l'avant-dernier robot utilise une règle de *LIFO* il y a un peu de pertes. Pour les autres robots, le choix entre les règles *FIFO* et *LIFO* n'a pas un impact significatif sur les résultats. Pour une facilité et une rapidité de mise en application, il est donc préférable d'utiliser la règle *FIFO* pour tous les robots.

R1	R2	R3	R4	Produits perdus (%)
0	0	0	0	0.00
0	0	1	0	0.02
0	0	2	0	0.15
0	1	0	0	0.00
0	1	1	0	0.05
0	1	2	0	0.23
0	2	0	0	0.87
1	0	0	0	0.00
1	0	1	0	0.00
1	0	2	0	0.38
1	1	0	0	0.00
1	1	1	0	0.01
1	2	0	0	0.92

TABLE 5.1 – Résultats des règles de planification individuelles avec moins de 1% de pertes de produits.

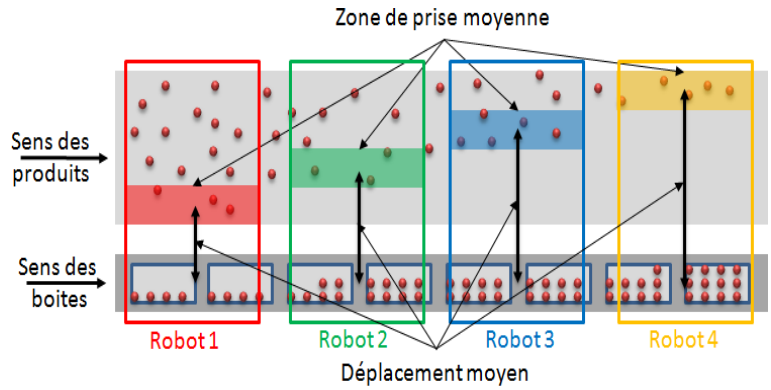

FIGURE 5.3 – Temps moyen de prise et de dépose avec la règle *SPT*.

Comme indiqué avant ces tests, ce résultat n'est applicable que pour un système avec une configuration de convoyeurs en co-courant. En effet, pour une application avec des convoyeurs en contre-courant, si un robot utilise la règle *FIFO* pour la prise et la dépose, ou inversement, la distance de déplacement entre la prise et la dépose est beaucoup plus importante que si le robot utilisait la règle *LIFO* pour la prise et la règle *FIFO* pour la dépose, voir Figure 5.4 où la règle de dépose est *FIFO*.

Dans le cas d'un système avec des convoyeurs en contre-courant, les robots doivent être séparés en deux catégories : les robots qui doivent finir de remplir les boîtes et les robots qui doivent prendre tous les produits. En suivant la logique des résultats obtenus, les robots finissant de remplir les boîtes doivent utiliser la règle *FIFO* pour la dépose et donc la règle *LIFO* pour la prise. Inversement, les robots finissant de prendre les produits doivent utiliser la règle *FIFO* pour la prise et donc la règle *LIFO* pour la dépose, voir Figure 5.5.

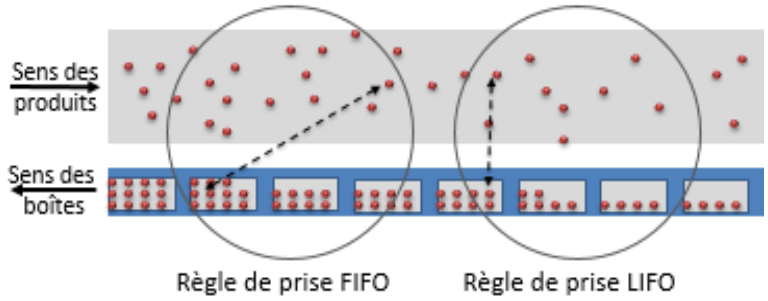

FIGURE 5.4 – Déplacement de l'effecteur des robots suivant la logique de prise pour des convoyeurs en contre-courant.

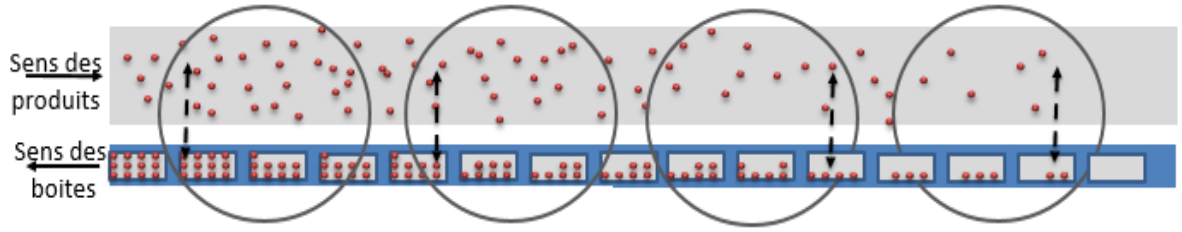

FIGURE 5.5 – Exemple de combinaison de règles de planification individuelles pour un système avec des convoyeurs en contre-courant.

5.4 Influence des stratégies de collaboration

Dans cette section, l'étude compare différentes stratégies de collaboration entre les robots. Au vu des résultats de la section 5.3 les règles de planification utilisées pendant l'étude sont la règle *FIFO* pour la prise et la dépose pour tous les robots si les convoyeurs sont en co-courant. Si les convoyeurs sont en contre-courant, les robots finissant de remplir les boîtes utilisent la règle *FIFO* pour la dépose et donc la règle *LIFO* pour la prise, tandis que les robots finissant de prendre les produits utilisent la règle *FIFO* pour la prise et la règle *LIFO* pour la dépose. L'étude de l'influence des stratégies de collaboration est divisée en plusieurs parties suivant les caractéristiques croisées de l'application suivante :

— Flux de produits positionnés aléatoirement ou en ligne.
— Convoyeur co-courant ou contre-courant.
— Prise simple ou multiple.
— Système surdimensionné, dimensionné au plus juste ou sous-dimensionné.
— Produits simples, orientés, avec poids ou avec type.

En premier lieu, l'étude se concentrera sur les produits simples (produits sans caractéristiques spécifiques). Durant cette partie, toutes les configurations principales d'applications pick & place seront analysées. En deuxième lieu, l'étude se portera sur les produits avec caractéristiques (orientés, avec poids et avec type).

Dans le reste de l'étude, la stratégie de collaboration *Nothing* signifie qu'il n'y pas de stratégie de collaboration utilisée.

5.4.1 Produits simples

Flux de produits simples positionnés aléatoirement avec convoyeurs en co-courant et prise simple

Pour cette partie, le cahier des charges initial est utilisé. Dans un premier temps le système est sous-dimensionné. Pour ce test, la stratégie *PlacingPriority* n'est pas utilisée afin de limiter les contraintes sur le système. Le nombre de boîtes non remplies est donc présent. Le système étant sous-dimensionné, tous les robots fonctionnent à 100%, l'équilibre de la charge de travail n'a pas lieu d'être étudiée. Lors de la comparaison entre les stratégies de prise, aucune stratégie de dépose n'est utilisée. De même, lors de la comparaison entre les stratégies de dépose, aucune stratégie de prise n'est utilisée.

Stratégie de prise	**Produits perdus (%)**	**Boîtes non remplies (%)**	**TW (%)**	**APP (ms)**
Nothing	19.9	99.8	392	377
DownToUp	19.9	99.8	392	376
DownToUp Cyclic	19.9	99.7	392	377
IntToExt	19.6	99.5	392	377
IntToExt Cyclic	20.0	99.7	392	377
IntToExt Alt	20.0	99.7	392	377
Horizontal	20.3	99.6	391	380
Vertical	19.9	99.8	392	378
Stratégie de dépose				
Linear	19.9	99.8	392	377
DownToUp Cyclic	19.9	94.9	392	376

TABLE 5.2 – Résultats des stratégies de collaboration pour un système sous-dimensionné ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple

Le Tableau 5.2 présente les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système sous-dimensionné ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple. Nous remarquons que lorsque le système est sous-dimensionné, toutes les stratégies de collaboration obtiennent des résultats très similaires. Les stratégies ne peuvent donc pas être comparées sur ce genre de système. On peut cependant noter que la durée moyenne d'un aller-retour est inférieure au temps de cycle de pick & place (APP = 377 ms en moyenne pour 383 ms). Cela peut être dû au fait que les robots ont pris les produits les plus proches de l'intérieur du convoyeur et ont laissé ceux à l'extérieur.

Dans un second temps, le système étudié est surdimensionné. Les résultats des Tableaux 5.3, 5.4 et 5.5 sont obtenus respectivement avec les stratégies de dépose *Nothing*, *Linear* et *DownToUp Cyclic*.

Les Tableaux 5.3, 5.4 et 5.5 présentent les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système surdimensionné ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple. Tout d'abord, nous pouvons noter que la pire stratégie pour la prise est *Horizontal*

Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
Nothing	**0.90**	**80.4**	311	378
DownToUp	0.00	7.77	347	417
DownToUp Cyclic	0.00	9.20	344	413
IntToExt	0.00	5.92	346	415
IntToExt Cyclic	0.00	9.64	344	413
IntToExt Alt	**0.08**	8.65	338	406
Horizontal	**3.09**	23.1	326	405
Vertical	0.00	26.5	331	399

TABLE 5.3 – Résultats des stratégies de collaboration pour un système surdimensionné ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple n'utilisant pas de stratégie de dépose.

Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
Nothing	0.00	25.3	337	407
DownToUp	0.00	7.10	348	417
DownToUp Cyclic	0.00	8.99	344	412
IntToExt	0.00	3.94	345	414
IntToExt Cyclic	0.00	9.43	343	412
IntToExt Alt	0.03	8.68	336	403
Horizontal	**1.69**	7.31	329	402
Vertical	0.00	24.1	325	391

TABLE 5.4 – Résultats des stratégies de collaboration pour un système surdimensionné ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple utilisant la stratégie de dépose *Linear*.

pour toutes les stratégies de dépose. Lorsque le système utilise cette stratégie, il y a beaucoup de pertes de produits (de 1.41% à 3.09%). Le temps de prise et de dépose augmentent du premier robot au dernier, voir Figure 5.3. Lorsqu'il n'y a aucune stratégie de prise ou de dépose, le système perd des produits (0.90%). De plus, le système est très déséquilibré (BW = 80.4%), ce qui peut entraîner une usure prématurée de certains robots. La perte de produits est due à ce déséquilibre. En effet, les premiers robots travaillent à leur maximum tandis que le dernier ne travaille presque pas.

Les résultats de simulation montrent également que l'utilisation de la stratégie *PlacingPriority* arrête le convoyeur de sortie afin de remplir toutes les boîtes. Lorsque le système fonctionne, les premiers robots remplissent presque toutes les boîtes mais laissent passer quelques produits (car le système n'est pas assez surdimensionné pour fonctionner avec seulement trois robots). Ces produits sont donc pris par le dernier robot qui finit de remplir les quelques boîtes restantes. Cependant, il arrive à certains moments que le dernier robot n'ait pas de produits pour remplir une boîte. Le convoyeur s'arrête afin d'attendre des produits. Pendant ce temps-là, les autres robots continuent de remplir les boîtes. Une fois que le dernier robot a fini de remplir la boîte, le convoyeur de sortie redémarre. Néanmoins pendant un moment, il n'y aura plus de boîtes à remplir, car elles ont été remplies lors de l'arrêt du convoyeur et donc les produits ne pourront pas être pris.

Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
Nothing	0.00	6.68	349	419
DownToUp	0.00	11.7	332	399
DownToUp Cyclic	0.01	13.1	330	396
IntToExt	0.00	6.97	333	399
IntToExt Cyclic	0.00	7.76	332	398
IntToExt Alt	0.00	1.46	321	385
Horizontal	**1.41**	16.6	324	394
Vertical	0.00	5.94	340	408

TABLE 5.5 – Résultats des stratégies de collaboration pour un système surdimensionné ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple utilisant la stratégie de dépose *DownToUp Cyclic*.

L'utilisation d'une stratégie de collaboration de prise permet d'éviter ce comportement. Elle permet de prendre tous les produits ou de n'en perdre que très peu (moins de 0.08% sauf pour la stratégie *Horizontal*) ce qui garantit une charge de travail mieux équilibrée entre les robots (en moyenne BW = 8%).

Nous remarquons, ensuite, que la durée moyenne d'un aller-retour est augmentée par rapport à un système sous-dimensionné (en moyenne APP = 377 ms à 404 ms). Cela est dû à la réduction du débit de produits en entrée. Les produits arrivant dans la zone de travail des robots sont pris plus rapidement car il n'y en a moins. Les robots ont donc une trajectoire plus curviligne par rapport aux convoyeurs et donc mettent plus de temps pour prendre et déposer un produit.

Finalement, la stratégie de dépose *Linear* permet d'améliorer la répartition de la charge de travail entre les robots (de 8% à 7.6% en moyenne) et permet de prendre tous les produits lorsqu'il n'y a pas de stratégies de prise. La stratégie de dépose *DownToUp Cyclic*, quant à elle, permet de réduire la charge totale de travail de l'application.

Dans un troisième temps le système est dimensionné au plus juste. Les résultats des Tableaux 5.6, 5.7 et 5.8 sont obtenus respectivement avec les stratégies de dépose *Nothing*, *Linear* et *DownToUp Cyclic*.

Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
Nothing	0.00	**5.73**	**386**	378
DownToUp	0.03	1.38	390	383
DownToUp Cyclic	0.00	1.37	390	383
IntToExt	**1.88**	0.12	392	392
IntToExt Cyclic	0.02	1.82	390	382
IntToExt Alt	0.07	1.58	390	383
Horizontal	**5.10**	10.1	376	390
Vertical	**2.25**	12.0	386	388

TABLE 5.6 – Résultats des stratégies de collaboration pour un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple n'utilisant pas de stratégies de dépose.

Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
Nothing	0.00	4.82	387	379
DownToUp	**0.01**	1.42	390	383
DownToUp Cyclic	0.00	1.38	390	383
IntToExt	**1.75**	0.16	392	391
IntToExt Cyclic	0.01	1.74	390	382
IntToExt Alt	0.02	1.66	390	382
Horizontal	**4.38**	5.92	381	391
Vertical	**2.21**	12.6	385	387

TABLE 5.7 – Résultats des stratégies de collaboration pour un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple utilisant la stratégie de dépose *Linear*.

Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
Nothing	0.02	0.74	391	383
DownToUp	**0.24**	1.63	391	384
DownToUp Cyclic	0.15	1.90	390	383
IntToExt	**6.26**	5.91	381	399
IntToExt Cyclic	0.07	1.25	391	383
IntToExt Alt	0.04	0.68	388	381
Horizontal	**6.73**	13.2	373	392
Vertical	**2.00**	1.42	391	391

TABLE 5.8 – Résultats des stratégies de collaboration pour un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple utilisant la stratégie de dépose *DownToUp Cyclic*.

Les Tableaux 5.6, 5.7 et 5.8 présentent les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple. Les pires résultats sont obtenus avec les stratégies de prises *IntToExt*, *Horizontal* et *Vertical* pour toutes les stratégies de dépose. Les stratégies *IntToExt* et *Horizontal* ont le même comportement que la règle *SPT*. Le temps de prise et de dépose augmente du premier robot au dernier, voir Figure 5.3. Lorsque le système utilise ces stratégies, il y a beaucoup de pertes de produits (de 1.75% à 6.73%). Au niveau de la dépose, la pire stratégie est *DownToUp Cyclic*. Elle augmente le taux de pertes de produits pour chaque stratégie de prise (de 0.01% à 0.24% pour la stratégie *DownToUp* par exemple). Les autres stratégies de prise permettent de ne pas perdre ou très peu de produits (moins de 0.07%). Lorsqu'il n'y a pas de stratégie de prise, l'équilibre de la charge de travail est légèrement plus élevé (BW = 5.73% pour en moyenne 1.45% avec une stratégie) cependant la charge de travail totale du système est plus faible (TW = 386% pour en moyenne 390% avec une stratégie), ce qui permet d'augmenter légèrement la cadence si nécessaire.

Après cette première série de simulation, les stratégies de prise *IntToExt*, *Horizontal* et *Vertical* et la stratégie de dépose *DownToUp Cyclic* ne seront plus utilisées lors des

prochaines simulation. En effet ces stratégies entraînent une perte significative de produits.

Flux de produits simples positionnés aléatoirement avec convoyeurs en co-courant et prise multiple

Dans cette partie, outre le cahier des charges initial, de nouvelles données sur la prise multiple de produits par les robots sont ajoutées. Les robots prennent et déposent trois produits les uns à la suite des autres. Les robots prenant et déposant plusieurs produits les uns à la suite des autres, la cadence moyenne d'un robot est modifiée. Lorsque les robots prennent trois produits, leur cadence est égale à 3.29 produits par seconde, ce qui correspond à un cycle pick & place de 0.305s.

Dans un premier temps le système étudié est sous-dimensionné. Pour ce test, la stratégie *PlacingPriority* n'est pas utilisée afin de limiter les contraintes sur le système. Le nombre de boîtes non remplies est donc présent. Le système étant sous-dimensionné, tous les robots fonctionnent à 100%, l'équilibre de la charge de travail n'a pas lieu d'être étudié. Lors de la comparaison entre les stratégies de prise, aucune stratégie de dépose n'est utilisée. De même, lors de l'utilisation de la stratégie de dépose *Linear*, aucune stratégie de prise n'est utilisée. La cadence ayant changé, un nouveau dimensionnement doit être effectué. Pour un système sous-dimensionné, selon la méthode décrite dans la section 5.1, un débit de produits est choisi $FPs = 16.44$ produits par seconde. Pour sous-dimensionner le système, une marge de sécurité négative est ajoutée de telle sorte que l'arrondi ne rend pas la marge positive, $Mfp = -25\%$. Le nombre minimum de robots est $NRmin = FPs * (1 + Mfp)/PPas = 16.44 * (1 + -0.25)/3.29 = 3.75$ arrondi à 4. Comme une marge de sécurité a été utilisée, le nombre final de robots est $NRf = NRmin = 4$. En s'appuyant sur le nombre de produits dans une boîte, le débit de boîtes est égal à $FBs = FPs/NPb = 16.44/9 = 1.83$ boîtes par seconde. En utilisant la distance entre les boîtes, la vitesse du convoyeur de sortie doit être égale à $Vo = FBs * EB = 1.83 * 0.15 = 0.27$ m/s. Les résultats sont regroupés dans le Tableau 5.9 suivant l'utilisation ou non de la règle *SPT* en plus de la stratégie, voir section 4.2.2.

Le Tableau 5.9 présente les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système sous-dimensionné ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise multiple. Nous remarquons que la durée moyenne d'un aller-retour est proche du temps de cycle de pick & place (APP = 305 ms). Cela veut dire que les robots prennent et déposent en moyenne au milieu des convoyeurs. Ces résultats permettent de détecter des stratégies plus performantes que d'autres. En effet la stratégie *IntToExt Cyclic* avec et sans la règle *SPT* et les stratégies *DownToUp* et *DownToUp Cyclic* avec la règle *SPT* perdent moins de produits et de boîtes que les autres stratégies. Ce qui peut être expliqué par leur temps de déplacement plus faible (environ 300 ms).

Le temps d'attente du système (temps d'attente de produits, temps de calcul, temps d'exécution du programme etc.) est de 12 ms en moyenne. Il est plus élevé que le temps d'attente d'un système simple prise (8 ms en moyenne). Cela est normal car pour un cycle de pick & place le premier système prend et dépose trois produits donc doit rechercher trois fois un produit et trois fois une place alors que pour le deuxième système il ne doit rechercher qu'un seul produit et qu'une seule place. Le système étant sous-dimensionné, il n'y a pas d'attente de produit. De plus le système avec une prise multiple recherche

	Stratégie de prise	Produits perdus (%)	Boîtes non remplies (%)	TW (%)	APP (ms)
	Nothing	21.8	74.6	385	304
No SPT	*DownToUp*	21.9	76.7	385	304
No SPT	*DownToUp Cyclic*	22.0	73.3	385	305
No SPT	*IntToExt Cyclic*	20.5	70.5	385	**299**
No SPT	*IntToExt Alt*	22.0	76.4	385	305
SPT	*DownToUp*	20.6	76.7	385	**299**
SPT	*DownToUp Cyclic*	20.7	76.0	385	**300**
SPT	*IntToExt Cyclic*	20.5	68.3	385	**298**
SPT	*IntToExt Alt*	22.0	77.1	385	305
	Stratégie de dépose				
	Linear	21.8	78.0	385	304

TABLE 5.9 – Résultats des stratégies de collaboration pour un système sous-dimensionné ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise multiple.

deux produits et places de plus, donc on peut estimer le temps de recherche d'un produit ou d'une place à $(12-8)/4 = 1$ ms. En appliquant cette valeur, un système avec prise multiple passe donc 6 ms à chercher les produits et les places lors d'un cycle pick & place. On peut en conclure que l'application a besoin d'un temps fixe de $12-6 = 6$ ms en plus pour fonctionner.

Afin de confirmer ces résultats, quatre simulations sont effectuées avec une prise multiple de un à quatre produits. La charge de travail totale du système ainsi que la durée moyenne d'un aller-retour étant très proches entre les différentes stratégies, seule la stratégie *Nothing* est utilisée.

Nombre de prise multiple	APP (ms)	T_{Wait} (ms)	TW (%)
1	377	8	392
2	322	10	388
3	304	12	385
4	295	14	382

TABLE 5.10 – Résultats des stratégies de collaboration pour un système sous-dimensionné ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise multiple n'utilisant ni de stratégie de prise ni de stratégie de dépose pour différentes valeur de prise multiple.

Le Tableau 5.10 présente les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système sous-dimensionné ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et différentes valeurs de prise multiple. Tout d'abord nous pouvons noter que la durée moyenne d'un aller-retour diminue lorsque la prise multiple augmente. Ce comportement est normal car la distance entre les produits est plus faible que la distance entre produits et boîtes. Nous remarquons aussi que le temps d'attente augmente régulièrement de 2 ms. On peut donc estimer le temps de recherche d'un produit ou d'une place à 1 ms. Ce qui confirme les résultats précédents.

	Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
	Nothing	**1.67**	**83.9**	301	306
No SPT	*DownToUp*	0.00	12.5	350	316
	DownToUp Cyclic	0.00	30.9	324	314
	IntToExt Cyclic	0.01	5.24	276	311
	IntToExt Alt	0.15	8.28	289	315
SPT	*DownToUp*	0.01	3.05	285	311
	DownToUp Cyclic	0.02	7.10	307	311
	IntToExt Cyclic	0.00	6.14	291	311
	IntToExt Alt	0.11	6.58	291	315

TABLE 5.11 – Résultats des stratégies de collaboration pour un système surdimensionné ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise multiple n'utilisant pas de stratégie de dépose.

	Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
	Nothing	0.00	33.5	307	314
No SPT	*DownToUp*	0.00	1.91	383	316
	DownToUp Cyclic	0.00	22.1	343	314
	IntToExt Cyclic	0.00	4.44	276	310
	IntToExt Alt	**0.05**	5.75	294	314
SPT	*DownToUp*	**0.03**	7.80	290	311
	DownToUp Cyclic	0.00	6.22	300	310
	IntToExt Cyclic	0.01	7.17	298	310
	IntToExt Alt	0.09	5.60	292	314

TABLE 5.12 – Résultats des stratégies de collaboration pour un système surdimensionné ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise multiple utilisant la stratégie de dépose *Linear*.

Dans un second temps le système est surdimensionné. La cadence ayant changé, un nouveau dimensionnement doit être effectué. Pour un système surdimensionné, selon la méthode décrite dans la section 5.1, un débit de produits est choisi $FPs = 10.52$ produits par seconde. Pour surdimensionner le système, une marge de sécurité $Mfp = 20\%$ est ajoutée. Le nombre minimum de robots est $NRmin = FPs * (1 + Mfp)/PPas = 10.52 * (1 + 0.2)/3.29 = 3.84$ arrondi à 4. Comme une marge de sécurité a été utilisée, le nombre final de robots est $NRf = NRmin = 4$. En s'appuyant sur le nombre de produits dans une boîte, le débit de boîtes est égal à $FBs = FPs/NPb = 10.52/9 = 1.17$ boîtes par seconde. En utilisant la distance entre les boîtes, la vitesse du convoyeur de sortie doit être égale à $Vo = FBs * EB = 1.17 * 0.15 = 0.18$ m/s. Les résultats des Tableaux 5.11 et 5.12 sont obtenus respectivement avec les stratégies de dépose *Nothing* et *Linear*.

Les Tableaux 5.11 et 5.12 présentent les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système surdimensionné ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise

multiple. Tout d'abord nous remarquons que, comme pour un système avec une simple prise, la durée moyenne d'un aller-retour est augmentée. De plus, il y a la même tendance qui se dégage. Lorsqu'il n'y a aucune stratégie de prise ou de dépose, le système perd des produits (1.67%). Par ailleurs, le système est très déséquilibré (BW = 83.9%), ce qui peut entraîner une usure prématurée de certains robots. L'utilisation d'une stratégie de collaboration de prise permet de prendre tous les produits ou de n'en perdre que très peu (moins de 0.3% sauf pour la stratégie *IntToExt Alt*). La charge de travail est mieux équilibrée entre les robots (en moyenne BW = 9.97%). La stratégie de dépose *Linear* permet d'améliorer la répartition de la charge de travail entre les robots (de 9.97% à 7.62% en moyenne) et permet de prendre tous les produits lorsqu'il n'y a pas de stratégies de prise. Elle permet de plus d'améliorer légèrement le nombre de produits pris lors d'une utilisation d'une stratégie de prise. Comme prévu avec le système sous-dimensionné les meilleures stratégies de prise sont la stratégie *IntToExt Cyclic* avec et sans la règle *SPT* et les stratégies *DownToUp* et *DownToUp Cyclic* avec la règle *SPT*. La pire stratégie de prise est *IntToExt Alt* (plus de 0.05% de pertes de produits). Pour un système surdimensionné, l'ajout de la règle *SPT* n'apporte pas d'amélioration.

Dans un troisième temps le système est dimensionné au plus juste. La cadence ayant changé, un nouveau dimensionnement doit être effectué. Pour un système dimensionné au plus juste, selon la méthode décrite dans la section 5.1, un débit de produits est choisi $FPs = 12.89$ produits par seconde. Pour surdimensionner le système, une marge de sécurité $Mfp = 2\%$ est ajoutée. Le nombre minimum de robots est $NRmin = FPs * (1 + Mfp)/PPas = 12.89 * (1 + 0.02)/3.29 = 4$. Comme une marge de sécurité a été utilisée, le nombre final de robots est $NRf = NRmin = 4$. En s'appuyant sur le nombre de produits dans une boîte, le débit de boîtes est égal à $FBs = FPs/NPb = 12.89/9 = 1.43$ boîtes par seconde. En utilisant la distance entre les boîtes, la vitesse du convoyeur de sortie doit être égale à $Vo = FBs * EB = 1.43 * 0.15 = 0.22$ m/s. Les résultats des Tableaux 5.13 et 5.14 sont obtenus respectivement avec les stratégies de dépose *Nothing* et *Linear*.

	Stratégie de prise	**Produits perdus (%)**	**BW (%)**	**TW (%)**	**APP (ms)**
	Nothing	0.23	0.10	385	305
No SPT	*DownToUp*	0.56	0.30	385	305
	DownToUp Cyclic	0.42	0.20	385	305
	IntToExt Cyclic	**0.02**	2.37	382	303
	IntToExt Alt	**0.75**	0.36	385	306
SPT	*DownToUp*	**0.01**	0.78	384	304
	DownToUp Cyclic	**0.01**	1.72	383	303
	IntToExt Cyclic	**0.02**	2.46	382	303
	IntToExt Alt	0.72	0.29	385	306

TABLE 5.13 – Résultats des stratégies de collaboration pour un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise multiple n'utilisant pas de stratégie de dépose.

Les Tableaux 5.13 et 5.14 présentent les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système dimensionné au plus juste ayant des

	Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
	Nothing	0.41	0.03	385	305
No SPT	*DownToUp*	0.44	0.03	385	305
No SPT	*DownToUp Cyclic*	0.44	0.03	385	305
No SPT	*IntToExt Cyclic*	**0.01**	2.52	382	303
No SPT	*IntToExt Alt*	0.82	0.11	385	307
SPT	*DownToUp*	**0.04**	0.40	384	304
SPT	*DownToUp Cyclic*	**0.01**	1.43	383	303
SPT	*IntToExt Cyclic*	**0.00**	1.89	383	303
SPT	*IntToExt Alt*	0.86	0.17	385	307

TABLE 5.14 – Résultats des stratégies de collaboration pour un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise multiple utilisant la stratégie de dépose *Linear*.

convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise multiple. Tout d'abord nous remarquons que malgré la marge de sécurité, il y a une perte de produits pour toutes les stratégies, ce qui traduit d'un léger sous-dimensionnement. Cela vient du rapport temps de calcul/temps de déplacement qui est plus élevé avec un système utilisant une prise multiple. Comme indiqué dans le chapitre 4 pour l'orientation, un robot se déplace sur moins de distance entre produits qu'entre produits et boîtes. Ce qui entraîne un temps de déplacement moyen réduit alors que le temps de calcul moyen est le même. Il faut donc prendre une marge de sécurité plus grande avec une prise multiple qu'avec une simple prise. Nous notons ensuite, comme pour un système sous-dimensionné, la stratégie *IntToExt Cyclic* avec et sans la règle *SPT* et les stratégies *DownToUp* et *DownToUp Cyclic* avec la règle *SPT* sont les plus efficaces. Elles perdent très peu de produits (moins de 0.04%). La stratégie de dépose *Linear* n'a pas d'impact significatif sur les résultats. Finalement, dans tous les cas, la pire stratégie de dépose est *IntToExt Alt* (plus de 0.75% de produits perdus).

Un dernier test a été effectué afin de trouver l'apport de la stratégie *IntToExt Cyclic* avec la règle *SPT* par rapport à une application n'utilisant pas de stratégie (application industrielle). Ce test consiste à trouver le débit maximum pour les deux configurations avec lequel il n'y pas de pertes de produits. Il effectue plusieurs simulations en incrémentant le débit dans un premier temps de 0.2 produit par seconde puis dans un second temps de 0.05 produit par seconde. Chaque simulation dure 30 minutes. Une application sans stratégie de collaboration peut fonctionner correctement jusqu'à 12.8 produits par seconde tandis que l'utilisation de la stratégie de prise *IntToExt Cyclic* avec la règle *SPT* permet d'augmenter le maximum à 12.95 produits par seconde. Une application utilisant cette stratégie de prise peut donc prendre environ 1% de produits de plus qu'une application n'utilisant aucune stratégie.

Après cette seconde série de simulation, la stratégie de prise *IntToExt Alt* ne sera plus utilisée lors des prochaines simulations. En effet cette stratégie est la pire stratégie de cette série et n'était pas significative durant la première série de simulation.

Flux de produits simples positionnés aléatoirement avec convoyeurs en contre-courant et prise simple

Dans cette partie, outre le cahier des charges initial, de nouvelles données sur les convoyeurs sont ajoutées. Les convoyeurs fonctionnent en contre-courant. Les robots utilisent donc les règles de planification individuelles inversées. Les robots finissant de remplir les boîtes utilisent la règle *FIFO* pour la dépose et la règle *LIFO* pour la prise, tandis que les robots finissant de prendre les produits utilisent la règle *FIFO* pour la prise et la règle *LIFO* pour la dépose. De plus la zone de travail des robots est décalée de 200 mm dans le sens du convoyeur d'entrée pour les robots finissant de prendre les produits et dans le sens du convoyeur de sortie pour les robots finissant de remplir les boîtes.

Dans un premier temps le système est sous-dimensionné. Pour ce test, la stratégie *PlacingPriority* n'est pas utilisée afin de limiter les contraintes sur le système. Le nombre de boîtes non remplies est donc présent. Le système étant sous-dimensionné, tous les robots fonctionnent à 100%, l'équilibre de la charge de travail n'a pas lieu d'être étudiée. Lors de la comparaison entre les stratégies de prise, aucune stratégie de dépose n'est utilisée. De même, lors de l'utilisation de la stratégie de dépose *Linear*, aucune stratégie de prise n'est utilisée.

Stratégie de prise	**Produits perdus (%)**	**Boîtes non remplies (%)**	**TW (%)**	**APP (ms)**
Nothing	19.9	99.6	392	377
DownToUp	19.9	99.9	392	377
DownToUp Cyclic	20.1	99.7	392	377
IntToExt Cyclic	20.1	99.7	392	377
Stratégie de dépose				
Linear	19.9	99.8	392	377

TABLE 5.15 – Résultats des stratégies de collaboration pour un système sous-dimensionné ayant des convoyeurs en contre-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple n'utilisant pas de stratégie de dépose.

Le Tableau 5.15 présente les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système sous-dimensionné ayant des convoyeurs en contre-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple. Comme pour un système avec des convoyeurs en co-courant, nous remarquons que lorsqu'un système est sous-dimensionné, toutes les stratégies de collaboration obtiennent des résultats très similaires. Les stratégies ne peuvent donc pas être comparées sur ce genre de système. On peut cependant noter que la durée moyenne d'un aller-retour est inférieure au temps de cycle de pick & place (APP = 377ms pour 383 ms). Cela veut dire que les robots prennent les produits les plus à l'intérieur du convoyeur d'entrée.

Dans un second temps, le système étudié est surdimensionné. Les résultats des Tableaux 5.16 et 5.17 sont obtenus respectivement avec les stratégies de dépose *Nothing* et *Linear*.

Les Tableaux 5.16 et 5.17 présentent les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système surdimensionné ayant des convoyeurs en contre-courant, avec un flux de produits simples positionnés aléatoirement et une prise

Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
Nothing	0.00	**62.5**	329	403
DownToUp	0.00	16.8	346	415
DownToUp Cyclic	0.00	19.1	345	414
IntToExt Cyclic	0.00	19.3	345	414

TABLE 5.16 – Résultats des stratégies de collaboration pour un système surdimensionné ayant des convoyeurs en contre-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple n'utilisant pas de stratégie de dépose.

Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
Nothing	0.00	**65.2**	326	401
DownToUp	0.00	12.8	347	417
DownToUp Cyclic	0.00	17.7	344	413
IntToExt Cyclic	0.00	16.4	344	413

TABLE 5.17 – Résultats des stratégies de collaboration pour un système surdimensionné ayant des convoyeurs en contre-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple utilisant la stratégie de dépose *Linear*.

simple. Tout d'abord, nous pouvons noter que pour toutes les stratégies de prise et de dépose, tous les produits sont pris. Comme expliqué dans la section 4.4, les robots en amont du convoyeur de sortie ont peu de produits pour beaucoup de places tandis que les robots en aval du convoyeur de sortie ont beaucoup de produits pour peu de places. Cela permet de trouver rapidement un produit pour combler une place restante en aval et une place pour déposer un produit restant en amont. On retrouve cependant le déséquilibre de la charge de travail entre les robots lorsqu'il n'y a pas de stratégie de collaboration pour la prise (BW = 63.85% en moyenne). La stratégie de dépose *Linear* permet d'améliorer légèrement l'équilibre de la charge de travail entre les robots lors de l'utilisation d'une stratégie de prise (BW = 15.6 % pour 18.4% en moyenne). Comme précédemment, la durée moyenne d'un aller-retour est augmentée.

Dans un troisième temps le système est dimensionné au plus juste. Les résultats des Tableaux 5.18 et 5.19 sont obtenus respectivement avec les stratégies de dépose *Nothing* et *Linear*.

Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
Nothing	0.02	0.17	392	384
DownToUp	0.01	3.47	390	382
DownToUp Cyclic	**0.04**	3.18	390	382
IntToExt Cyclic	0.01	3.48	390	382

TABLE 5.18 – Résultats des stratégies de collaboration pour un système dimensionné au plus juste ayant des convoyeurs en contre-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple n'utilisant pas de stratégies de dépose.

Les Tableaux 5.18 et 5.19 présentent les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système dimensionné au plus juste ayant des

Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
Nothing	0.01	0.18	392	384
DownToUp	0.02	3.22	390	382
DownToUp Cyclic	**0.04**	3.11	390	382
IntToExt Cyclic	0.02	3.01	390	382

TABLE 5.19 – Résultats des stratégies de collaboration pour un système dimensionné au plus juste ayant des convoyeurs en contre-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple utilisant la stratégie de dépose *Linear*.

convoyeurs en contre-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple. Les résultats obtenus sont similaires pour toutes stratégies de prise et de dépose. Il y a un faible pourcentage de produits perdus (moins de 0.04%). La charge de travail totale de l'application est proche de la valeur maximale (environ 391%, TW = 392% pour un système sous-dimensionné), ce qui indique que le débit du système ne peut pas être augmenté et donc que le système est légèrement sous-dimensionné. Comparé à un système avec des convoyeurs en co-courant, celui-ci possède un débit maximum inférieur. Cette donnée confirme le résultat trouvé dans la littérature par [Comba et al., 2013].

Flux de produits simples positionnés en ligne avec convoyeurs co-courant et prise simple

Dans cette partie, outre le cahier des charges initial, de nouvelles données sur le positionnement des produits sur le convoyeur d'entrée sont ajoutées. Les produits sont positionnés en ligne. Les simulations sont effectuées pour différents nombres de lignes de produits : 4, 7 et 10.

Dans un premier temps le système est sous-dimensionné. Pour ce test, la stratégie *PlacingPriority* n'est pas utilisée afin de limiter les contraintes sur le système. Le nombre de boîtes non remplies est donc présent. Le système étant sous-dimensionné, tous les robots fonctionnent à 100%, l'équilibre de la charge de travail n'a pas lieu d'être étudié.

Le Tableau 5.20 présente les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système sous-dimensionné ayant des convoyeurs en co-courant, avec un flux de produits simples en ligne et une prise simple. Comme pour les autres systèmes avec prise simple, nous remarquons que lorsqu'un système est sous-dimensionné, toutes les stratégies de collaboration obtiennent des résultats très similaires. Les stratégies ne peuvent donc pas être comparées sur ce genre de système. On peut cependant noter que la durée moyenne d'un aller-retour est inférieure au temps de cycle de pick & place (APP = 371 ms en moyenne pour 383 ms). Cela veut dire que les robots prennent les produits les plus à l'intérieur du convoyeur d'entrée. Pour un système qui n'utilise pas de stratégie, cette observation est flagrante. En effet, le système a le même comportement que lorsque la règle *SPT* est utilisée. Le temps de prise et de dépose augmente du premier robot au dernier, voir Figure 5.3. Le remplissage se fait de façon régressive. Nous remarquons aussi que plus le nombre de lignes est grand, plus la durée moyenne d'un aller-retour baisse légèrement et donc plus le nombre de produits perdus baisse légèrement. De plus, le nombre de produits perdus et le temps d'un aller-retour sont inférieurs à celui d'un système avec un flux de produits positionnés aléatoirement (18.8% contre 19.9% et 371 ms pour 377

Stratégie de prise stratégie de dépose	Nombre de lignes	Produits perdus (%)	Boîtes non remplies (%)	TW (%)	APP (ms)
Nothing Nothing	4	18.8	99.6	392	371
	7	18.4	99.6	391	370
	10	18.3	99.6	391	369
DownToUp Cyclic Nothing	4	20.3	99.8	392	377
	7	18.9	99.6	392	370
	10	18.8	99.6	392	370
Nothing Linear	4	18.8	99.4	392	371
	7	18.4	99.6	391	369
	10	18.3	99.6	391	369
DownToUp Cyclic Linear	4	20.3	99.8	392	377
	7	18.9	99.7	392	370
	10	18.8	99.6	392	370

TABLE 5.20 – Résultats des stratégies de collaboration pour un système sous-dimensionné ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés en ligne et une prise simple.

ms en moyenne).

Dans un second temps le système étudié est surdimensionné.

Le Tableau 5.21 présente les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système surdimensionné ayant des convoyeurs en co-courant, avec un flux de produits simples en ligne et une prise simple. Les résultats obtenus suivent la même tendance que ceux obtenus avec un flux de produits positionnés aléatoirement. Il n'y a pas de perte de produits lorsqu'une stratégie de prise ou de dépose est appliquée. Il y a un déséquilibre de la charge de travail entre les robots lorsqu'il n'y a aucune stratégie de collaboration pour la prise et pour la dépose (BW = 79% en moyenne). Cependant, comme pour un système sous-dimensionné, le taux de produits perdus sans stratégie est légèrement plus faible que pour un système avec un flux de produits positionnés aléatoirement (0.35% en moyenne contre 0.90%). Nous remarquons que plus il y a de lignes de produits moins il y a de pertes de produits. Lorsqu'il n'y a pas de stratégie de prise, l'équilibre de la charge de travail entre les robots est amélioré (de 80.0% à 77.8%), mais le résultat inverse est obtenu avec une stratégie de prise (de 5.80% à 9.76%).

Dans un troisième temps le système est dimensionné au plus juste.

Le Tableau 5.22 présente les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits simples en ligne et une prise simple. Tout d'abord, nous remarquons que, pour toutes les stratégies de prise et de dépose, tous les produits sont pris. De plus, la stratégie *Nothing* possède une meilleure charge de travail totale (TW = 385%) et un meilleur temps d'aller-retour (APP = 380 ms). Cela indique que le système peut avoir une marge de sécurité plus faible qu'avec une stratégie. Finalement, le nombre de lignes de produits influe de la même façon qu'avec un système surdimensionné.

Stratégie de prise stratégie de dépose	**Nombre de lignes**	**Produits perdus (%)**	**BW (%)**	**TW (%)**	**APP (ms)**
Nothing Nothing	4	**0.44**	**80.0**	312	384
	7	**0.32**	**78.8**	313	387
	10	**0.29**	**77.8**	314	388
DownToUp Cyclic Nothing	4	0.00	6.32	348	418
	7	0.00	9.71	342	410
	10	0.00	9.41	339	406
Nothing Linear	4	0.00	26.1	334	407
	7	0.00	25.5	333	404
	10	0.00	25.2	331	403
DownToUp Cyclic Linear	4	0.00	**5.80**	348	418
	7	0.00	8.56	341	409
	10	0.00	**9.76**	339	406

TABLE 5.21 – Résultats des stratégies de collaboration pour un système surdimensionné ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés en ligne et une prise simple.

Stratégie de prise stratégie de dépose	**Nombre de lignes**	**Produits perdus (%)**	**BW (%)**	**TW (%)**	**APP (ms)**
Nothing Nothing	4	0.00	6.90	**385**	**380**
	7	0.00	6.43	385	380
	10	0.00	6.04	386	380
DownToUp Cyclic Nothing	4	0.00	0.72	391	383
	7	0.00	1.29	391	382
	10	0.00	1.63	390	382
Nothing Linear	4	0.00	4.00	388	382
	7	0.00	3.45	389	383
	10	0.00	3.82	388	382
DownToUp Cyclic Linear	4	0.00	0.77	391	383
	7	0.00	1.22	391	382
	10	0.00	1.59	390	382

TABLE 5.22 – Résultats des stratégies de collaboration pour un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés en ligne et une prise simple.

Flux variable de produits simples positionnés aléatoirement et prise simple

Dans cette partie, outre le cahier des charges initial, de nouvelles données sur le flux de produits sur le convoyeur d'entrée sont ajoutées. Le flux varie entre 10.96 prod/s et 9.39 prod/s toutes les 5 secondes ce qui correspond à une marge de sécurité variant de -5% à 10%. Le système est initialement dimensionné avec une marge de sécurité de 2%. Cela permet de fixer la vitesse du convoyeur de sortie à 0.17 m/s. Au cours de la simulation seul le flux d'entrée varie. Dans un premier temps le système étudié possède des convoyeurs en co-courant. Les résultats des Tableaux 5.23 et 5.24 sont obtenus respectivement avec les stratégies de dépose *Nothing* et *Linear*.

Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
Nothing	**0.71**	**11.3**	380	380
DownToUp	0.24	0.46	390	387
DownToUp Cyclic	0.29	1.63	390	386
IntToExt Cyclic	0.19	1.81	390	385

TABLE 5.23 – Résultats des stratégies de collaboration pour un système ayant des convoyeurs en co-courant, avec un flux variable de produits simples positionnés aléatoirement et une prise simple n'utilisant pas de stratégie de dépose.

Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
Nothing	**0.69**	**7.55**	384	383
DownToUp	0.16	1.34	391	386
DownToUp Cyclic	0.15	1.82	390	387
IntToExt Cyclic	0.14	1.61	390	385

TABLE 5.24 – Résultats des stratégies de collaboration pour un système ayant des convoyeurs en co-courant, avec un flux variable de produits simples positionnés aléatoirement et une prise simple utilisant la stratégie de dépose *Linear*.

Les Tableaux 5.23 et 5.24 présentent les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système ayant des convoyeurs en co-courant, avec un flux variable de produits simples positionnés aléatoirement et une prise simple. Tout d'abord nous remarquons qu'il y a une perte de produits pour tous les ensembles de stratégies de collaboration. L'absence de stratégie pour la prise augmente la perte de produits (environ 0.70% de pertes). Comme pour les simulations précédentes, cette absence entraîne un déséquilibre de la charge de travail entre les robots (BW = 9.5% pour environ 1.5% en moyenne avec une stratégie pour la prise). La stratégie de dépose *Linear* réduit légèrement le nombre de produits perdus.

Dans un second temps le système possède des convoyeurs en contre-courant. Les résultats des Tableaux 5.25 et 5.26 sont obtenus respectivement avec les stratégies de dépose *Nothing* et *Linear*.

Les Tableaux 5.25 et 5.26 présentent les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système ayant des convoyeurs en contre-courant, avec un flux variable de produits simples positionnés aléatoirement et une prise simple.

Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
Nothing	0.35	3.30	388	385
DownToUp	0.07	5.09	388	384
DownToUp Cyclic	0.10	7.96	388	383
IntToExt Cyclic	0.07	4.86	388	383

TABLE 5.25 – Résultats des stratégies de collaboration pour un système ayant des convoyeurs en contre-courant, avec un flux variable de produits simples positionnés aléatoirement et une prise simple n'utilisant pas de stratégie de dépose.

Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
Nothing	0.25	2.49	389	386
DownToUp	0.01	4.84	388	383
DownToUp Cyclic	0.10	4.41	388	383
IntToExt Cyclic	0.09	4.11	388	383

TABLE 5.26 – Résultats des stratégies de collaboration pour un système ayant des convoyeurs en contre-courant, avec un flux variable de produits simples positionnés aléatoirement et une prise simple utilisant la stratégie de dépose *Linear*.

Tout d'abord nous remarquons qu'il y a une perte de produits pour tous les ensembles de stratégies de collaboration. Cependant cette perte est inférieure à celle du système avec des convoyeurs en co-courant (en moyenne 0.13% de produits perdus pour des convoyeurs en contre-courant pour 0.32% de produits perdus pour des convoyeurs en co-courant). Cela montre que la configuration avec des convoyeur en contre-courant permet de perdre moins de produits en cas de perturbations que celle avec des convoyeurs en co-courant, résultat trouvé dans la littérature par [Comba et al., 2013].

5.4.2 Produits avec caractéristiques

Flux aléatoire avec produits orientés et convoyeurs en co-courant et prise simple

Dans cette partie, outre le cahier des charges initial, les nouvelles données suivantes sont ajoutées :

- Les produits sont positionnés aléatoirement sur le convoyeur d'entrée avec une orientation devant être prise en compte.
- Les robots travaillent avec une vitesse linéaire de 5 m/s et une accélération linéaire de 50 m/s^2.
- Les robots travaillent avec une vitesse angulaire de 10π rad/s et une accélération angulaire de 100π rad/s^2.
- Les boîtes contiennent 8 produits.

Le dimensionnement du système est fait en ne prenant pas en compte l'orientation des produits (ceci afin de montrer l'impact de ce paramètre). Cependant la vitesse et l'accélération maximum des robots étant réduites, la cadence moyenne des robots est modifiée. Un cycle de pick & place pour un robot prend en moyenne 0.538 secondes. Le robot peut prendre $PPas = 1/0.538 = 1.86$ produits par seconde.

Dans un premier temps le système étudié est sous-dimensionné. Pour ce test, la stratégie *PlacingPriority* n'est pas utilisée afin de limiter les contraintes sur le système. Le nombre de boîtes non remplies est donc présent. Le système étant sous-dimensionné, tous les robots fonctionnent à 100%, l'équilibre de la charge de travail n'a pas lieu d'être étudié. Lors de la comparaison entre les stratégies de prise, aucune stratégie de dépose n'est utilisée. De même, lors de l'utilisation de la stratégie de dépose *Linear*, aucune stratégie de prise n'est utilisée. La cadence ayant changé, un nouveau dimensionnement doit être effectué. Pour un système sous-dimensionné, selon la méthode décrite dans la section 5.1, un débit de produits est choisi $FPs = 9.29$ produits par seconde. Pour sous-dimensionner le système, une marge de sécurité négative est ajoutée de telle sorte que l'arrondi ne rend pas la marge positive, $Mfp = -25\%$. Le nombre minimum de robots est $NRmin = FPs * (1 + Mfp)/PPas = 9.29 * (1 + -0.25)/1.86 = 3.75$ arrondi à 4. Comme une marge de sécurité a été utilisée, le nombre final de robots est $NRf = NRmin = 4$. En s'appuyant sur le nombre de produits dans une boîte, le débit de boîtes est égal à $FBs = FPs/NPb = 9.29/8 = 1.16$ boîtes par seconde. En utilisant la distance entre les boîtes, la vitesse du convoyeur de sortie doit être égale à $Vo = FBs * EB = 1.16 * 0.15 = 0.17$ m/s.

Stratégie de prise	**Produits perdus (%)**	**Boîtes non remplies (%)**	**TW (%)**	**APP (ms)**
Nothing	22.7	90.9	394	550
DownToUp	22.7	91.3	394	551
DownToUp Cyclic	22.6	91.4	394	550
IntToExt Cyclic	22.9	91.1	394	552
Stratégie de dépose				
Linear	22.7	91.7	394	551

TABLE 5.27 – Résultats des stratégies de collaboration pour un système sous-dimensionné ayant des convoyeurs en co-courant, avec un flux de produits orientés positionnés aléatoirement et une prise simple.

Le Tableau 5.27 présente les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système sous-dimensionné ayant des convoyeurs en co-courant, avec un flux de produits avec rotation positionnés aléatoirement et une prise simple. Tout d'abord, nous remarquons que le temps d'un aller-retour est plus grand que le temps d'un cycle pick & place (APP = 551 ms pour 538 ms), ce qui est normal car le temps d'un cycle pick & place a été donné pour un système qui ne prend pas en compte l'orientation des produits. Cela montre qu'il est donc important de la considérer. Nous pouvons noter que lorsqu'un système est sous-dimensionné, toutes les stratégies de collaboration obtiennent des résultats très similaires. Les stratégies ne peuvent donc pas être comparées sur ce genre de système.

Dans un second temps le système étudié est surdimensionné. La cadence ayant changé, un nouveau dimensionnement doit être effectué. Pour un système surdimensionné, selon la méthode décrite dans la section 5.1, un débit de produits est choisi $FPs = 5.94$ produits par seconde. Pour surdimensionner le système, une marge de sécurité $Mfp = 20\%$ est ajoutée. Le nombre minimum de robots est $NRmin = FPs * (1 + Mfp)/PPas = 5.94 *$

$(1+0.2)/1.86 = 3.83$ arrondi à 4. Comme une marge de sécurité a été utilisée, le nombre final de robots est $NRf = NRmin = 4$. En s'appuyant sur le nombre de produits dans une boîte, le débit de boîtes est égal à $FBs = FPs/NPb = 5.94/8 = 0.74$ boîtes par seconde. En utilisant la distance entre les boîtes, la vitesse du convoyeur de sortie doit être égale à $Vo = FBs * EB = 0.74 * 0.15 = 0.11$ m/s. Les résultats des Tableaux 5.28 et 5.29 sont obtenus respectivement avec les stratégies de dépose *Nothing* et *Linear*.

Stratégie de prise	**Produits perdus (%)**	**BW (%)**	**TW (%)**	**APP (ms)**
Nothing	1.23	71.7	323	554
DownToUp	0.01	11.9	364	616
DownToUp Cyclic	0.02	10.6	360	608
IntToExt Cyclic	0.02	11.2	361	610

TABLE 5.28 – Résultats des stratégies de collaboration pour un système surdimensionné ayant des convoyeurs en co-courant, avec un flux de produits orientés positionnés aléatoirement et une prise simple n'utilisant pas de stratégie de dépose.

Stratégie de prise	**Produits perdus (%)**	**BW (%)**	**TW (%)**	**APP (ms)**
Nothing	0.00	17.0	369	**623**
DownToUp	0.00	7.65	342	579
DownToUp Cyclic	0.00	6.36	338	571
IntToExt Cyclic	0.00	4.13	338	572

TABLE 5.29 – Résultats des stratégies de collaboration pour un système surdimensionné ayant des convoyeurs en co-courant, avec un flux de produits orientés positionnés aléatoirement et une prise simple utilisant la stratégie de dépose *Linear*.

Les Tableaux 5.28 et 5.29 présentent les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système surdimensionné ayant des convoyeurs en co-courant, avec un flux de produits avec rotation positionnés aléatoirement et une prise simple. Nous remarquons que les résultats suivent la même tendance que pour une application avec des produits simples tant au niveau perte de produits qu'au niveau charge de travail. Seul le temps d'un aller-retour pour la stratégie de prise *Nothing* et la stratégie de dépose *Linear* est différent. Il est plus élevé par rapport aux autres stratégies de prise (APP = 623 ms pour 575 ms) alors que pour des produits simples, il est légèrement inférieur (APP = 407 ms pour 415 ms).

Dans un troisième temps le système est dimensionné au plus juste. La cadence ayant changé, un nouveau dimensionnement doit être effectué. Pour un système dimensionné au plus juste, selon la méthode décrite dans la section 5.1, un débit de produits est choisi $FPs = 7.28$ produits par seconde. Pour surdimensionner le système, une marge de sécurité $Mfp = 2\%$ est ajoutée. Le nombre minimum de robots est $NRmin = FPs * (1 + Mfp)/PPas = 7.28 * (1 + 0.02)/1.86 = 4$. Comme une marge de sécurité a été utilisée, le nombre final de robots est $NRf = NRmin = 4$. En s'appuyant sur le nombre de produits dans une boîte, le débit de boîtes est égal à $FBs = FPs/NPb = 7.28/8 = 0.91$ boîtes par seconde. En utilisant la distance entre les boîtes, la vitesse du convoyeur de sortie doit

être égale à $Vo = FBs * EB = 0.91 * 0.15 = 0.14$ m/s. Les résultats des Tableaux 5.30 et 5.31 sont obtenus respectivement avec les stratégies de dépose *Nothing* et *Linear*.

Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
Nothing	1.69	0.13	394	553
DownToUp	1.76	0.10	394	554
DownToUp Cyclic	1.77	0.09	394	554
IntToExt Cyclic	1.59	0.07	394	553

TABLE 5.30 – Résultats des stratégies de collaboration pour un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits orientés positionnés aléatoirement et une prise simple n'utilisant pas de stratégie de dépose.

Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
Nothing	1.82	0.11	394	554
DownToUp	1.50	0.10	394	552
DownToUp Cyclic	1.36	0.12	394	552
IntToExt Cyclic	1.29	0.11	394	551

TABLE 5.31 – Résultats des stratégies de collaboration pour un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits orientés positionnés aléatoirement et une prise simple utilisant la stratégie de dépose *Linear*.

Les Tableaux 5.30 et 5.31 présentent les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits avec rotation positionnés aléatoirement et une prise simple. Nous remarquons que le système est sous-dimensionné, par le fait de l'orientation des produits. Pour pallier ce problème, il faudrait prendre en compte l'orientation dans le dimensionnent.

Lorsque le système utilise une simple prise, il n'y a peu de différences entre des produits simples et des produits orientés. Cela se traduit par une question de cadence des robots. Á part se préoccuper de l'orientation dans le dimensionnement, il n'y a pas de stratégies qui la prennent en compte en temps réel.

Flux aléatoire avec produits orientés et convoyeurs en co-courant et prise multiple

Dans cette partie, outre le cahier des charges initial, les nouvelles données suivantes sont ajoutées :

- Les produits sont positionnés aléatoirement sur le convoyeur d'entrée avec une orientation devant être prise en compte.
- Les robots travaillent avec une vitesse linéaire de 5 m/s et une accélération linéaire de 50 m/s^2.
- Les robots travaillent avec une vitesse angulaire de 10π rad/s et une accélération angulaire de 100π rad/s^2.
- Les boîtes contiennent 8 produits.

Les stratégies avec suffixe *SPT* utilisent la même règle *SPT* que pour des temps de translation, voir section 4.2.2. Les stratégies avec le suffixe *Rot* prennent en compte le temps de translation et le temps d'orientation, pour prendre et déposer les produits, dans leurs calculs, voir section 4.3. Le dimensionnement du système est fait en ne prenant pas en compte l'orientation des produits (ceci afin de montrer l'impact de ce paramètre). Cependant la vitesse et l'accélération maximum des robots étant réduites et les robots prenant et déposant plusieurs produits, la cadence moyenne des robots est modifiée. Un cycle de pick & place pour un robot prend en moyenne 0.374 seconde. Le robot peut prendre $PPas = 1/0.374 = 2.67$ produits par seconde.

Dans un premier temps le système étudié est sous-dimensionné. La cadence ayant changé, un nouveau dimensionnement doit être effectué. Selon la méthode décrite dans la section 5.1, un débit de produits est choisi $FPs = 13.36$ produits par seconde. Pour sous-dimensionner le système, une marge de sécurité négative est ajoutée de telle sorte que l'arrondi ne rend pas la marge positive, $Mfp = -25\%$. Le nombre minimum de robots est $NRmin = FPs*(1+Mfp)/PPas = 13.36*(1+-0.25)/2.67 = 3.82$ arrondi à 4. Comme une marge de sécurité a été utilisée, le nombre final de robots est $NRf = NRmin = 4$. En s'appuyant sur le nombre de produits dans une boîte, le débit de boîtes est égal à $FBs = FPs/NPb = 13.636/8 = 1.67$ boîtes par seconde. En utilisant la distance entre les boîtes, la vitesse du convoyeur de sortie doit être égale à $Vo = FBs*EB = 1.67*0.15 = 0.25$ m/s. Pour ce test, la stratégie *PlacingPriority* n'est pas utilisée afin de limiter les contraintes sur le système. Le nombre de boîtes non remplies est donc présent. Le système étant sous-dimensionné, tous les robots fonctionnent à 100%, l'équilibre de la charge de travail n'a pas lieu d'être étudiée. Lors de la comparaison entre les stratégies de prise, aucune stratégie de dépose n'est utilisée. De même, lors de l'utilisation de la stratégie de dépose *Linear*, aucune stratégie de prise n'est utilisée.

	Stratégie de prise	**Produits perdus (%)**	**Boîtes non remplies (%)**	**TW (%)**	**APP (ms)**
	Nothing	30.5	79.8	389	424
	DownToUp	30.5	77.5	389	424
	DownToUp Cyclic	30.6	79.2	389	424
	IntToExt Cyclic	30.0	78.8	389	421
SPT	*DownToUp*	30.1	79.0	389	421
SPT	*DownToUp Cyclic*	30.2	81.0	389	422
SPT	*IntToExt Cyclic*	30.0	77.0	389	424
Rot	*DownToUp*	30.5	80.1	389	424
Rot	*DownToUp Cyclic*	30.6	79.7	389	424
Rot	*IntToExt Cyclic*	**28.8**	**16.9**	**389**	**414**
	Stratégie de dépose				
	Linear	27.9	85.7	389	425

TABLE 5.32 – Résultats des stratégies de collaboration pour un système sous-dimensionné ayant des convoyeurs en co-courant, avec un flux de produits orientés positionnés aléatoirement et une prise multiple.

Le Tableau 5.32 présente les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système sous-dimensionné ayant des convoyeurs en co-courant,

avec un flux de produits orientés positionnés aléatoirement et une prise multiple. Comme pour une application avec des produits simples, ces résultats permettent de détecter des stratégies plus performantes que d'autres. En effet la stratégie *IntToExt Cyclic Rot* perd moins de produits et de boîtes que les autres stratégies. Ce qui peut être expliqué par son temps de déplacement plus faible (environ 414ms). Les stratégies de prise avec la règle *SPT* sont moins performantes, mais plus qu'en absence de stratégies. Les mauvaises performances des autres stratégies prenant en compte l'orientation peuvent être expliquées par le fait que le système est sous-dimensionné donc que des produits ne soient pas pris par les robots. Si un robot rate un produit, la prise de ces derniers va être décalée. En effet, les produits sont assignés par groupe (ici des groupes de trois) suivant le temps de prise entre chaque produit. Si un robot rate un des produits du groupe, il n'en prendra donc que deux, et finira par utiliser son effecteur avec un produit d'un autre groupe. Ceci peut entraîner une durée de déplacement plus longue qu'initialement prévue car le produit peut être dans une orientation non favorable.

Dans un second temps le système étudié est surdimensionné. La cadence ayant changé, un nouveau dimensionnement doit être effectué. Pour un système surdimensionné, selon la méthode décrite dans la section 5.1, un débit de produits est choisi $FPs = 8.55$ produits par seconde. Pour surdimensionner le système, une marge de sécurité $Mfp = 20\%$ est ajoutée. Le nombre minimum de robots est $NRmin = FPs * (1 + Mfp)/PPas = 8.55 * (1 + 0.2)/2.67 = 3.84$ arrondi à 4. Comme une marge de sécurité a été utilisée, le nombre final de robots est $NRf = NRmin = 4$. En s'appuyant sur le nombre de produits dans une boîte, le débit de boîtes est égal à $FBs = FPs/NPb = 8.55/8 = 1.07$ boîtes par seconde. En utilisant la distance entre les boîtes, la vitesse du convoyeur de sortie doit être égale à $Vo = FBs * EB = 1.07 * 0.15 = 0.16$ m/s. Les résultats des Tableaux 5.33 et 5.34 sont obtenus respectivement avec les stratégies de dépose *Nothing* et *Linear*.

	Stratégie de prise	**Produits perdus (%)**	**BW (%)**	**TW (%)**	**APP (ms)**
	Nothing	0.09	43.4	346	424
	DownToUp	0.00	18.0	374	441
	DownToUp Cyclic	0.02	5.35	375	436
	IntToExt Cyclic	0.05	6.31	**354**	437
SPT	*DownToUp*	0.00	3.28	355	438
SPT	*DownToUp Cyclic*	0.03	8.80	362	435
SPT	*IntToExt Cyclic*	0.03	7.58	**361**	435
Rot	*DownToUp*	0.01	12.7	373	440
Rot	*DownToUp Cyclic*	0.02	9.25	367	436
Rot	*IntToExt Cyclic*	0.01	9.57	**344**	431

TABLE 5.33 – Résultats des stratégies de collaboration pour un système surdimensionné ayant des convoyeurs en co-courant, avec un flux de produits orientés positionnés aléatoirement et une prise multiple n'utilisant pas de stratégie de dépose.

Les Tableaux 5.33 et 5.34 présentent les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système surdimensionné ayant des convoyeurs en co-courant, avec un flux de produits orientés positionnés aléatoirement et une prise multiple. Tout d'abord, il est notable que la stratégie de dépose *Linear* améliore le nombre de

	Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
	Nothing	0.00	12.7	373	446
	DownToUp	0.00	8.91	369	432
	DownToUp Cyclic	0.00	2.57	372	432
	IntToExt Cyclic	**0.01**	9.37	359	431
SPT	*DownToUp*	0.00	9.54	362	432
SPT	*DownToUp Cyclic*	0.00	7.20	363	430
SPT	*IntToExt Cyclic*	0.00	3.64	360	349
Rot	*DownToUp*	0.00	29.1	364	434
Rot	*DownToUp Cyclic*	0.00	5.95	370	430
Rot	*IntToExt Cyclic*	0.00	9.75	355	523

TABLE 5.34 – Résultats des stratégies de collaboration pour un système surdimensionné ayant des convoyeurs en co-courant, avec un flux de produits orientés positionnés aléatoirement et une prise multiple utilisant la stratégie de dépose *Linear*.

produits perdus (il passe à moins de 0.01%) ainsi que la durée moyenne d'un aller-retour pour toutes les stratégies de prise (il est réduit de 6 ms en moyenne). Un résultat étonnant est la perte de produits avec la stratégie *Nothing* alors qu'il possède un temps moyen d'aller-retour le plus petit. Pour le moment, nous ne pouvons pas expliquer ce résultat. De futurs tests devront être effectués. Nous remarquons que les résultats suivent la même tendance que pour une application avec une prise simple. L'utilisation d'une stratégie prenant en compte l'orientation des produits n'a que très peu d'influence. Cela est dû au surdimensionnement de l'application. Les robots "ont du temps" pour prendre les produits. Pour un système surdimensionné, la seule amélioration qu'apporte une stratégie de collaboration prenant en compte l'orientation des produits est la réduction de la charge de travail totale de l'application (TW = 344% au lieu de 354% pour la stratégie *IntToExt Cyclic*.

Dans un troisième temps le système est dimensionné au plus juste. La cadence ayant changé, un nouveau dimensionnement doit être effectué. Pour un système dimensionné au plus juste, selon la méthode décrite dans la section 5.1, un débit de produits est choisi $FPs = 10.48$ produits par seconde. Pour surdimensionner le système, une marge de sécurité $Mfp = 2\%$ est ajoutée. Le nombre minimum de robots est $NRmin = FPs * (1 + Mfp)/PPas = 10.48 * (1 + 0.02)/2.67 = 4$. Comme une marge de sécurité a été utilisée, le nombre final de robots est $NRf = NRmin = 4$. En s'appuyant sur le nombre de produits dans une boîte, le débit de boîtes est égal à $FBs = FPs/NPb = 10.48/8 = 1.31$ boîtes par seconde. En utilisant la distance entre les boîtes, la vitesse du convoyeur de sortie doit être égale à $Vo = FBs * EB = 1.34 * 0.15 = 0.20$ m/s. Les résultats des Tableaux 5.35 et 5.36 sont obtenus respectivement avec les stratégies de dépose *Nothing* et *Linear*.

Les Tableaux 5.35 et 5.36 présentent les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits orientés positionnés aléatoirement et une prise multiple. Tout d'abord, nous remarquons que pour chaque ensemble de stratégies de collaboration il y a une perte de produits. Cela est dû au dimensionnement de cette application. En effet il ne prend pas en compte l'orientation des produits et donc l'augmen-

	Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
	Nothing	11.6	1.43	386	425
	DownToUp	11.6	1.92	387	425
	DownToUp Cyclic	11.8	1.66	387	426
	IntToExt Cyclic	11.1	1.04	388	423
SPT	*DownToUp*	11.2	1.51	387	423
SPT	*DownToUp Cyclic*	11.4	2.20	387	424
SPT	*IntToExt Cyclic*	11.2	1.32	388	423
Rot	*DownToUp*	11.7	2.28	387	425
Rot	*DownToUp Cyclic*	11.8	2.07	387	426
Rot	*IntToExt Cyclic*	**10.1**	1.49	387	**418**

TABLE 5.35 – Résultats des stratégies de collaboration pour un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits orientés positionnés aléatoirement et une prise multiple n'utilisant pas de stratégie de dépose.

	Stratégie de prise	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
	Nothing	11.8	0.03	389	428
	DownToUp	11.7	0.03	389	427
	DownToUp Cyclic	11.8	0.06	389	427
	IntToExt Cyclic	11.3	0.15	389	425
SPT	*DownToUp*	11.3	0.11	389	425
SPT	*DownToUp Cyclic*	11.4	0.22	389	425
SPT	*IntToExt Cyclic*	11.3	0.08	389	425
Rot	*DownToUp*	11.7	0.07	389	427
Rot	*DownToUp Cyclic*	11.7	0.08	389	427
Rot	*IntToExt Cyclic*	**10.3**	0.07	389	**420**

TABLE 5.36 – Résultats des stratégies de collaboration pour un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits orientés positionnés aléatoirement et une prise multiple utilisant la stratégie de dépose *Linear*.

tation possible du temps de prise et de dépose. Ce qui entraîne un sous-dimensionnement du système. Cependant, comme pour la simulation avec un système sous-dimensionné, la stratégie *IntToExt Cyclic* prenant en compte l'orientation permet de perdre moins de produits (10.2% au lieu de 11.5% de moyenne pour les autres stratégies). Ce qui peut être expliqué son temps de déplacement plus faible (environ 420 ms). Les stratégies de prise avec la règle *SPT* sont moins performantes, mais plus qu'en absence de stratégie (APP = 424 ms pour 457 ms en moyenne). La stratégie de dépose *Linear* permet d'améliorer l'équilibre de la charge de travail entre les robots. Elle augmente légèrement la durée moyenne d'un aller-retour (de 2 ms en moyenne). Nous pouvons aussi noter que la durée moyenne d'un aller-retour pour toutes les stratégies a augmenté par rapport au système sous-dimensionné précédemment (APP = 425 ms pour 422 ms en moyenne) car le débit de produits en entrée a été réduit et donc les robots prennent les produits sur plus de surface de convoyeur.

Un dernier test a été effectué afin de trouver l'apport de la stratégie *IntToExt Cyclic Rot* par rapport à une application n'utilisant pas de stratégie (application industrielle). Ce test consiste à trouver le débit maximum pour les deux configurations avec lequel il n'y pas de pertes de produits. Il effectue plusieurs simulations en incrémentant le débit dans un premier temps de 0.2 produit par seconde puis de 0.05 produit par seconde dans un second temps. Chaque simulation dure 30 minutes. Une application sans stratégie de collaboration peut fonctionner correctement jusqu'à 9.25 prod/s tandis que l'utilisation de la stratégie de prise *IntToExt Cyclic Rot* et de la stratégie de dépose *Linear* permet d'augmenter le maximum à 9.4 prod/s. Une application utilisant cette stratégie de prise peut donc prendre environ 1.6% de produits de plus qu'une application n'utilisant aucune stratégie.

Flux aléatoire avec produits avec poids et convoyeurs en contre-courant et prise simple

Dans cette partie, outre le cahier des charges initial, de nouvelles données sur les produits et une sur les convoyeurs sont ajoutées. Les produits sont positionnés aléatoirement sur le convoyeur d'entrée avec un poids devant être pris en compte. Comme indiqué dans la section 4.4, les convoyeurs doivent être en contre-courant pour avoir le plus de choix. De plus la zone de travail des robots est décalée de 200 mm dans le sens du convoyeur d'entrée pour les robots finissant de prendre les produits et dans le sens du convoyeur de sortie pour les robots finissant de remplir les boîtes. Finalement, la stratégie de dépose *Linear* n'est pas utilisée.

Lorsque le poids des produits doit être pris en compte, le poids des boîtes est un paramètre important. Lors des simulations, d'autres paramètres vont être donc analysés et comparés. Comme précédent, le pourcentage de produits perdus est reporté. Il y a de plus l'erreur sur le poids moyen, la variance du poids, le poids maximum et le poids minimum des boîtes. L'erreur sur le poids moyen des boîtes est définie par l'équation (5.5) :

$$EAvW = \frac{AvWT - AvW}{AvWT} * 1000 \tag{5.5}$$

où $AvWT$ est le poids théorique souhaité pour les boîtes et AvW le poids moyen des

boîtes mesuré. La variance du poids est définie par l'équation (5.6) :

$$Var = \frac{\sqrt{\sum_{k=1}^{NB}(AvWT - AvW_k)^2}}{NB} \tag{5.6}$$

où NB est le nombre de boîtes, $AvWT$ est le poids théorique souhaité pour les boîtes et AvW_k est le poids de la $k^{ième}$ boîte. L'objectif souhaité est d'avoir le poids des boîtes le plus proche du poids théorique souhaité. Donc le AvW doit être le plus proche de $AvWT$ et la variance la plus petite possible.

Dans un premier temps le système étudié est sous-dimensionné. Pour ce test, la stratégie *PlacingPriority* n'est pas utilisée afin de limiter les contraintes sur le système. Le nombre de boîtes non remplies est donc présent. Le système étant sous-dimensionné, tous les robots fonctionnent à 100%, l'équilibre de la charge de travail n'a pas lieu d'être étudiée. Lors de la comparaison entre les stratégies de prise, aucune stratégie de dépose n'est utilisée.

Stratégie de prise	**Produits perdus (%)**	**EAvW (‰)**	**Var (g)**	**Wmax (g)**	**Wmin (g)**
WeightMean	21.9	-0.03	3.92	996	885
WeightAdjust	22.9	-0.20	4.24	1002	896
WeightAdjustLast	25.1	0.20	6.29	1001	879

TABLE 5.37 – Résultats des stratégies de collaboration pour un système sous-dimensionné ayant des convoyeurs en contre-courant, avec un flux de produits avec poids positionnés aléatoirement et une prise simple.

Le Tableau 5.37 présente les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système sous-dimensionné ayant des convoyeurs en co-courant, avec un flux de produits avec poids positionnés aléatoirement et une prise simple. Tout d'abord nous pouvons noter que la pire règle est *WeightAdjustLast*. En effet, elle perd plus de produits (25.1 %) et a une variance plus élevée (6.29 g). La règle *WeightMean* est légèrement meilleure que la règle *WeightAdjust*. Elle permet de perdre moins de produits (21.9 % pour 22.9 %) et conduit à une erreur sur le poids moyen plus petite (EAvW = -0.03 ‰pour -0.20 ‰) et une meilleure variance (3.92 g pour 4.24 g).

Dans un second temps le système étudié est surdimensionné.

Stratégie de prise	**Produits perdus (%)**	**EAvW (‰)**	**Var (g)**	**Wmax (g)**	**Wmin (g)**
WeightMean	0.00	73.0	5.50	1015	880
WeightAdjust	0.00	68.1	1.62	906	878
WeightAdjustLast	0.11	55.4	4.73	921	878

TABLE 5.38 – Résultats des stratégies de collaboration pour un système surdimensionné ayant des convoyeurs en contre-courant, avec un flux de produits avec poids positionnés aléatoirement et une prise simple.

Le Tableau 5.38 présente les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système surdimensionné ayant des convoyeurs en co-courant,

avec un flux de produits avec poids positionnés aléatoirement et une prise simple. Nous remarquons que la règle *WeightAdjustLast* est la pire car elle perd des produits. La meilleure règle est *WeightAdjust* car elle a une erreur sur le poids moyen la plus petite (EAvW = 68.1 ‰pour 73.0 ‰), elle possède une meilleure variance (1.62 g pour 5.50 g) et elle ne perd pas de produits.

Dans un troisième temps le système étudié est dimensionné au plus juste.

Stratégie de prise	**Produits perdus (%)**	**EAvW (‰)**	**Var (g)**	**Wmax (g)**	**Wmin (g)**
WeightMean	1.02	0.25	4.97	1001	884
WeightAdjust	2.81	-0.03	1.69	999	892
WeightAdjustLast	6.48	0.31	5.07	100	880

TABLE 5.39 – Résultats des stratégies de collaboration pour un système dimensionné au plus juste ayant des convoyeurs en contre-courant, avec un flux de produits avec poids positionnés aléatoirement et une prise simple.

Le Tableau 5.39 présente les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits avec poids positionnés aléatoirement et une prise simple. Tout d'abord, nous notons que comme pour une application ayant des convoyeurs en contre-courants avec des produits simples, il y a une perte de produits. Cependant celle-ci est plus importante avec des produits avec poids. Ceci est dû à la recherche du produit avec le bon poids. Cette recherche peut entrainer des déplacements plus grands afin de prendre le bon produit. Nous remarquons que comme pour le système sous-dimensionné, la règle *WeightAdjustLast* est la pire. La règle *WeightMean* permet de perdre moins de produits que la règle *WeightAdjust* mais possède une moins bonne erreur sur le poids moyen (EAvW = 0.25 ‰contre -0.03 ‰) et une moins bonne variance (4.97 ‰contre 1.69 ‰).

Pour une application où le poids des produits doit être pris en compte afin de remplir correctement les boîtes, il est préférable d'utiliser la règle *WeightAdjust* qui permet d'avoir une erreur sur le poids moyen faible ainsi qu'une petite variance.

Flux aléatoire avec produits avec type et convoyeurs en contre-courant et prise simple

Dans cette partie, outre le cahier des charges initial, de nouvelles données sur les produits et une sur les convoyeurs sont ajoutées. Les produits sont positionnés aléatoirement sur le convoyeur d'entrée avec un type devant être pris en compte. Comme indiqué dans la section 4.5, les convoyeurs doivent en être contre-courant pour avoir le plus de choix. De plus la zone de travail des robots est décalée de 200 mm dans le sens du convoyeur d'entrée pour les robots finissant de prendre les produits et dans le sens du convoyeur de sortie pour les robots finissant de remplir les boîtes.

Durant cette partie, deux règles de planifications individuelles vont être utilisées : *SearchProductType* et *SearchPlaceType*. Ces règles doivent être utilisées ensemble. Plus précisément certains robots doivent utiliser la première et les autres la deuxième. Si le robot le plus en aval du convoyeur de sortie utilise la règle *SearchPlaceType*, il peut

arriver rapidement que les places restantes dans la dernière boîte ne soient pas du type du produit tenu par le robot. Si la règle *PlacingPriority* est utilisée, la ligne de production est totalement arrêtée, sinon des boîtes sont perdues. De même, si le robot le plus en aval du convoyeur d'entrée utilise la règle *SearchPickType*, il peut arriver que les produits restant ne soient pas du type de la place voulue. Ce phénomène entraîne des pertes de produits le temps qu'un produit avec le bon type arrive. Les robots doivent donc avoir différentes règles. Les robots finissant de prendre les produits utilisent la règle *SearchPlaceType* et les robots finissant de remplir les boîtes utilisent la règle *SearchProductType*. Cependant ces règles suivent la même logique que pour des produits simples. Les robots finissant de remplir les boîtes vont chercher, pour la boîte la plus en aval du convoyeur de sortie (équivalent à la règle *FIFO* pour la dépose), le produit correspondant au type recherché le plus en amont du convoyeur d'entrée (équivalent à la règle *LIFO* pour la prise). Le résultat inverse est obtenu pour les robots finissant de prendre les produits.

Dimensionnement	Produits perdus (%)	BW (%)	TW (%)	APP (ms)
Sous-dimensionné	20.9	1.17	390	379
Dimensionné au plus juste	0.02	0.81	390	383
Surdimensionné	0.00	75.5	315	390

TABLE 5.40 – Résultats des stratégies de collaboration pour un système suivant différent dimensionnement ayant des convoyeurs en contre-courant, avec un flux de produits avec type positionnés aléatoirement et une prise simple.

Le Tableau 5.40 montre les résultats de simulation où l'on compare différentes stratégies de collaboration sur un système ayant des convoyeurs en co-courant, avec un flux de produits avec type positionnés aléatoirement et une prise simple suivant différents dimensionnements. Tout d'abord nous pouvons remarquer que les résultats suivent la même tendance que pour des produits simples. Il y a une légère perte de produits pour un système dimensionné au plus juste et aucune perte pour un système surdimensionné. Cependant, pour ce dernier système, le déséquilibre de la charge de travail entre les robots est plus forte (BW = 75.5 % contre 62.5%). De plus, la charge de travail totale est légèrement inférieure pour tous les systèmes (TW= 390% et 315% pour 392% et 329%).

Il est à noter que lorsque le système est sous-dimensionné, le temps d'attente des robots utilisant la règle *SearchPlaceType* pour la recherche de places est toujours de 8 ms, tandis que pour les robots utilisant la règle *SearchPickType* il augmente jusqu'à 12 ms. Cette augmentation est due à la règle *SearchPickType* qui est une combinaison de deux règles : la recherche d'une boîte puis la recherche d'un produit.

5.5 Conclusion sur l'étude comparative en simulation

Dans ce chapitre, nous avons vu la dernière étape de la partie simulation du cycle de développement de la Figure 5.1. Cette étape consistait à tester les différents algorithmes trouvés dans la littérature (voir section 2.2) et développés (voir chapitre 4) afin d'une part de vérifier leur fonctionnement et de les corriger s'il y a des erreurs (erreur d'assignation par exemple), et d'une autre part de les comparer pour faire ressortir les meilleurs algorithmes. Cette étude a été séparée en deux, d'un côté, les règles de planification individuelles applicables pour chaque robot indépendamment des autres et de

l'autre, les stratégies de collaboration entre les robots. Pour chaque côté, plus spécialement pour les stratégies, les résultats dépendent de la configuration de l'application (sens des convoyeurs, type de prise, caractéristiques des produits etc.). Il ressort cependant de cette étude plusieurs grandes lignes :

- Les robots devant finir de remplir les boîtes doivent utiliser de préférences la règle *FIFO* pour la dépose et les robots devant finir de prendre les produits la règle *FIFO* pour la prise.
- Lorsque le système est surdimensionné, il est préférable d'utiliser une stratégie de collaboration pour équilibrer la charge de travail entre les robots, afin d'éviter l'usure prématurée d'un robot.
- Pour un système utilisant une simple prise, il n'est pas nécessaire d'utiliser une stratégie de collaboration si le système est dimensionné au plus juste.
- Pour ce type de dimensionnement, si l'application utilise une prise multiple, la stratégie *IntToExt Cyclic* avec la règle *SPT* permet de prendre 1% de produits simples de plus qu'une application n'utilisant aucune stratégie comme dans l'industrie.
- Si l'orientation des produits doit être prise en compte, la stratégie *IntToExt Cyclic Rot* permet de prendre 1.6% de produits de plus qu'une application n'utilisant aucune stratégie de collaboration comme dans l'industrie.
- Un système avec des convoyeurs en contre-courant possède un débit maximum légèrement inférieur Ã un système avec des convoyeurs en co-courant mais permet de perdre moins de produits en cas de perturbation (sur-cadence ou sous-cadence).
- Il est préférable d'utiliser ce genre de système si le poids ou le type des produits doit être pris en compte.

L'utilisation des algorithmes peut être résumé avec le Tableau 5.41.

	Dimensionné au plus juste	**Surdimensionné**	**Prise multiple**	**Contre-courant**	**Produits orientés**
FIFO	X	X	X	X	X
Nothing	X			X	
IntToExt Cyclic		X		X	
IntToExt Cyclic SPT			X		
IntToExt Cyclic Rot					X

TABLE 5.41 – Résumé de l'utilisation des algorithmes.

Des tests complémentaires ont été réalisés dans l'Annexe B.

Ces résultats montrent un des intérêts de l'utilisation d'un tel outil de simulation par les industriels comme Schneider Electric. Il permet d'aider au dimensionnement ainsi qu'au choix de la stratégie de commande suivant un cahier des charges donné au vu du développement d'une nouvelle application industrielle. En effet, comme indiqué dans les grandes lignes, la simulation a permis non seulement de tester la fonctionnalité des algorithmes et ainsi avoir une comparaison entre eux, mais cette étude a aussi donné un premier retour par rapport aux performances de ceux-ci.

La partie simulation étant effectuée, les prochaines étapes sont le transfert des lois de commande vers l'expérimentation (passage de la simulation à l'expérimentation du cycle de développement de la Figure 5.1) ainsi que leur validation sur des démonstrateurs, étape *Validation Algorithmes* du cycle de développement.

Chapitre 6

Étude comparative entre simulation et expérimentation

Sommaire

Jusqu'à maintenant, le développement et les tests n'ont été effectués qu'en simulation. Dans ce chapitre, nous allons effectuer une étude comparative entre, d'un côté, la simulation et d'un autre côté l'expérimentation. Cette étude validera ou invalidera l'utilisation de la simulation et la méthodologie proposée avec le cycle de développement d'une application pick & place de la Figure 6.1.

Dans un premier temps, les différents démonstrateurs et les outils utilisés pour l'expérimentation durant cette étude vont être présentés. Puis dans un second temps, nous allons comparer, pour les mêmes applications en simulation et en expérimentation, les résultats obtenus suivant les différents algorithmes.

6.1 Présentation détaillée des démonstrateurs

Cette thèse étant faite en collaboration avec Schneider Electric, la partie expérimentation est réalisée à l'aide de leurs produits. C'est pour cette raison que la solution PacDrive 3 ainsi que leur outil logiciel SoMachine Motion ont été retenus.

L'offre PacDrive 3 est la solution pour la réalisation des machines de production dans l'emballage, l'assemblage, la manutention et de manière générale pour toutes les applications hautes performances jusqu'à 99 axes. Un exemple d'architecture hardware est présenté sur la Figure 2.1. PacDrive 3 est une approche centralisée et intégrée qui permet de réduire les coûts matériels de configuration et d'ingénierie. Sa structure logicielle modulaire diminue de façon significative le temps de développement des applications les plus

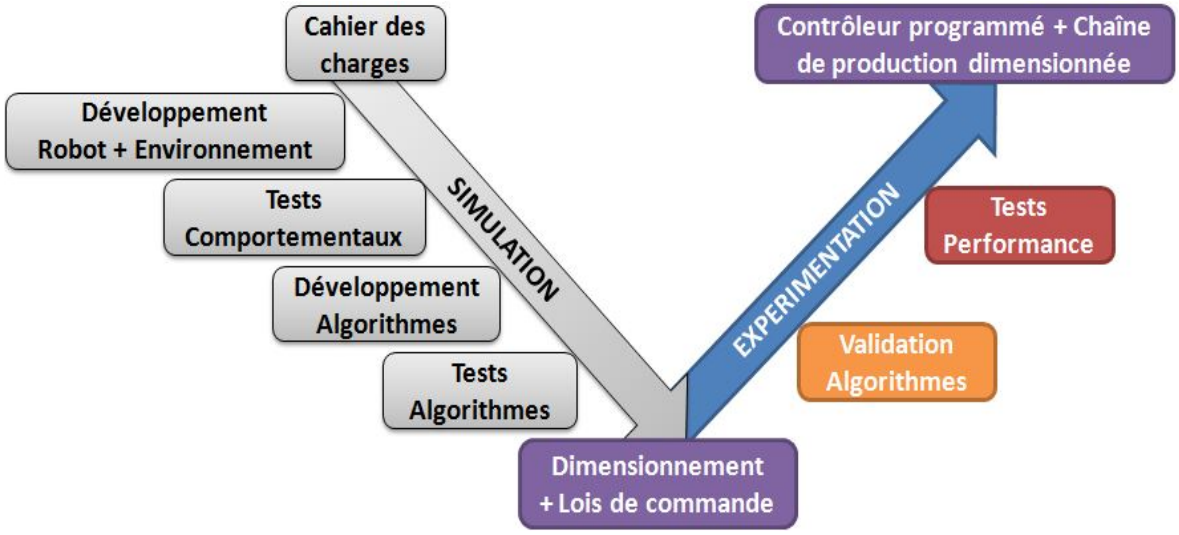

FIGURE 6.1 – Partie expérimentation du cycle de développement d'une application pick & place.

complexes. Une architecture hardware utilisant le PAcDrive 3 est composée de plusieurs éléments :

- Le LMC 600C (Logic Motion Controller) est un contrôleur de mouvement logique de 99 axes synchronisés en temps réel, avec des entrées et sorties numériques et analogiques intégrées. Le contrôleur intègre à la fois des entrées/sorties standards et des entrées rapides. Il a la possibilité de simuler des axes ou des robots. Cela permet de tester et de débugger un programme avant une mise en service.
- Le réseau SERCOS III est un bus de terrain utilisé pour la commande des variateurs et des axes ainsi que pour la sécurité. Chaque contrôleur dispose d'un réseau Ethernet TCP/IP pour l'intégration dans le réseau usine.
- Le module servo-variateur Lexium LXM62 est une solution mono axe ou multi axes fournissant les courants de phases nécessaires pour le contrôle de la position des servomoteurs SH3 connectés.
- Le servo-module ILM62 combine un servo-variateur et un servomoteur. Il permet de gagner de la place dans l'armoire électrique où il n'y a besoin que du contrôleur et du bloc d'alimentation.
- Le SLC (Safety Logic Controller) gère les tâches centrales relatives à la sécurité dans une application.
- Le TM5 I/O est le bloc d'entrées/sorties. Il est composé d'une interface de bus de terrain et des modules d'entrées sorties analogiques ou digitales peuvent être ajoutés.
- Le Magelis XBT GT est un terminal graphique IHM (Interface Homme / Machine) à écran tactile.
- Le XUB0APSNM12 est un capteur laser détectant la présence ou l'absence d'un produit.

SoMachine Motion est un logiciel conçu spécifiquement pour les constructeurs de ma-

chines OEM (Original Equipment Manufacturer). Il fournit un outil commun permettant de développer, configurer, programmer et mettre en service les systèmes de contrôle des machines d'une entreprise. SoMachine Motion rassemble en un seul programme des outils pour le dimensionnement des entraînements et la conception de profils de mouvements (ECAM), le développement de programmes (EPAS avec ETEST, Vijeo Designer), le diagnostic (Diagnostics) et la gestion de données (Assistants). Pour le développement des automatismes de sécurité, un éditeur de sécurité (Safety Editor) est intégré dans SoMachine Motion. EDESIGN est l'élément central pour une nouvelle forme de structuration graphique des fonctions machines, et représente un pas supplémentaire vers la réduction de la complexité lors de la conception de programmes.

Comme indiqué dans la section 3.5, durant la thèse deux démonstrateurs ont été utilisés. Le premier est situé à l'AIP RAO à l'INSA de Lyon et est fourni par Schneider Electric, voir Figure 3.12. Sa composition est détaillée avec les Figures 6.2 et 6.3.

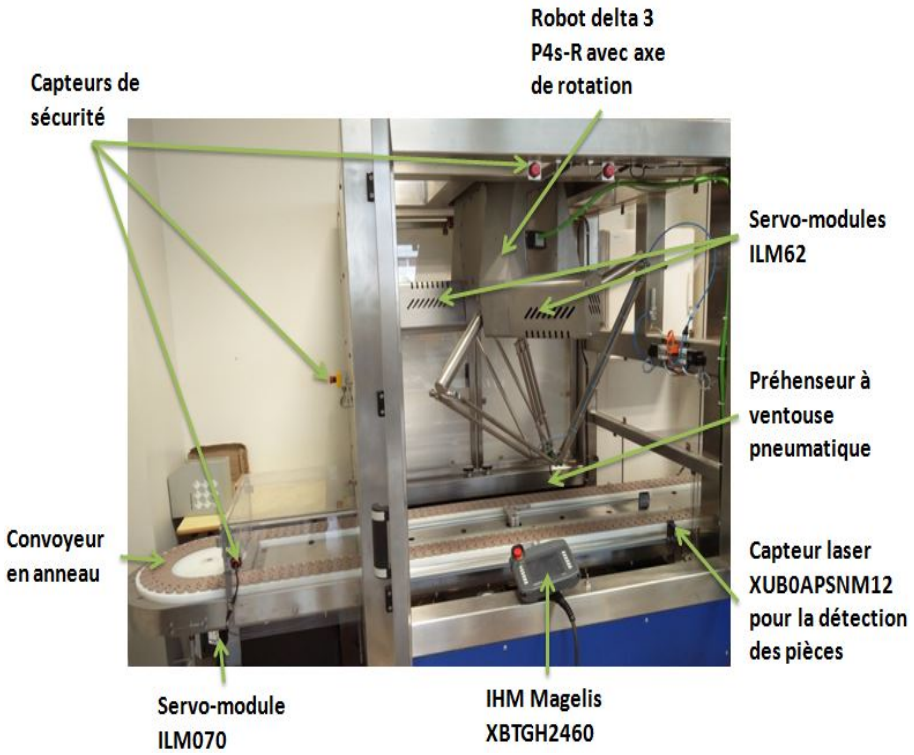

FIGURE 6.2 – Démonstrateur avec un robot et un convoyeur en anneau - Site INSA de Lyon : AIP RAO - France.

Le second démonstrateur est situé sur le site de Schneider Electric à Marktheidenfeld en Allemagne. Il est composé de trois robots delta 3 P4s-F Schneider Electric pour le pick & place avec un préhenseur à ventouse pneumatique sans axe de rotation sur l'organe terminal, de deux convoyeurs parallèles indépendants avec une caméra pour la détection de produits et d'un convoyeur de retour pour réinjecter les produits non-pris. Il est commandé par un contrôleur Schneider Electric LMC600C et est programmé par SoMachine Motion, voir Figure 3.13. Il possède plusieurs IHM Magelis ainsi qu'un contrôleur de sécurité.

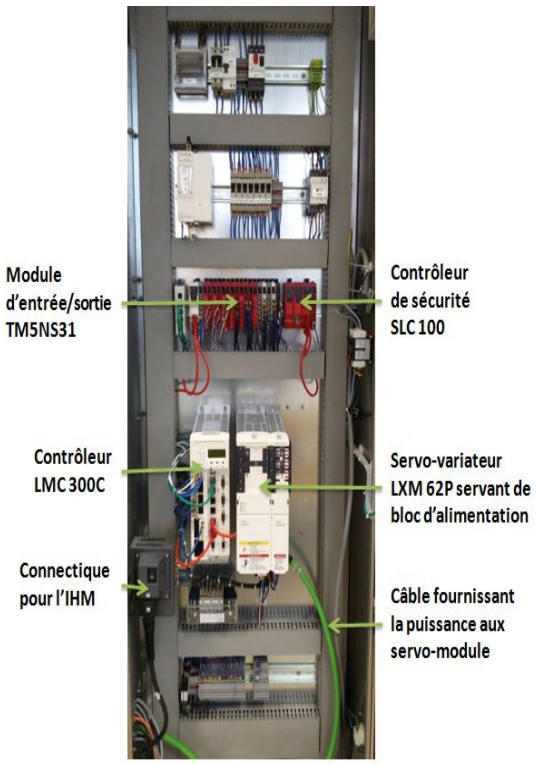

FIGURE 6.3 – Armoire électrique du démonstrateur à l'INSA.

6.2 Présentation de la méthodologie de tests

L'étude suit entièrement la démarche du cycle de développement d'une application pick & place de la Figure 6.1 et ceci pour chaque démonstrateur. Tout d'abord, il y a la réalisation des démonstrateurs en simulation, étapes *Développement Robot et Environnement* et *Tests Comportementaux* du cycle. Les plugins permettant de donner les comportements des composants d'une application pick & place étant déjà développés dans la section 3.4, seul le prototypage est à réaliser (description détaillée fonctionnelle et géométrique des composants). Cette opération prend en compte le nombre, la taille, la disposition et les performances des robots et des convoyeurs. Ces paramètres seront détaillés dans la section décrivant chaque démonstrateur.

Les étapes suivantes concernent les algorithmes. Du point de vue simulation, l'étape *Développement Algorithmes* du cycle étant décrite dans le chapitre 4, les algorithmes ont simplement besoin d'être paramétrés suivant les caractéristiques des démonstrateurs. En effet, pour des applications différentes (par exemple le nombre de robots ainsi que la position des convoyeurs sont modifiés), il n'y a pas besoin de refaire un développement des algorithmes. Seuls les paramètres d'entrée des blocs fonctionnels sont à adapter, voir Figure 6.4. Cela permet d'effectuer seulement l'étape *Tests Algorithmes* suivant les deux configurations des démonstrateurs et donc suivant d'autres applications si le besoin s'en fait ressentir.

Á ce niveau toute la partie simulation a été réalisée. Il faut donc effectuer la transition entre la simulation et l'expérimentation. Cette transition a été expliquée dans la section 3.5 où a été présenté le parallélisme entre les architectures de la simulation et

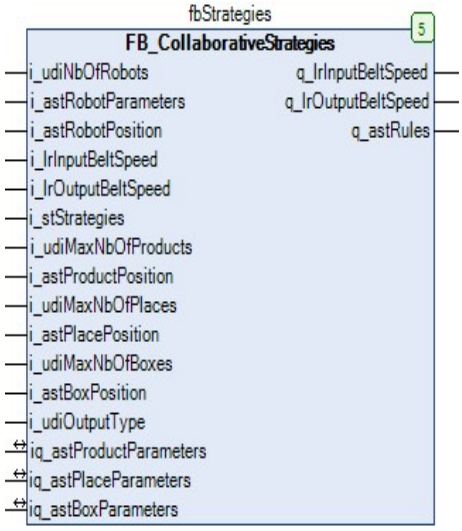

FIGURE 6.4 – Bloc fonctionnel des stratégies de collaboration.

du contrôleur. Les algorithmes souhaités peuvent donc être traduits et mis en place dans les blocs fonctionnels adéquats : règles de planification individuelles ou stratégies de collaboration. L'avantage de ce parallélisme est de retranscrire, algorithme par algorithme, le programme de la partie simulation dans les blocs fonctionnels du contrôleur. En effet, les algorithmes peuvent être considérés comme des fonctions à part entière permettant de simplement traduire ces fonctions sans avoir à ajouter une couche de programmation afin de les mettre en œuvre. Cependant, dans l'état actuel, le seul inconvénient durant ce passage est la nécessité de modifier la syntaxe du programme. Les outils logiciel $SOSi^{TM}$ et *SoMachine Motion* utilisent tous deux un langage orienté objet, mais chacun possède leur propre syntaxe. Un exemple d'une règle de planification individuelle programmée dans les deux langages est présenté dans l'Annexe A. Un travail futur serait de rajouter une extension à $SOSi^{TM}$, par exemple, afin de rendre cette traduction automatique.

En plus de permettre d'avoir un gain de temps pour la transition entre la simulation et l'expérimentation, les blocs fonctionnels permettent de gagner du temps lors de la phase de tests. En effet, tous les algorithmes souhaités peuvent être implémentés dans les blocs fonctionnels. C'est-à-dire que le contrôleur contient tous les algorithmes. Cela permet donc de changer d'algorithmes en ligne sans avoir besoin de recompiler le programme. Donc lorsque l'on désire tester plusieurs stratégies de collaboration par exemple, toutes les stratégies peuvent être essayées les unes à la suite des autres durant un temps donné en un seul test. Cela sera fait afin de réaliser les étapes *Validation Algorithmes* et *Tests Performance* du cycle de développement.

Comme indiqué dans la section précédente, deux démonstrateurs ont pu être utilisés durant cette thèse. L'étude sera donc séparée en deux parties. La première sera faite avec le démonstrateur avec un seul robot et la deuxième avec le démonstrateur avec trois robots. Ces démonstrateurs permettront de tester différents paramètres qui seront vus ultérieurement.

6.3 Résultats comparatifs entre simulation et expérimentation

6.3.1 Étude avec un démonstrateur un robot

La première partie de l'étude est réalisée à l'aide du démonstrateur avec un robot. Cette partie a pour objectif, premièrement de vérifier le bon fonctionnement de l'architecture du contrôleur de la Figure 3.11. Plus précisément, elle vérifie l'interaction entre les nouveaux blocs fonctionnels règles individuelles et stratégies collaboratives avec le programme automate déjà existant. Deuxièmement, cette partie a pour but de vérifier la similitude entre la simulation et l'expérimentation. Pour cela le temps de prise et de dépose en seconde ainsi que le nombre de produits sont comparés. De plus une erreur est calculée via l'expression (6.1) :

$$Erreur = |\frac{Expérimentation - Simulation}{Expérimentation}| * 100 \tag{6.1}$$

Finalement, comme l'utilisation d'un unique robot implique l'impossibilité d'utiliser des stratégies de collaboration, seules les règles de planification individuelles suivantes sont utilisées et comparées : la règle *FIFO*, la règle *LIFO* et la règle *SPT*.

Les résultats des Tableaux 6.1 et 6.2 sont obtenus avec le cahier des charges suivant :

- La vitesse du convoyeur est choisie égale à 0.1m/s. Le convoyeur étant en anneau, le convoyeur d'entrée et de sortie sont en contre-courant.
- La vitesse maximum de l'extrémité de l'effecteur est choisie égale à 1.2 m/s. Son accélération est égale à 20 m/s^2.
- Pour des facilités de reproduction, la trajectoire de l'extrémité de l'effecteur est la suivante : mouvement linéaire jusqu'au dessus de la cible, descente puis montée linéaire.
- La zone de travail des robots est un disque de diamètre 600 mm.
- Les produits dans le flux d'entrée sont positionnés en ligne espacés de 50 mm.
- Les boites dans le flux de sortie sont positionnées en ligne espacées de 200 mm et peuvent contenir deux produits.
- La logique de prise est la règle *FIFO*. La logique de dépose est la règle *LIFO*.
- Le temps pour chaque essai est fixé à 10 minutes en simulation et en expérimentation.
- L'étude est effectuée en régime permanent.

Les différents tests (Cahier des charges dans les Tableaux 6.1 et 6.2) sont effectués avec les conditions suivantes :

1. Cahier des charges nominal.
2. Cahier des charges nominal avec vitesse convoyeur : 0.02 m/s, vitesse robot : 0.3 m/s.
3. Cahier des charges nominal avec la logique de prise : règle *LIFO*.
4. Cahier des charges nominal avec la logique de prise : règle *SPT*.

Les résultats sont regroupés selon l'affichage : résultats de simulation / résultats d'expérimentation.

Le Tableau 6.1 montre les résultats de simulation et d'expérimentation suivant les différents cahiers des charges. Tout d'abord nous remarquons que l'on obtient les meilleurs résultats avec la règle *FIFO* en logique de prise (0.759 s pour la prise et 0.756 s pour la dépose en simulation). En effet les temps de prise et de dépose sont les plus faibles ce qui entraine un nombre de produits pris plus élevé. La règle *LIFO* en logique de prise est la pire (0.873 s pour la prise et 0.951 s pour la dépose en simulation). Ceci est dû à la configuration des convoyeurs. Comme expliqué dans la section 5.3, si les convoyeurs sont en contre-courant, les logiques de prise et de dépose doivent être inversées : si la règle *LIFO* est utilisée pour la dépose, la règle *FIFO* doit être utilisée pour la prise afin de diminuer la distance de déplacement sinon la distance de déplacement est plus grande, voir Figure 5.4. Pour des convoyeurs en co-courant, les logiques de prise et de dépose doivent être les mêmes.

Cahier des charges	Temps de prise (s)	Temps de dépose (s)	Nombre de produits pris
1	**0.759** / 0.748	**0.756** / 0.762	319 / 298
2	2.872 / 2.866	2.864 / 2.900	91 / 88
3	**0.873** / 0.862	**0.951** / 0.976	284 / 282
4	0.822 / 0.812	0.824 / 0.826	298 / 296

TABLE 6.1 – Résultats de simulation / expérimentation avec un robot.

Le Tableau 6.2 présente l'erreur entre les résultats de simulation et d'expérimentation. Pour toutes les hypothèses de tests, l'erreur entre simulation et expérimentation est très faible. Cela montre l'intérêt de l'outil de simulation pour tester et améliorer la performance d'applications pick & place multi-robots. En effet, comme l'erreur est faible, des applications pick & place peuvent être testées, corrigées et améliorées avant la mise en service. Ce qui permet de gagner du temps si cette partie est faite en même temps que la construction de l'application. De plus, certains problèmes pouvant être rencontrés durant la mise en service peuvent être évités (collision des robots, produits non accessibles etc.).

Cahier des charges	Erreur sur le temps de prise (%)	Erreur sur le temps de dépose (%)	Erreur sur le nombre de produits pris (%)
1	1.47	0.79	7.05
2	0.21	1.24	3.41
3	1.28	2.56	0.71
4	1.23	0.24	0.68

TABLE 6.2 – Erreur entre les résultats de simulation / expérimentation.

Cette première partie d'étude a permis d'une part de valider l'architecture du contrôleur en vérifiant son bon fonctionnement et d'une autre part de vérifier la fiabilité de la simulation par rapport à l'expérimentation. En plus des deux points cités, ce démonstrateur a permis la prise en main de l'outil logiciel SoMachine Motion ainsi que l'environnement matériel Schneider Electric.

6.3.2 Étude avec un démonstrateur trois robots

La deuxième partie de l'étude est réalisée à l'aide du démonstrateur trois robots. Cette partie a pour objectif de vérifier le bon fonctionnement et les performances des stratégies de collaboration. Différents paramètres sont analysés et comparés comme le pourcentage de produits perdus, de boîtes non remplies, l'équilibre de la charge de travail des robots et la charge de travail totale du système. Les différents calculs ont été expliqués dans la Section 5.2.

Les résultats des Tableaux 6.3 et 6.4 sont obtenus avec le cahier des charges suivant :

- Tous les robots fonctionnent avec une vitesse maximum de 5 m/s et une accélération de 50 m/s^2.
- Les robots prennent et déposent un seul produit à la fois.
- Le premier robot a une zone de prise et dépose large de 230 mm, le deuxième a une zone de prise de 450 mm et une zone de dépose de 600 mm, le troisième robot a une zone de prise de 400 mm et une zone de dépose de 550 mm.
- Un cycle de pick & place pour un robot prend en moyenne 0.74 secondes. Le robot peut prendre $PPas = 1/0.74 = 1.35$ produits par seconde.
- Les logiques de prise et de dépose suivent la règle *FIFO*.
- Les convoyeurs sont positionnés en parallèle avec des flux dans le même sens.
- Les produits dans le flux d'entrée sont positionnés aléatoirement.
- Les boîtes mesurent 0.13 m * 0.13 m et contiennent 9 produits ($NPb = 9$).
- La vitesse du convoyeur d'entrée est choisie arbitrairement égale à 0.2 m/s ($Vi = 0.2$).
- La distance entre le centre des boîtes est choisie arbitrairement égale à 0.15 m ($EB = 0.15$).
- Le temps de simulation pour chaque essai est fixé à 10 minutes.
- L'étude est effectuée en régime permanent.

Le démonstrateur est composé de plusieurs robots, l'impact de la présence ou non de stratégies de collaboration peut donc être étudié. Les stratégies comparées sont *Nothing* (pas de stratégie appliquée), *IntToExt Cyclic* et *Linear*. En parallèle, une étude a été menée pour comparer l'influence de la stratégie *PlacingPriority*. Finalement, une répartition de la charge de travail entre les robots suivant le type de stratégies et la marge de sécurité est présentée.

Une marge de sécurité est appliquée au débit d'entrée afin de surdimensionner le système. L'influence de cette marge de sécurité est aussi étudiée. Le débit d'entrée et la vitesse de sortie sont calculés selon la méthode décrite dans la section 5.1. Voici un exemple pour une marge de sécurité de 10%. Un débit de produits est choisi $FPs = 3.64$ produits par seconde. Pour surdimensionner le système, une marge de sécurité $Mfp = 10\%$ est ajoutée. Le nombre minimum de robots est $NRmin = FPs * (1 + Mfp)/PPas = 3.64 * (1 + 0.1)/1.35 = 2.96$ arrondie à 3. Comme une marge de sécurité a été utilisée, le nombre final de robots est $NRf = NRmin = 3$. En s'appuyant sur le nombre de produits dans une boîte, le débit de boites est égal à $FBs = FPs/NPb = 3.64/9 = 0.404$ boites par secondes. En utilisant la distance entre les boites, la vitesse du convoyeur de sortie doit être égale à $Vo = FBs * EB = 0.404 * 0.15 = 0.061$ m/s. L'étude comparative a été effectuée pour une marge de 2.5%, de 6.25%, de 10% et de 15%. On obtient donc respectivement les débits d'entrée suivantes : 3.95 produits/s, 3.80 produit/s, 3.64 produits/s et 3.44

produits/s et les vitesses de sortie : 0.066 m/s, 0.063 m/s, 0.061 m/s et 0.057 m/s.

Le Tableau 6.3 présente les résultats de simulation et d'expérimentation en fonction de différentes stratégies et suivant différentes marges de sécurité sans la stratégie *Placing-Priority*. Tout d'abord nous remarquons que les résultats entre la simulation et l'expérimentation sont proches et suivent les mêmes tendances. Il y a une différence entre les résultats car les mouvements de l'extrémité de l'effecteur du robot ne sont pas identiques pour ce démonstrateur. Le démonstrateur actuel avec son programme existait déjà avant que la simulation ne soit créée. Les trajectoires utilisées dans le démonstrateur n'étaient pas encore implémentées dans la simulation. Cependant, la simulation a été faite de telle sorte que les temps moyens de prise et de dépose soient très proches. Les temps moyens de prise et de dépose ont été mesurés, puis réinjectées dans le modèle de simulation. Cela démontre la possibilité d'une mise au point du modèle de simulation grâce aux résultats expérimentaux.

De plus, le démonstrateur étant un outil de démonstration multi-usages, il possède des sécurités intrinsèques qui peuvent ralentir la prise, réduire la zone de travail, etc. Ces sécurités sont difficiles à retranscrire en simulation. Pour les tests de la section 6.3.1, le programme du contrôleur a été réalisé en parallèle du programme en simulation. Les mouvements de l'extrémité de l'effecteur du robot étaient identiques et donc les résultats étaient très proches.

Marge de sécurité	**Stratégies**	**Produits perdus (%)**	**Boîtes non remplies (%)**	**BW (%)**	**TW (%)**
2.5 %	*Nothing*	1.57 / 1.36	20.2 / 18.3	0.05 / 0.11	280 / 280
	IntToExt Cyclic	2.17 / 1.74	24.3 / 20.2	0.13 / 0.20	280 / 280
	Linear	1.40 / **1.36**	17.5 / **15.2**	0.06 / 0.01	280 / 280
6.25 %	*Nothing*	0.00 / 0.00	0.00 / 0.00	7.62 / 2.51	272 / 277
	IntToExt Cyclic	0.00 / **0.13**	0.00 / 0.00	2.88 / 1.26	278 / 279
	Linear	0.00 / 0.04	0.00 / 0.00	1.18 / 1.67	279 / 279
10 %	*Nothing*	0.00 / 0.00	11.9 / **11.1**	14.9 / **14.9**	266 / **265**
	IntToExt Cyclic	0.00 / 0.00	11.1 / 9.91	3.54 / 6.91	279 / 274
	Linear	0.00 / 0.00	11.4 / **10.3**	3.23 / **9.33**	278 / **272**
15 %	*Nothing*	0.00 / 0.00	9.57 / 9.61	31.4 / 33.6	249 / 247
	IntToExt Cyclic	0.00 / 0.00	8.70 / 7.45	13.3 / 10.2	269 / 259
	Linear	0.00 / 0.00	9.52 / 6.55	13.9 / 9.92	268 / 258

TABLE 6.3 – Résultats des stratégies de collaboration suivant différentes marge de sécurité pour la simulation et l'expérimentation sans la stratégie *PlacingPriority* pour trois robots.

Nous remarquons que pour une marge de sécurité supérieure ou égale à 6.25% il n'y a pas ou très peu de pertes de produits (inférieur à 0.13%). Ceci est dû au fait qu'il y a un temps de calcul nécessaire aux robots pour prendre la décision de prendre ou de déposer un produit. En dessous de ce seuil, le système est sous-dimensionné, au-dessus il est surdimensionné. De plus, au-delà de ce seuil il y a encore la présence de boîtes non remplies, ce qui est dû aux erreurs d'arrondi dans les calculs. En effet, les arrondis ne permettent pas d'avoir exactement le nombre de produits égale aux nombre de places.

Nous pouvons ajouter que pour un système surdimensionné, les stratégies de collaboration permettent de remplir plus de boîtes qu'en absence de stratégie (pertes inférieures

à 10.3% contre 11.1% en expérimentation pour une marge de 10%). De plus il y a un meilleur équilibre de la charge de travail entre les robots (BW inférieur à 9.33% contre 14.9% en expérimentation pour une marge de 10%). Cependant, la charge de travail totale est augmentée (TW supérieur à 271.7% contre 265% en expérimentation pour une marge de 10%). Pour un système sous-dimensionné, ce qui peut arriver lors d'une augmentation de la cadence, la stratégie *Linear* permet de perdre moins de produits et de boîtes (respectivement 1.36% et 15.2% en expérimentation). Il s'agit de résultats que nous avons retrouvés dans la section 5.4.

La Figure 6.5 montre la répartition de la charge de travail entre les robots suivant les différentes stratégies et en fonction de la marge de sécurité. Lorsqu'il n'y a pas de stratégie utilisée, les premiers robots fonctionnent à 100% quelle que soit la marge de sécurité, tandis que la charge de travail du dernier robot va diminuer lorsque le seuil entre sous-dimensionné et surdimensionné est dépassé. Avec une stratégie de collaboration, la charge de travail de tous les robots diminue. Cependant, la charge de travail du deuxième robot diminuera après celle des autres robots.

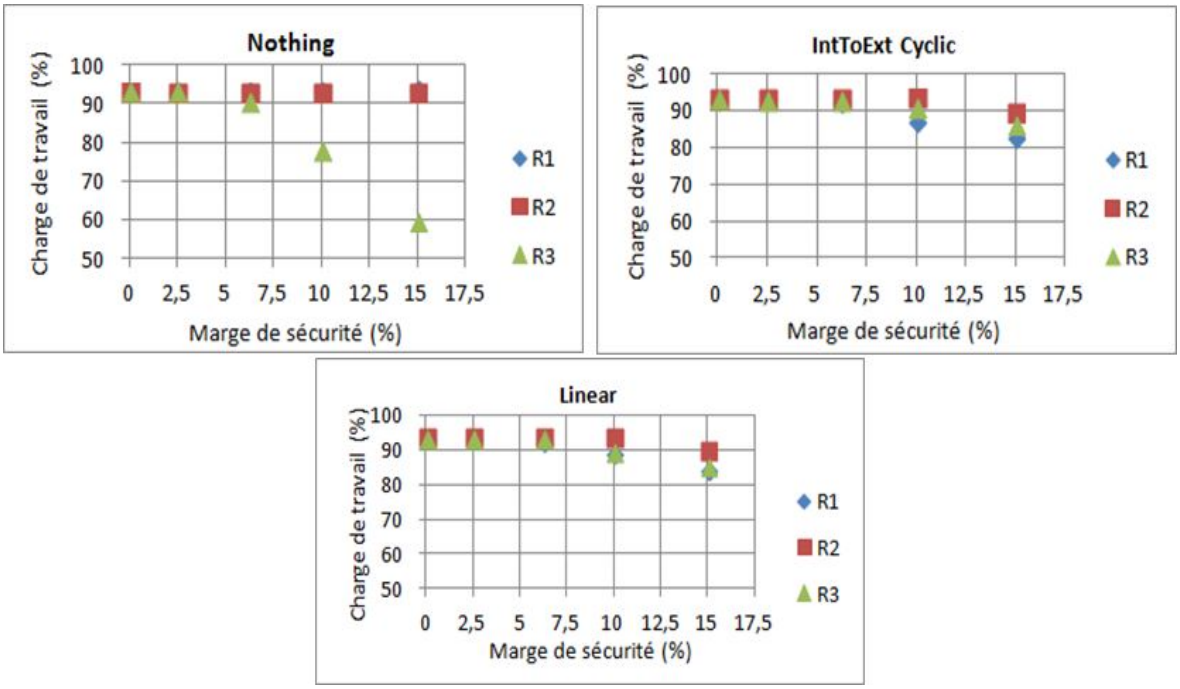

FIGURE 6.5 – Répartition de la charge de travail sur trois robots suivant la marge de sécurité.

Le Tableau 6.4 présente les résultats expérimentaux en fonction de différentes stratégies et suivant différentes marges de sécurité en utilisant la logique de priorité à la dépose. Tout d'abord nous remarquons que la règle de priorité à la dépose permet de ne pas avoir de boîtes non remplies. Pour un système sous-dimensionné, cela n'augmente que très légèrement le nombre de produits perdus (1.46% contre 1.36% en expérimentation avec la stratégie *Linear*). Tandis que pour un système surdimensionné, le nombre de produits perdus est toujours nul. La contrainte qu'ajoute la règle de priorité à la dépose, est une légère diminution de l'équilibre de la charge de travail entre les robots (BW = 9.42 %

contre 9.33% en expérimentation avec la stratégie *Linear* et une marge de 10%).

Marge de sécurité	Stratégies	Produits perdus (%)	Boîtes non remplies (%)	BW (%)	TW (%)
2.5 %	*Nothing*	2.08 / 1.41	0.00 / 0.00	0.19 / 0.25	280 / 280
	IntToExt Cyclic	2.59 / 1.77	0.00 / 0.00	0.13 / 0.21	280 / 280
	Linear	4.03 / **1.46**	0.00 / 0.00	0.00 / 0.04	280 / 280
10 %	*Nothing*	0.00 / 0.00	0.00 / 0.00	17.6 / 19.6	262 / 260
	IntToExt Cyclic	0.00 / 0.00	0.00 / 0.00	7.06 / 11.3	274 / 270
	Linear	0.00 / 0.00	0.00 / 0.00	6.04 / **9.42**	275 / 270
15 %	*Nothing*	0.00 / 0.00	0.00 / 0.00	33.8 / 35.8	246 / 244
	IntToExt Cyclic	0.00 / 0.00	0.00 / 0.00	16.9 / 13.1	265 / 256
	Linear	0.00 / 0.00	0.00 / 0.00	15.4 / 10.0	267 / 255

TABLE 6.4 – Résultats des stratégies de collaboration suivant différentes marge de sécurité pour la simulation et l'expérimentation avec la stratégie *PlacingPriority* pour trois robots.

6.4 Conclusion sur l'étude comparative entre simulation et expérimentation

Dans ce chapitre, nous avons réalisé une analyse comparative entre d'un côté des tests en simulation et l'autre côté des tests sur des démonstrateurs réels. Afin d'effectuer cette étude, le cycle de développement d'une application pick & place de la Figure 6.1 a été entièrement suivi : du stade *Cahier des charges* au stade *Contrôleur programmé + Chaîne de production dimensionnée*. Plusieurs étapes ou parties d'étapes ont déjà été achevées dans les sections 3.4 et 3.5 et les chapitres 4 et 5 :

— Les plugins pour le comportement des différents composants.
— L'étape *Développement Algorithme*.
— La similitude entre l'architecture de la simulation et du contrôleur.

Pour ces étapes, seul un paramétrage est nécessaire. Cela octroie un gain de temps non négligeable lorsque des tests sur différentes configurations d'applications doivent être effectués.

Pour réaliser la partie expérimentation, deux démonstrateurs ont été utilisés permettant de vérifier différents paramètres. Le premier démonstrateur possédant un seul robot a permis de vérifier le bon fonctionnement de l'architecture du contrôleur de la Figure 3.11, de vérifier la similitude entre la simulation et l'expérimentation et de comparer les règles de planification individuelles. Le deuxième démonstrateur possédant trois robots a permis de vérifier le bon fonctionnement et les performances des stratégies de collaboration avec le même objectif de montrer le parallélisme entre la simulation et l'expérimentation.

La première partie de l'étude a tout d'abord aidé à prendre en main l'outil logiciel et l'environnement matériel de travail. Elle a permis de montrer expérimentalement l'utilisation inversée des règles de planification individuelles (règle *FIFO* pour la prise et règle *LIFO* pour la dépose par exemple) pour un système avec des convoyeurs en contre-courant. De plus, elle a permis d'indiquer la similitude entre les résultats de simulation et d'expérimentation. Ce parallèle est un avantage de la méthodologie développée au travers de

l'outil logiciel $SOSi^{TM}$. Hormis la différence syntaxique entre les deux outils logiciels, il permet de copier un algorithme depuis un outil puis de le coller dans l'autre. Un outil de traduction peut être envisagé pour réaliser cela.

La deuxième partie de l'étude a permis de conforter les résultats trouvés dans la section 5.4. Les stratégies de collaboration obtiennent un équilibre de la charge de travail entre les robots pour un système surdimensionné, ce qui permet d'éviter l'usure prématurée d'un robot. De plus lorsque que le système est dimensionné au plus juste ou sous-dimensionné et utilise une prise simple, l'utilisation d'une stratégie de collaboration n'est pas nécessaire. Finalement, la stratégie *PlacingPriority* permet d'éviter d'avoir des boîtes non remplies en échange d'une légère diminution de l'équilibre de la charge de travail entre les robots.

Cette deuxième partie a aussi démontré l'intérêt sur site de l'utilisation de cette méthodologie et de cet outil logiciel. Le deuxième démonstrateur se situant en Allemagne et étant utilisé régulièrement par Schneider Electric, sa fenêtre d'exploitation n'était que d'une semaine et son fonctionnement ne devait pas être altéré. Or, l'outil logiciel a permis de préparer en amont les différents tests, et le parallélisme de l'architecture a permis le fonctionnement du démonstrateur avec les blocs fonctionnels développés dès le premier jour. De plus, tous les algorithmes étant implémentés dans les blocs fonctionnels, ils peuvent être changés en ligne sans avoir besoin de recompiler le programme. Les stratégies peuvent donc être essayées les unes à la suite des autres durant un temps donné en un seul test.

Chapitre 7

Conclusion générale et perspectives

Conclusion

Les travaux de cette thèse s'inscrivent dans le contexte de l'emballage et du conditionnement. Plus particulièrement dans la partie chaîne de production permettant de réaliser ce contexte. Les robots et les modules robotisés de pick & place hautes performances sont de plus en plus présents dans l'industrie manufacturière afin de répondre à la problématique d'augmentation de la productivité. Afin d'améliorer les performances de ces applications, il est nécessaire de mieux maîtriser le dimensionnement du système de production tout en améliorant la gestion des flux et la gestion de la charge de travail entre les robots lorsque plusieurs sont utilisés. Nous avons présenté dans le chapitre 2, dans un premier temps, un état de l'art des cellules multi-robots pick & place afin d'expliquer la complexité et la diversité de ces applications et l'importance d'un cahier des charges détaillé. Dans un second temps, une présentation non-exhaustive d'algorithmes utilisés pour ce genre d'application a été faite. Ces algorithmes peuvent être séparés en deux catégories : les règles de planification individuelles et les heuristiques. Cependant ces derniers demandent un temps de calcul important incompatible avec une mise en œuvre en temps réel.

Afin de dimensionner et de mettre en œuvre des applications de pick & place, les industriels s'appuient souvent sur leurs expériences du terrain. Ils n'utilisent aucun outil d'aide au dimensionnement ou à l'analyse de performances. Nous avons détaillé à l'aide de la Figure 3.1, le cycle de développement d'une application pick & place. À notre connaissance, dans le monde industriel et académique, il n'existe pas d'outil de simulation qui prenne en compte toutes les étapes du cycle de développement. Il existe des outils performants répondant à un seul besoin mais aucun logiciel répondant à tous. Il est nécessaire qu'à chaque étape, une prise en main et un développement doivent être de nouveau effectués. Cependant nous avons indiqué que les industriels ont commencé à acquérir de nouveaux savoir-faire afin de réaliser d'autres étapes du cycle de développement autres que la programmation de contrôleur. Cette tendance est accompagnée par l'émergence de nouveaux outils logiciels englobant plusieurs étapes du cycle.

Une des contributions de cette thèse est donc le développement d'une méthodologie pour la création et le dimensionnement d'une application pick & place en utilisant un seul outil logiciel (du développement d'une cellule robotique, en passant par le développement

et les tests d'algorithmes pour finir avec le passage de la simulation à l'expérimentation). Pour réaliser cela, $SOSi^{TM}$, un outil logiciel de simulation interne à la société française Solystic, a été utilisé. L'utilisation de $SOSi^{TM}$ a été détaillée en présentant les différentes étapes du cycle de dimensionnement de la Figure 3.1 à travers son utilisation dans la section 3.4. Plus précisément, la mise en place de l'architecture de l'application pick & place a été montrée ainsi que le développement du comportement des robots et de leur environnement.

À ce niveau, les deux premières étapes du cycle de développement ont été réalisées (étapes *Développement Robot et Environnement* et *Tests comportementaux*). Les deux étapes suivantes se focalisent sur le développement et le test des algorithmes permettant d'établir les lois de commande du système pick & place. En préambule du chapitre consacré aux algorithmes, il a été présenté une autre contribution de cette thèse : les blocs fonctionnels contenant les futurs algorithmes développés. Comme pour la littérature, ces blocs fonctionnels sont séparés en deux types : les règles de planification individuelles et les stratégies de collaboration. L'objectif des blocs de règles de planification individuelles est de fournir la position ou l'identifiant du produit ou de la place à atteindre par le robot associé, tandis que l'objectif du bloc de stratégies de collaboration des robots est de gérer les algorithmes de prise et de dépose et le partage de la charge de travail. Les caractéristiques de ces blocs fonctionnels leur permettent de contenir tous les algorithmes nécessaires permettant de les tester et de les comparer en ligne, en changeant simplement l'identifiant de l'algorithme souhaité, sans avoir à compiler de nouveau le programme dans le contrôleur. Cela octroie un gain de temps non négligeable lorsque des tests sur différentes configurations d'applications doivent être effectués.

Une troisième contribution de cette thèse est le développement de différents algorithmes (étape *Développement Algorithmes* du cycle de développement). Certains algorithmes, dont les règles de planifications individuelles, sont inspirés de travaux réalisés dans la littérature. Les stratégies de collaboration ont été pensées pour s'approcher du seuil de performance d'une application pick & place avec convoyeurs parallèles. Leur principe est d'assigner les produits aux robots avant qu'ils n'arrivent dans la zone de travail du premier robot. Les stratégies ont été développées selon les caractéristiques des produits utilisés dans l'application.

Après le développement des algorithmes, leurs tests en simulation ont pu être effectués (étape *Tests Algorithmes* du cycle). Comme pour les blocs fonctionnels, les tests ont été séparés en deux parties : l'influence des règles de planification individuelles et l'influence des stratégies de collaboration. La première partie a permis de souligner l'utilisation de la règle *FIFO*. En effet, les robots devant finir de remplir les boîtes doivent utiliser de préférence la règle *FIFO* pour la dépose et les robots devant finir de prendre les produits la règle *FIFO* pour la prise afin de ne pas perdre de produits ou de ne pas avoir de boîtes non remplies. La deuxième partie, quant à elle, a été divisée suivant différentes configurations d'applications exploitées en industrie. Cependant, nous avons pu mettre en avant plusieurs points importants :

— Lorsque le système est surdimensionné, il est préférable d'utiliser une stratégie de collaboration pour équilibrer la charge de travail entre les robots, afin d'éviter l'usure prématurée d'un robot.
— Pour un système utilisant une simple prise, il n'est pas nécessaire d'utiliser une

stratégie de collaboration si le système est dimensionné au plus juste.
— Pour ce type de dimensionnement, si l'application utilise une prise multiple, la stratégie *IntToExt Cyclic* avec la règle *SPT* permet de prendre 1% de produits de plus qu'une application n'utilisant aucune stratégie.
— Si l'orientation des produits doit être prise en compte, la stratégie *IntToExt Cyclic Rot* permet de prendre 1.6% de produits de plus qu'une application n'utilisant aucune stratégie.
— Un système avec des convoyeurs en contre-courant possède un débit maximum légèrement inférieur à un système avec des convoyeurs en co-courant mais permet de perdre moins de produits en cas de perturbations (sur-cadence, sous-cadence, etc.).
— Il est préférable d'utiliser un système avec des convoyeurs en contre-courant si le poids ou le type des produits doit être pris en compte.

Ces résultats montrent un des intérêts de l'utilisation d'un tel outil de simulation par des industriels comme Schneider Electric. Il permet d'aider au dimensionnement ainsi qu'au choix de la stratégie de commande suivant un cahier des charges donné en vue du développement d'une nouvelle application industrielle. En effet, nous pouvons noter que la simulation a permis non seulement de tester la fonctionnalité des algorithmes et ainsi avoir une comparaison entre eux, mais cela a donné un premier retour par rapport aux performances de ceux-ci.

L'étude en *Simulation* étant achevée, le chapitre 6 a présenté l'aboutissement du cycle de développement. En effet, une étude comparative entre la simulation et l'expérimentation a été effectuée. Afin de valider la méthodologie et l'outil développés durant ces travaux de thèse, tout le cycle de développement a été suivi pour deux démonstrateurs. L'étude en *Expérimentation* a donc été abordée avec la présentation des deux démonstrateurs utilisés ainsi que le passage entre la simulation et l'expérimentation en expliquant le parallélisme entre les deux architectures.

Deux campagnes d'essais expérimentaux ont été effectuées à l'aide des démonstrateurs. Un premier démonstrateur possédant un seul robot a permis de vérifier le bon fonctionnement de l'architecture du contrôleur, de vérifier la similitude entre la simulation et l'expérimentation et de comparer les règles de planification individuelles. Ce parallèle est un avantage de la méthodologie développée au travers de l'outil logiciel $SOSi^{TM}$. Hormis la différence syntaxique entre les deux outils logiciels ($SOSi^{TM}$ et *SoMachine Motion*), il permet de copier un algorithme depuis un outil puis de le coller dans l'autre.

Le deuxième démonstrateur possédant trois robots a permis de vérifier le bon fonctionnement et les performances des stratégies de collaboration avec l'objectif de montrer la cohérence entre la simulation et l'expérimentation. Cette deuxième partie a aussi démontré l'intérêt de l'utilisation sur site de cette méthodologie et de cet outil logiciel. Il a permis de préparer en amont les différents tests et le parallélisme de l'architecture a permis le fonctionnement du démonstrateur avec les blocs fonctionnels développés dès le premier jour des essais.

Nous pouvons conclure que l'utilisation de la méthodologie et de l'outil logiciel développés au cours de cette thèse a plusieurs avantages. Travailler sur une application en simulation à un coup plus réduit que de travailler sur une application réelle. Des applications pick & place peuvent être testées, corrigées et améliorées avant la mise en service

sans contraintes mécaniques et de sécurité. Cela permet de gagner du temps car la mise en service d'une machine peut se réaliser plus rapidement (les tests ont été effectués antérieurement et les architectures de simulation et du contrôleur sont identiques). De plus, certains problèmes pouvant être rencontrés durant la mise en service peuvent être évités. Finalement, il peut y avoir un intérêt stratégique, plusieurs solutions peuvent être montrées aux clients avant une prise de décisions et d'achats. Cet outil permet d'une certaine manière d'accompagner le client dans la définition et la mise en place d'une solution.

Perspectives

Les voies d'améliorations de la méthodologie et de l'outil logiciel sont nombreuses. En effet chaque étape de la méthodologie peut être un axe de recherche. Tout d'abord, au niveau de l'étape *Développement Robot et Environnement*, une amélioration du visuel des composants peut être faite afin de se rapprocher au plus près du réel. En effet, actuellement, l'application a été seulement dessinée sous *Autodesk 3ds Max*. Cependant, il y a la possibilité d'importer des fichiers CAO des robots. De plus l'ergonomie de l'outil logiciel peut être améliorée pour faciliter le développement d'une application.

Ensuite, il y a une amélioration perpétuelle au niveau de l'étape *Développement Algorithmes*. En effet, celle-ci consiste à rechercher des algorithmes afin de pouvoir optimiser les performances des applications pick & place multi-robots.

Il y a une voie de recherche pour faciliter le passage entre *Simulation* et *Expérimentation*. Effectivement, le parallélisme entre les architectures de simulation et du contrôleur peut être amélioré en les rendant davantage similaire. Une simplification au maximum des programmes simulation et en expérimentation peut être envisagée afin d'améliorer les temps de calcul mais aussi de pouvoir développer un outil de traduction pour réaliser rapidement ce passage.

Finalement, une amélioration des étapes *Validation Algorithmes* et *Tests Performances* est à réaliser en passant par une simplification des blocs fonctionnels, dans leur utilisation et dans leur fonctionnement. On peut envisager par exemple une intelligence des blocs fonctionnels qui choisit automatiquement les algorithmes suivant les caractéristiques du système et des évènements externes (sur-cadence etc.), ces algorithmes étant au préalable sélectionnés à l'aide de tests en simulation et en expérimentation.

En résumé, tous ces axes de recherches ont pour objectif de simplifier le développement d'une application pick & place afin de le rendre plus rapide et plus efficace.

Un dernier axe de recherche peut être réalisé sur la définition du cycle de développement. Actuellement, il est effectué dans un seul sens sans retour possible. Un début de changement a été initié dans la section 6.3.1 où des données du démonstrateur ont été réinjectées dans la simulation. C'est dans cette optique que le cycle actuel peut être transformé en un cycle de développement itératif reprenant les mêmes étapes, voir Figure 7.1.

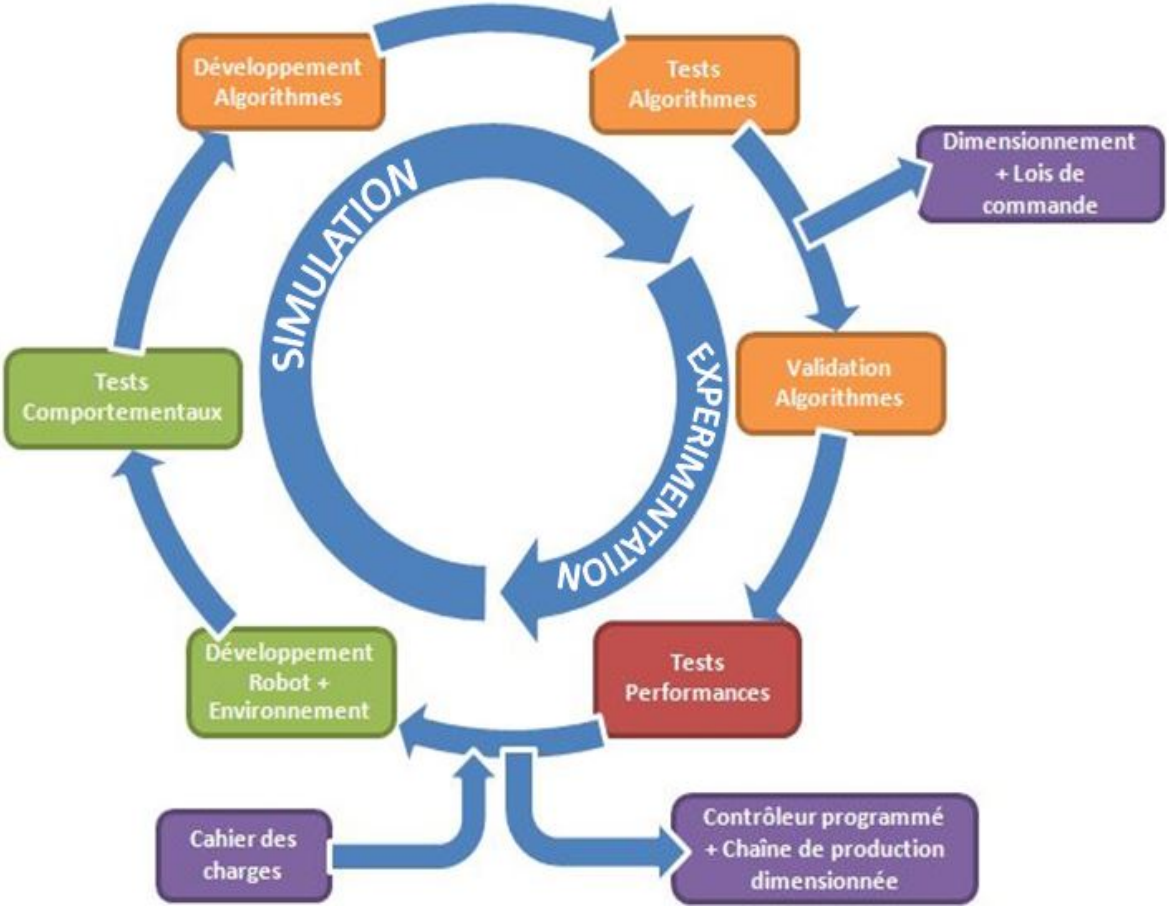

FIGURE 7.1 – Cycle de développement itératif d'une application pick & place.

Glossaire

ACO : Ant Colony Optimization. Optimisation par colonie de fourmis.

AIP RAO : Atelier Inter-établissement de Productique Rhône-Alpes Ouest. Centre de ressources et de compétences.

CAD : Computer-Aided Design. CAO : Conception Assistée par Ordinateur.

CAN : Controller area network. Bus de terrain série.

CAO : Conception Assistée par Ordinateur.

CATIA : Conception Assistée Tridimensionnelle Interactive Appliquée. Logiciel de CAO.

Ethernet : Protocole de réseau local à commutation de paquets.

GRAFCET : GRAphe Fonctionnel de Commande des Etapes et Transitions. Langage graphique représentant le fonctionnement d'un automatisme

Heuristique : Méthode de calcul qui fournit une solution réalisable, pas nécessairement optimale, pour un problème d'optimisation.

Phéromone : Substances chimiques comparables aux hormones. Vient des termes pherein (transporter) et hormon (exciter).

Probléme du voyageur de commerce : Probléme d'optimisation qui, étant donnée une liste de villes, et des distances entre toutes les paires de villes, détermine un plus court chemin qui visite chaque ville une et une seule fois et qui termine dans la ville de départ.

Problème du sac à dos : Problème d'optimisation combinatoire. Il modélise une situation analogue au remplissage d'un sac à dos, ne pouvant supporter plus d'un certain poids. Les objets mis dans le sac à dos doivent maximiser la valeur totale, sans dépasser le poids maximum.

SERCOS : SErial Real-time COmmunications System. Protocole de communication utilisé notamment en robotique.

SIMAN : Processeur et langage de simulation.

TCP/IP : Ensemble des protocoles utilisés pour le transfert des données sur Internet.

VBA : Visual Basic for Applications. Langage informatique.

XML : Extensible Markup Language. Langage de balisage extensible.

Bibliographie

[Obs, 2012] (2012). Observatoire de l'emballage (2012).

[ABB, 2014] ABB (2014). Abb robotstudio picking powerpac simplifies the design and programming of complex robotic pick-and-pack systems.

[Aguero et al., 2015] Aguero, C., Koenig, N., Chen, I., Boyer, H., Peters, S., Hsu, J., Gerkey, B., Paepcke, S., Rivero, J., Manzo, J., Krotkov, E., and Pratt, G. (2015). Inside the virtual robotics challenge : Simulating real-time robotic disaster response. *Automation Science and Engineering, IEEE Transactions on*, 12(2) :494–506.

[Alaya et al., 2006] Alaya, I., Solnon, C., and Ghedira, K. (2006). Optimisation par colonies de fourmis pour le problème du sac-à-dos multidimensionnel. Technical Report RR-LIRIS-2006-005, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon.

[Bonert et al., 2010] Bonert, M., Shu, L., and Benhabib, B. (2010). Motion planning for multi-robot assembly systems. *International Journal of Computer Integrated Manufacturing*, 13(4) :301–310.

[Bosch, 2011] Bosch, R. (2011). Drive & control profile.

[Bozma and Kalalioglu, 2012] Bozma, H. I. and Kalalioglu, M. (2012). Multirobot coordination in pick-and-place tasks on a moving conveyor. *Robotics and Computer-Integrated Manufacturing*, 28 :530–538.

[Chang et al., 2012] Chang, P., Huang, W., and C.J., T. (2012). Developing a varietal ga with esma strategy for solving the pick and place problem in printed circuit board assembly line. *Journal of Intelligent Manufacturing*, 23(5) :1589–1602.

[Clavel, 1990] Clavel, R. (1990). Device for the movement and positioning of an element in space.

[Comba et al., 2013] Comba, L., Belforte, G., and Gay, P. (2013). Plant layout and pick-and-place strategies for improving performances in secondary packaging plants of food products. *Packaging Technology and Science*, 26(6) :339–354.

[Concannon et al., 2003] Concannon, K., Hunter, K., and Tremble, J. (2003). Simul8-planner simulation-based planning and scheduling. In *Proceedings of the 2003 Winter Simulation Conference, 2003.*, number 2, pages 1488 – 1493. IEEE.

[Daoud et al., 2014a] Daoud, S., Chehade, H., Yalaoui, F., and Amodeo, L. (2014a). Solving a robotic assembly line balancing problem using efficient hybrid methods. *Journal of Heuristics*, 20(3) :235–259.

[Daoud et al., 2014b] Daoud, S., Hicham, C., Farouk, Y., and Lionel, A. (2014b). Efficient metaheuristics for pick and place robotic systems optimization. *Journal of Intelligent Manufacturing*, 25 :27–41.

[DGCIS, 2009] DGCIS (2009). La robotisation des pmi françaises. *Rapport*. DGCIS : Direction Générale de la Compétitivité dans l'Industrie et les Services.

[Dorigo, 1992] Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy.

[Dorigo et al., 1991] Dorigo, M., Colorni, A., and Maniezzo, V. (1991). Distributed optimization by ant colonies. In *Proceedings of ECAL91 - European conference on Artificial Life, Paris, France*, volume 142, pages 134–142. Elsevier.

[Edan et al., 2004] Edan, Y., Berman, S., Boteach, E., and Mendelson, E. (2004). Distributed multi-robot assembly-packaging algorithms. *Intelligent Automation and Soft Computing,*, 10(4) :349–357.

[Ehrat and Renner, 2010] Ehrat, M. and Renner, J. (2010). Process and apparatus for introducing products into containers in a pickerline. Patent US 2010/0242415 A1.

[Erlandsson-Warvelin and Knobel, 2012] Erlandsson-Warvelin, M. and Knobel, H. (2012). Control method for machines, including a system, computer program, data signal and gui. Patent US 20070108109 A1.

[Fanuc, 2013] Fanuc (2013). Fanuc america demonstrates learning capabilities with m-3ia/6s robot at pack expo 2013.

[Fujimoto et al., 1995] Fujimoto, H., Tanigawa, I., Yasuda, K., and Iwahashi, K. (1995). Applications of genetic algorithm and simulation to dispatching rule-based fms scheduling. In *Proceedings 1995 IEEE International Conference on Robotics and Automation*, volume 1, pages 190–195. IEEE.

[González-Palacios, 2012] González-Palacios, M. A. (2012). Advanced engineering platform for industrial development. *Journal of Applied Research and Technology*, 10(3) :309–326.

[Harmatia and Skrzypczykb, 2009] Harmatia, I. and Skrzypczykb, K. (2009). Robot team coordination for target tracking using fuzzy logic controller in game theoretic framework. *Robotics and Autonomous Systems*, 57 :75–86.

[Herzog, 2003] Herzog, F. (2003). Method and apparatus for filling containers with piece goods. Patent US 2003/0037515 A1.

[Hindle and Duffin, 2006] Hindle, K. and Duffin, M. (2006). Simul8-planner for composites manufacturing. In *Proceedings of the Winter Simulation Conference, 2006. WSC 06.*, pages 1779 – 1784. IEEE.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. MIT Press.

[Huang et al., 2012] Huang, Y., Chiba, R., Arai, T., Ueyama, T., and Ota, J. (2012). Part dispatching rule-based multi-robot coordination in pick-and-place task. In *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1887 – 1892. IEEE.

[Huang et al., 2013] Huang, Y., Chiba, R., Arai, T., Ueyama, T., and Ota, J. (2013). Integrated design of multi-robot system for pick-and-place tasks. In *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 970 – 975. IEEE.

[Humbert et al., 2015] Humbert, G., Pham, M., Brun, X., Guillemot, M., and Noterman, D. (2015). Comparative analysis of pick & place strategies for a multi-robot application. In *20th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) 2015*, Luxembourg. IEEE.

[Humbert et al., 2016a] Humbert, G., Pham, M., Brun, X., Guillemot, M., and Noterman, D. (2016a). Development of a methodology for performance analysis and synthesis of control strategies of multi-robot pick & place applications. In *International Joint Conference on Mechanics, Design Engineering & Advanced Manufacturing (JCM 2016)*, Catania, Italy. IEEE.

[Humbert et al., 2016b] Humbert, G., Pham, M., Brun, X., Guillemot, M., and Noterman, D. (2016b). Development of a methodology to improve the performance of multi-robot pick & place applications : from simulation to experimentation. In *2016 IEEE International Conference on Industrial Technology (ICIT2016)*, Taipei, Taiwan. IEEE.

[Humbert et al., 2016c] Humbert, G., Pham, M., Brun, X., Guillemot, M., and Noterman, D. (2016c). Développement d'une méthodologie pour l'amélioration des performances d'applications multi-robots pick & place : de la simulation à l'expérimentation. *Journal Européen des Systèmes Automatisés*.

[Huppi et al., 2003] Huppi, E., Dubendorfer, P., Kirgis, F., Sidler, M., Van Kogelenberg, J., and Schule, S. (2003). Method and apparatus for puting piece goods into containers. Patent US 2003/0182898 A1.

[Izumi et al., 2013] Izumi, T., Koyanagi, K., Matsukuma, K., and Hashiguchi, Y. (2013). Robot system. Patent US 8606400 B2.

[Johari et al., 2007] Johari, N., Haron, H., and Jaya, A. (2007). Robotic modeling and simulation of palletizer robot using workspace5. In *4th International Conference on Computer Graphics, Imaging and Visualization (CGIV 2007), August 14-16, 2007, Bangkok, Thailand*, pages 217 – 222. IEEE.

[Johnson, 1956] Johnson, R. V. (1956). A computing procedure for a line balancing problem. *Management Science*, 2(3) :261–271.

[Johnson, 1988] Johnson, R. V. (1988). Optimally balancing large assembly lines with "fable". *Management Science*, 34(2) :240–253.

[Keba, 2014] Keba (2014). Real world simulation package.

[Kennedy and Eberhart, 1995] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 4, pages 1942 – 1948. IEEE.

[Koenig and Howard, 2004] Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004 (IROS 2004).*, volume 3, pages 2149–2154, Sendai, Japan. IEEE.

[Makino, 1982] Makino, H. (1982). Assembly robot. Patent US4341502.

[Mattone et al., 1998] Mattone, R., Adduci, L., and Wolf, A. (1998). Online scheduling algorithms for improving performance of pick-and-place operations on a moving conveyor belt. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA-98, Leuven, Belgium, May 16-20, 1998*, volume 3, pages 2099 – 2105 vol.3. IEEE.

[Meiyun and Heguo, 2011] Meiyun, Z. and Heguo, X. (2011). Research on the production system of an enterprise simulation base on witness. In *International Conference on E -Business and E -Government (ICEE), 2011*, pages 1 – 4. IEEE.

[Mendelson et al., 2002] Mendelson, E., Nayer, O., Berman, S., and Edan, Y. (2002). Behavior-based control of multi-robot assembly palletizing systems. In *Proceedings of the 5th Biannual World Automation Congress, 2002*, volume 14, pages 1–6. IEEE.

[Mirzapourrezaei et al., 2011] Mirzapourrezaei, S., Lalmazloumian, M., Dargi, A., and Wong, K. Y. (2011). Simulation of a manufacturing assembly line based on witness. In *Third International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN), 2011*, pages 132 – 137.

[Najera and Brizuela, 2005] Najera, A. and Brizuela, C. (2005). Pcb assembly : an efficient genetic algorithm for slot assignment and component pick and place sequence problems. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC2005, 2-4 September 2005, Edinburgh, UK*, volume 2, pages 1485–1492. IEEE.

[Nikakhtar et al., 2011] Nikakhtar, A., Wong, K. Y., Zarei, M., and Memari, A. (2011). Comparison of two simulation software for modeling a construction process. In *Third International Conference on Computational Intelligence, Modelling and Simulation (CIMSiM), 2011*, pages 200 – 205. IEEE.

[OSEO, 2011] OSEO (2011). L'innovation dans les entreprises en 2010. *Synthèse sectorielle Emballage-Conditionnement.*

[Pegden, 2007] Pegden, C. (2007). Simio : A new simulation system based on intelligent objects. In *Simulation Conference, 2007 Winter*, pages 2293 – 2300. IEEE.

[Pegden and Sturrock, 2009] Pegden, C. and Sturrock, D. (2009). Introduction to simio. In *Simulation Conference (WSC), Proceedings of the 2009 Winter*, pages 314 – 321. IEEE.

[Peng and Zeng, 2013] Peng, G. and Zeng, K. (2013). An ad-hoc method with genetic algorithm for printed circuit board assembly optimization on the sequential pick-and-place machine. In *International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT 2013, Taipei, Taiwan, December 16-18, 2013*, pages 128–133. IEEE Computer Society Washington, DC, USA ©2013 ISBN : 978-1-4799-2419-6 doi>10.1109/PDCAT.2013.27.

[Rubinovitz et al., 1998] Rubinovitz, J., Bukchin, J., and Lenz, E. (1998). Ralb - a heuristic algorithm for design and balancing of robotic assembly lines. *CIRP Annals - Manufacturing Technology*, 42(1) :497–500.

[Rutschmann, 2010] Rutschmann, H. (2010). Picking line and method for inserting products into packaging container. Patent US 2010/0223883 A1.

[Sahin, 2005] Sahin, H. (2005). *Design of a secondary packaging robotic system.* PhD thesis, Middle est technical university.

[Sam et al., 2012] Sam, R., Arrifin, K., and Buniyamin, N. (2012). Simulation of pick and place robotics system using solidworks softmotion. In *International Conference on System Engineering and Technology (ICSET), 2012*, pages 1 – 6. IEEE.

[Schubert, 2000] Schubert, R. (2000). Process and apparatus for introducing products into containers. Patent US 6122895 A.

[Staubli, 2013] Staubli (2013). Robot linemanager system.

[Storn and Price, 1997] Storn, R. and Price, K. (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4) :341–359.

[Stumm et al., 2014] Stumm, S., Pintar, D., and Kuhlenkoetter, B. (2014). A novel concept for realistic simulation of industrial pick and place applications. In *Proceedings of ISR/Robotik 2014 ; 41st International Symposium on Robotics ;*, pages 1–7. VDE.

[Sturrock and Pegden, 2011] Sturrock, D. and Pegden, C. (2011). Recent innovations in simio. In *Proceedings of the 2011 Winter Simulation Conference (WSC)*, pages 52 – 62. IEEE.

[von Neumann, 1928] von Neumann, J. (1928). Zur theorie der gesellschaftspiele. *Mathematische Annalen*, 100 :295–320.

[von Neumann and Morgenstern, 1944] von Neumann, J. and Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press.

[Wappling and Murphy, 2012] Wappling, D. and Murphy, S. (2012). Pick and place. Patent US 20120165972 A1.

[Zadeh, 1964] Zadeh, L. (1964). Fuzzy sets. *Information and Control*, 8(3) :338–353.

[Zhu and Chen, 2009] Zhu, G.-Y. and Chen, Z.-J. (2009). A differential evolution optimization approach to solve the pick-and-placing problem. In *Fifth International Conference on Natural Computation, 2009. ICNC '09.*, volume 4, pages 66–70. IEEE.

Annexe A

Exemples de programme

Dans cette annexe nous allons détailler le programme de deux algorithmes : la règle individuelle *FIFO* et une stratégie de collaboration *IntToExt*. Ceci n'est qu'un extrait de 150 lignes de code parmi plus de 4000 lignes programmées.

```
Void soPickerAlgorithm::FIFOProduct(soPickerConveyor* Conveyor)
{
 static bool xFind = false;
 switch(m_iStateProduct)
 {
 case 0: // Initialization
  xFind          = false;
  m_fFIFOdist    = m_BaseCenter.z - fPickRadiusMax;
  m_iStateProduct = 10;
  break;

 case 10:
  for (int iProd = 0; iProd < Conveyor->GetObjectVec().GetCount(); iProd++)
  {// Product Searching
   D3DXMATRIX  Mat      = Conveyor->GetObjectVec().Get(iProd)->GetMatrix();
   D3DXVECTOR3 Position = D3DXVECTOR3(Mat._41,Mat._42,Mat._43);
   if(Position.z >= m_BaseCenter.z + m_fPickRadiusMin
      && Position.z < m_BaseCenter.z + m_fPickRadiusMax)
    xFind=true; // Object is in the robot working area

   if(xFind)
   {
    xFind=false;
    CEmulatorCloneTransform* pECT=Conveyor->GetObjectVec().Get(iProd);
    // Use of the object class to use the object function
    soPickerObject* pBOC=dynamic_cast<soPickerObject*>(pECT);
    if(pBOC->GetRobotID() <= m_iAlgorithmID)
    {// Robot picks its object or the object of the robot upstream
     if(Position.z > m_fFIFODist)
     {// The oldest product is the most downstream product
      m_fFIFODist      = Position.z;
      m_iProductID     = iProd;      // Product ID
      m_iStateProduct  = 20;
     }
    }
   }
  }
  break;

 case 20:
  m_xFound        = true; // Product Found
  m_iStateProduct = 0;
  break;
 }
}
```

FIGURE A.1 – Code de la règle de planification individuelle *FIFO* dans le langage C++.

```
CASE THIS^.diStateProduct OF
0 :      (* Initialization *)
 THIS^.xFound            := FALSE;
 THIS^.lrFIFODist        := THIS^.i_stRobotPosition.X - THIS^.i_stWorkingArea.stMaxReachPickArea.X;
 THIS^.diStateProduct    := 10;

10 : (* Products Searching *)
 FOR udiProd := 1 TO THIS^.i_udiMaxNbOfProducts DO
  stPosition := THIS^.i_astProductPosition[udiProd];

 (* Product not picked is in the robot working area *)
 IF NOT THIS^.i_stProductParameters[udiProd].xPicked THEN
  IF stPosition.X >= THIS^.i_stRobotPosition.X + THIS^.i_stWorkingArea.stMinReachPickArea.X
     AND stPosition.X < THIS^.i_stRobotPosition.X + THIS^.i_stWorkingArea.stMaxReachPickArea.X
     THEN
   THIS^.xFound := TRUE;
  END_IF
 END_IF

 IF THIS^.xFound THEN
  THIS^.xFound := FALSE;
  (* Robot picks its products or the products of the robot upstream *)
  IF THIS^.i_stProductParameters[udiValue].udiID <= THIS^.i_udiRobotID THEN
   IF stPosition.X > THIS^.lrFIFODist  THEN       (* The oldest product is the most downstream product *)
    THIS^.lrFIFODist       := stPosition.X;
    THIS^.q_diProductID := UDINT_TO_DINT(udiProd);
    THIS^.diStateProduct := 20;
   END_IF
  END_IF
 END_IF
 END_FOR

20 :
 THIS^.iq_stPickPlaceState.xFound        := TRUE;
 THIS^.diStateProduct                    := 0;
END_CASE
```

FIGURE A.2 – Code de la règle de planification individuelle *FIFO* dans le langage SoMachine Motion.

```
void soSupervisorAlgorithm::AssignProductsDownToUp(bool xSPT)
{
 asVector<int>    viProducts        = m_viMultiPicking;
 asVector<int>    viProductsCumul   = m_viMultiPickingCumul;
 static int       iProductID        = 0;

 for(int p = 0; p < m_pPickingConveyor->GetObjectVec().GetCount() ; p++)
 {// Product searching
  CEmulatorCloneTransform* pECT = m_pPickingConveyor->GetObjectVec().Get(p);
  // Use of the object class to use the object function
  soPickerObject* pBOC  = dynamic_cast<soPickerObject*>(pECT);
  D3DXMATRIX  Mat       = pECT->GetMatrix();
  D3DXVECTOR3 Position  = D3DXVECTOR3(Mat._41,Mat._42,Mat._43);
  float fConvBegin = m_pPickConv->GetCenter().z - m_pPickConv->GetBoxMax().z

  if(Position.z >= fConvBegin + m_fSensor*5 &&
     Position.z <= fConvBegin + m_fSensor*5 + 50)
  {// Product passes the virtual sensor
   if(pBOC->GetRobotID() == 0)
   {// Product is not assigned
    for(int i = 1; i <= m_iRobotNumber; i++)
    {// To each robot
     int k = i;
     for(int z = 1; z <= viProducts.Get(k); z++)
     {// Number of products to assign
      if(xSPT && z == 1 || !xSPT)
      {// First product or no use of the SPT rule
       for(int q = p; q < p + viProductsCumul.Get(m_iRobotNumber); q++)
       {// Oldest product searching
        CEmulatorCloneTransform* pECT = m_pPickConv->GetObjectVec().Get(q);
        // Use of the object class to use the object function
        soPickerObject* pBOC = dynamic_cast<soPickerObject*>(pECT);
        if(pBOC->GetRobotID() == 0)
        {// Product is not yet assigned
         D3DXMATRIX Mat = pECT->GetMatrix();
         Pos            = D3DXVECTOR3(Mat._41,Mat._42,Mat._43);
         iProductID     = q;        // Product ID
         m_PosEnd       = Pos;      // Product position
         break;
        }
       }
      }
```

```
      else if(xSPT && z > 1)
      {//Use of the SPT rule
       m_fNearDist = 10000;
       for(int q = p; q < p + viProductsCumul.Get(m_iRobotNumber) ; q++)
       {// Nearest product searching
        CEmulatorCloneTransform* pECT = m_pPickConv->GetObjectVec().Get(q);
        // Use of the object class to use the object function
        soPickerObject* pBOC = dynamic_cast<soPickerObject*>(pECT);
        if(pBOC->GetRobotID() == 0)
        {// Product is not yet assigned
         Mat                        = pECT->GetMatrix();
         D3DXVECTOR3 Pos            = D3DXVECTOR3(Mat._41,Mat._42,Mat._43);
         D3DXVECTOR3 OffsetVector   = Pos - m_PosInit;
         if(D3DXVec3Length(&OffsetVector) < m_fNearDist)
         {
          m_fNearDist = D3DXVec3Length(&OffsetVector);
          iProductID    = q;        // Product ID
          m_PosEnd      = Pos;      // Product position
         }
        }
       }
      }
      m_PosInit = m_PosEnd;
      CEmulatorCloneTransform* pECT = m_pPickConv-
>GetObjectVec().Get(iProductID);
      // Use of the object class to use the object function
      soPickerObject* pBOC = dynamic_cast<soPickerObject*>(pECT);
      pBOC->SetRobotID(k);                              // The object is assign
to the robot k
      pECT->SetColor(m_vColor.Get(k));          // The object takes the k
color
     }
    }
   }
  }
 }
}
```

FIGURE A.3 – Code de la stratégies de collaboration *DownToUp* avec possibilité d'utilisation de la règle *SPT* dans le langage C++.

Annexe B

Études comparatives complémentaires

Impact de la marge de sécurité

Dans cette partie, nous allons reprendre des tests initiés dans la section 5.4. L'objectif de cette étude complémentaire est de montrer les performances d'une application pour différentes stratégies de collaboration pour différentes marges de sécurité. Durant ces nouveaux tests, le cahier des charges est le même que celui utilisé durant les simulations initiales, cependant, seuls le nombre de produits perdus ainsi que la charge de travail de chaque robot sont reportés. Les tests approfondis sont les suivants :

— Flux de produits simples positionnés aléatoirement avec convoyeurs en co-courant et prise simple.
— Flux de produits simples positionnés aléatoirement avec convoyeurs en contre-courant et prise simple.

Convoyeurs en co-courant

La Figure B.1 montre le pourcentage de produits perdus en fonction de la marge de sécurité suivant différentes stratégies de collaboration pour un système ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple. Tout d'abord nous pouvons noter que pour une marge de sécurité entre 2% et 7.5% il n'y a aucun produit perdu pour les différentes stratégies de collaboration. De plus, pour une marge supérieure à 7.5% (correspondant à un système fortement surdimensionné) le nombre de produits perdus augmente de plus en plus en absence de stratégie. Tandis qu'en présence d'une stratégie, le nombre de produits perdus est nul. Pour les meilleures stratégies (*Nothing Linear* et *IntToExt Cyclic*), la marge de sécurité doit être inférieure à 1% pour avoir une perte de produits. Ceci est dû au temps de calcul nécessaire (environ 1 ms, voir section 5.4).

Les graphiques de la Figure B.2 montrent la charge de travail des robots en fonction de la marge de sécurité suivant différentes stratégies de collaboration pour un système ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple. Les graphiques sont regroupés suivant l'affichage Stratégie de prise / Stratégie de dépose. Tout d'abord, nous remarquons que lorsqu'il n'y a aucune stratégie de collaboration (Figure B.2.a), le déséquilibre de la charge de travail des robots dans le système augmente en fonction de la marge de sécurité. L'ajout d'une stratégie de

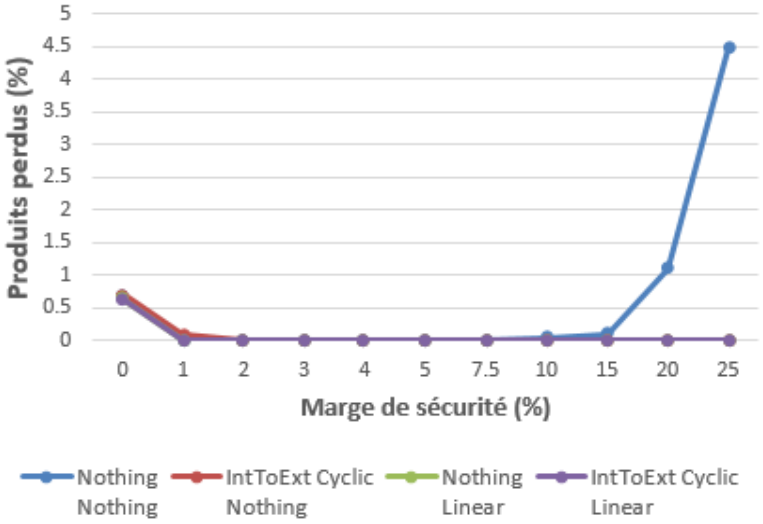

FIGURE B.1 – Pourcentage de produits perdus en fonction de la marge de sécurité suivant différentes stratégies de collaboration pour un système ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple.

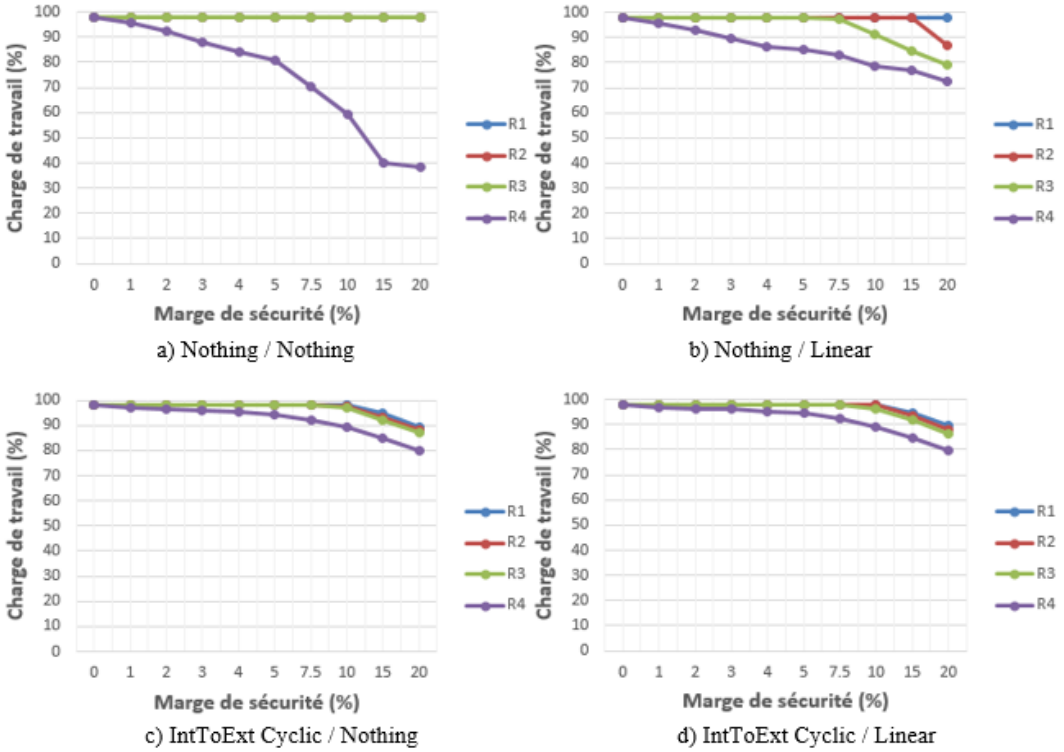

FIGURE B.2 – Charge de travail des robots en fonction de la marge de sécurité suivant différentes stratégies de collaboration pour un système ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple.

dépose (Figure B.2.b) permet de réduire ce déséquilibre. Cependant c'est l'utilisation d'une stratégie de prise qui permet d'équilibrer au mieux la charge travail entre les robots.

Convoyeurs en contre-courant

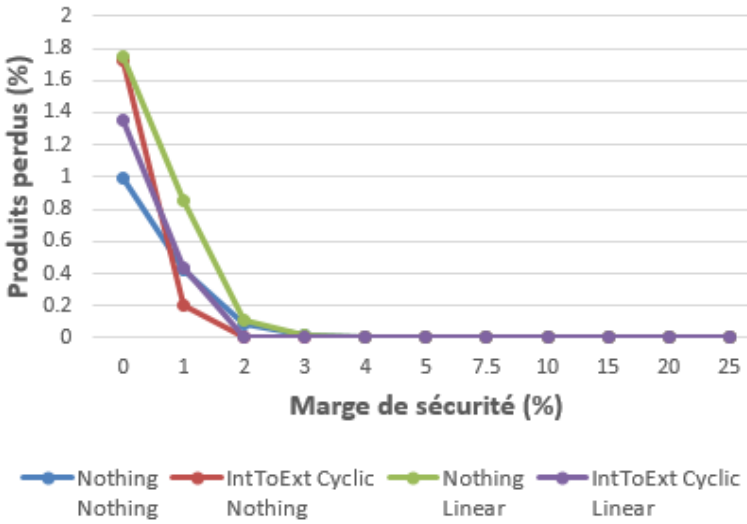

FIGURE B.3 – Pourcentage de produits perdus en fonction de la marge de sécurité suivant différentes stratégies de collaboration pour un système ayant des convoyeurs en contre-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple.

La Figure B.3 montre le pourcentage de produits perdus en fonction de la marge de sécurité suivant différentes stratégies de collaboration pour un système ayant des convoyeurs en contre-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple. Tout d'abord nous pouvons noter que pour une marge de sécurité supérieure à 3% il n'y a aucun produit perdu pour les différentes stratégies de collaboration. Pour une marge de sécurité inférieure à 3%, la pire stratégie est *Nothing Linear*. L'absence de toutes stratégies permet de limiter le nombre de produits perdus lorsque la marge de sécurité est nulle.

Les graphiques de la Figure B.4 montrent la charge de travail des robots en fonction de la marge de sécurité suivant différentes stratégies de collaboration pour un système ayant des convoyeurs en contre-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple. Les graphiques sont regroupés suivant l'affichage Stratégie de prise / Stratégie de dépose. Tout d'abord, nous pouvons noter que les résultats obtenus suivent les mêmes tendances que pour un système avec des convoyeurs en co-courant. Lorsqu'il n'y a aucune stratégie de collaboration, le déséquilibre de la charge de travail des robots dans le système augmente en fonction de la marge de sécurité. L'ajout d'une stratégie de dépose permet de réduire légèrement ce déséquilibre. C'est l'utilisation d'une stratégie de prise qui permet d'équilibrer au mieux la charge travail entre les robots. Cependant, nous remarquons que la charge de travail du premier robot est légèrement inférieure à celle des autres robots.

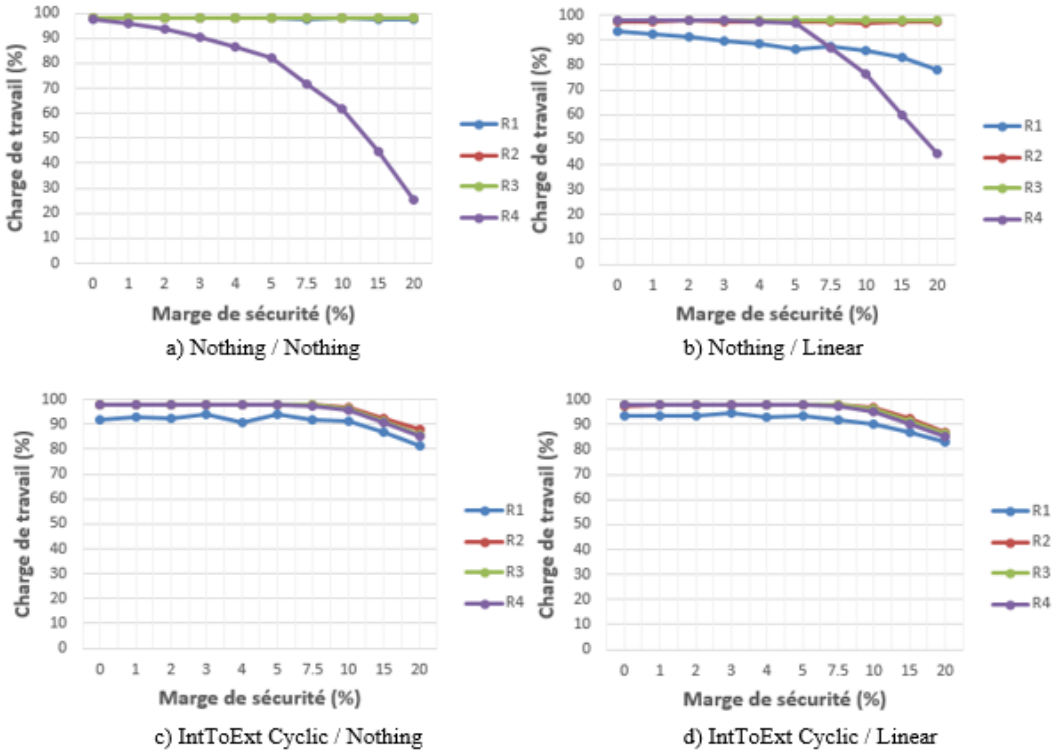

FIGURE B.4 – Charge de travail des robots en fonction de la marge de sécurité suivant différentes stratégies de collaboration pour un système ayant des convoyeurs en contre-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple.

Conclusion sur l'étude de la marge de sécurité

Durant cette étude, nous avons retrouvé les mêmes résultats que durant la section 5.4 :

— Sans stratégie de collaboration la charge de travail entre les robots est déséquilibrée.
— Pour un système avec convoyeurs en co-courant, lorsque le système est surdimensionné, la présence d'une stratégie permet de ne pas perdre de produits.
— Pour un système avec convoyeurs en contre-courant, lorsque le système est surdimensionné, la charge de travail entre les robots est toujours équilibrée.
— Un système avec convoyeur en co-courant permet d'avoir une marge de sécurité plus petite qu'un système avec convoyeurs en contre-courant.

De plus cette étude a permis de montrer la transition entre un système dimensionné au plus juste et un système fortement surdimensionné.

Vitesse des convoyeurs

Durant cette partie, l'étude se focalise sur l'impact que peut avoir la vitesse des convoyeurs sur les performances d'une application pick & place. Le nombre de produits perdus et le nombre de boîtes non remplies sont analysés pour différentes stratégies de collaboration suivant différentes vitesses d'entrée puis différentes vitesses de sortie. Outre le cahier des charges initial de la section 5.2, de nouvelles données sont ajoutées :

— Le système est dimensionné au plus juste.
— Il n'y a pas d'utilisation de la stratègie *PlacingPriority*.
— Le temps de simulation est de 20 min pour chaque configuration.

Vitesse convoyeur d'entrée

Durant cette simulation, la vitesse du convoyeur d'entrée varie entre 0.1 et 0.8 m/s mais en gardant le même débit de produits (c'est la densité de produits qui varie). Selon le cahier des charges initial, l'espacement entre le centre des boîtes est fixé arbitrairement à 0.15 m ce qui correspond à une vitesse de sortie de 0.17 m/s.

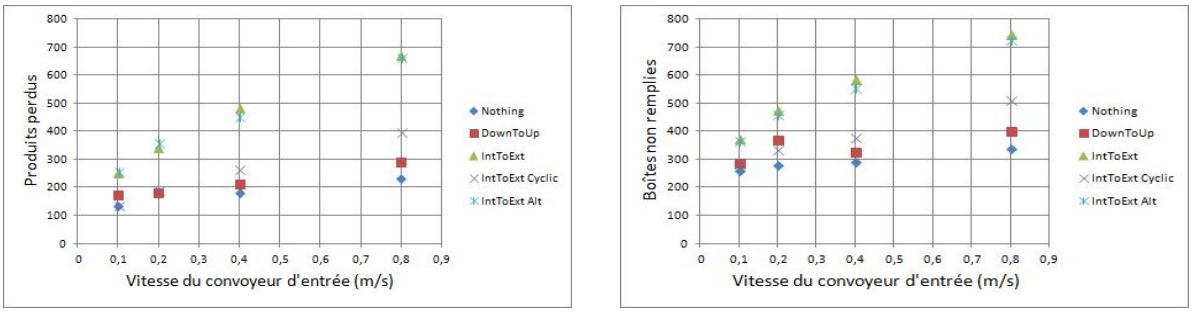

FIGURE B.5 – Produits perdus et boîtes non remplies en fonction de la vitesse du convoyeur d'entrée suivant différentes stratégies de collaboration pour un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple.

Les graphiques de la Figure B.5 montrent l'impact de la vitesse du convoyeur d'entrée sur le nombre de produits perdus et de boîtes non remplies suivant différentes stratégies de collaboration pour un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple. Il est à noter que plus la vitesse du convoyeur d'entrée est élevée plus les produits et les boîtes sont perdus. Cela se produit pour toutes les stratégies. Nous pouvons noter que les stratégies *IntToExt* et *IntToExt Alt* obtiennent les pires résultats comme vu précédemment. Les stratégies limitant le plus l'augmentation de la perte sont *Nothing* et *DownToUp*

Vitesse convoyeur de sortie

Durant cette simulation, la vitesse du convoyeur de sortie varie entre 0.1169 m/s (qui correspond à un espacement entre les boîtes de 0m) et 0.4 m/s (espacement de 0.1566m) mais en gardant le même débit de boîtes (c'est l'espacement entre les boîtes qui varie). Selon le cahier des charges, la vitesse du convoyeur d'entrée est fixée arbitrairement à 0.2 m/s.

Les graphiques de la Figure B.6 montrent l'impact de la vitesse du convoyeur de sortie sur le nombre de produits perdus et de boîtes non remplies selon différentes stratégies de collaboration pour un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple. Il est à noter que les meilleures stratégies perdent moins de produits et de boîtes avec

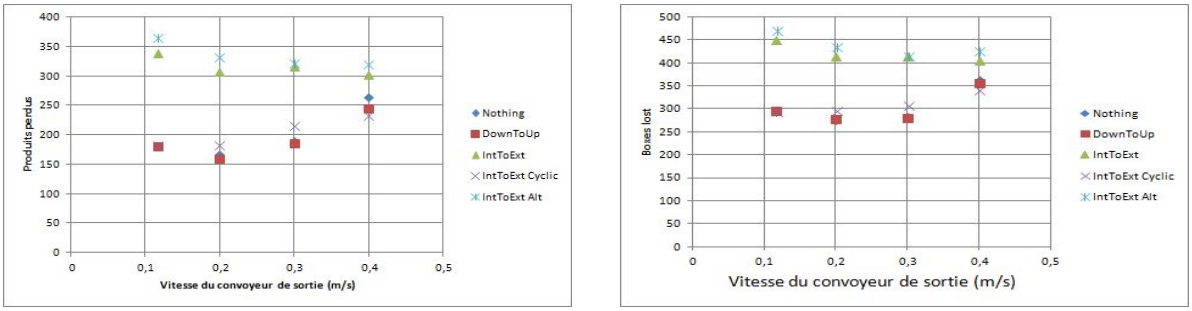

FIGURE B.6 – Produits perdus et boîtes non remplies en fonction de la vitesse du convoyeur de sortie suivant différentes stratégies de collaboration pour un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple.

une vitesse égale à 0,2 m/s. Les stratégies *IntToExt* et *IntToExt Alt* perdent moins de produits et de boîtes lorsque la vitesse augmente, mais reste plus élevées que les autres stratégies. Nous remarquons que la stratégie *IntToExt Cyclic* a une augmentation plus faible que les stratégies *Nothing* et *DownToUp*.

Afin de minimiser le nombre de produits perdus et de boîtes non remplies, la vitesse du convoyeur d'entrée doit être réduite à son minimum. Pour ce système, la vitesse du convoyeur de sortie doit être proche de 0,2 m/s (espace égal à 0,14 m).

Panne d'un robot

Dans cette partie, nous allons étudier l'impact que peut avoir la panne d'un robot ainsi que son emplacement dans le système multi-robots. Outre le cahier des charges initial de la section 5.2, de nouvelles données sont ajoutées :

- Seuls les trois premiers robots fonctionnement.
- Lors de la panne d'un robot le dernier robot prend le relais du robot défaillant.
- Le système est dimensionné au plus juste.
- Il n'y a qu'une panne par test.
- Le temps de simulation est de 10 min pour chaque configuration.
- La panne arrive à 5 min de simulation.
- Aucune stratégie de prise dépose n'est utilisée.

Le nombre de robots ayant changé, la cadence totale du système est aussi modifiée. Un nouveau dimensionnement doit être effectué. Pour un système dimensionné au plus juste, selon la méthode décrite dans la section 5.1, un débit de produits est choisi $FPs = 7.67$ produits par seconde. Pour surdimensionner le système, une marge de sécurité $Mfp = 2\%$ est ajoutée. Le nombre minimum de robots est $NRmin = FPs * (1 + Mfp)/PPas = 7.67 * (1 + 0.02)/2.61 = 3$. Comme une marge de sécurité a été utilisée, le nombre final de robots est $NRf = NRmin = 3$. En s'appuyant sur le nombre de produits dans une boîte, le débit de boîtes est égal à $FBs = FPs/NPb = 7.67/9 = 0.85$ boîtes par seconde. En utilisant la distance entre les boîtes, la vitesse du convoyeur de sortie doit être égale à $Vo = FBs * EB = 0.85 * 0.15 = 0.13$ m/s.

La Figure B.7 montre les résultats de simulation où l'on compare deux stratégies de

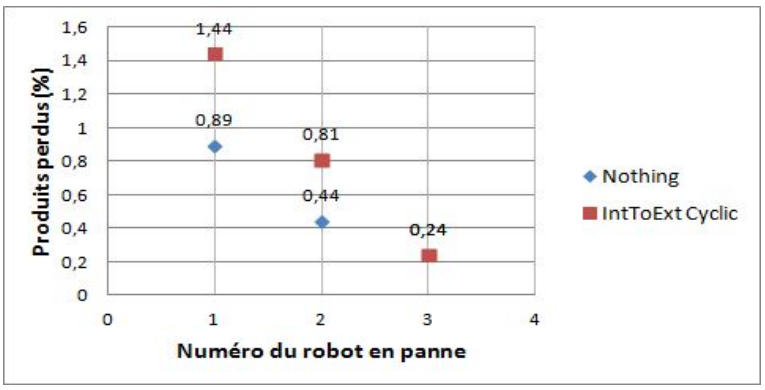

FIGURE B.7 – Produits perdus en fonction du numéro en panne suivant deux stratégies de collaboration pour un système dimensionné au plus juste ayant des convoyeurs en co-courant, avec un flux de produits simples positionnés aléatoirement et une prise simple.

collaboration sur un système dimensionné au plus ayant des convoyeurs en co-courant, avec un flux de produits positionnés aléatoirement et une prise simple suivant différents robots en panne. Tout d'abord nous pouvons constater que, pour les deux stratégies de collaboration, plus le la défaillance est en aval du système moins il y a de produits perdus. De plus, la stratégie *Nothing* permet de perdre moins de produits que la stratégie *IntToExt Cyclic*. Il y aura toujours une perte de produits lorsque qu'un robot subit un dysfonctionnement, même si d'autres robots sont présents pour compenser le problème. En effet, lorsque le robot ne fonctionne plus et que le robot de secours prend le relai, durant un moment, il n'y aura pas de places dans les boîtes présentes devant lui tandis que des produits traverseront sa zone de travail. La différence de vitesses entre les convoyeurs d'entrée et de sortie en est la cause, le convoyeur d'entrée étant plus rapide que celui de sortie.

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : HUMBERT
(avec précision du nom de jeune fille, le cas échéant)

DATE de SOUTENANCE : 25/04/2016

Prénoms : Gaël

TITRE : AUTOMATISATION ET AMÉLIORATION DE PERFORMANCES D'APPLICATIONS PICK & PLACE MULTI-ROBOTS

NATURE : Doctorat

Numéro d'ordre : 2016LYSEI036

Ecole doctorale : Electronique, Electrotechnique, Automatique

Spécialité : Automatique

RESUME : Cette thèse s'inscrit dans le domaine de l'emballage et du conditionnement. À notre connaissance, dans le monde industriel et académique, il n'existe pas d'outils de simulation qui prennent en compte toutes les étapes du cycle de développement afin d'aider au dimensionnement et à l'amélioration des performances d'application pick & place multi-robots. Il existe des outils performants répondant à un seul besoin mais aucun logiciel répondant à tous. L'objectif de ces travaux est le développement d'une méthodologie utilisable, à l'aide d'un unique outil logiciel, tout au long du cycle de création d'une application pick & place. Dans un premier temps, la cinématique des différents robots ainsi que leur environnement sont abordées. Dans un second temps, les stratégies de commande, permettant le travail individuel et collaboratif des robots, sont développées. En dernier lieu, le transfert du programme de la simulation à l'expérimentation est effectué de façon automatisée. A chaque étape, une phase de tests est initiée. Des tests comportementaux pour vérifier le fonctionnement des robots et leur interaction avec leur environnement sont effectués. Des essais en simulation des différents algorithmes sont réalisés suivant différentes configurations d'applications pick & place. Enfin, la comparaison entre les résultats obtenus en simulation et en expérimentation montre la pertinence de la démarche proposée.

MOTS-CLÉS : applications pick & place, stratégies de collaboration, règles de planification individuelles, dimensionnement, outil logiciel, expérimentation.

Laboratoire (s) de recherche : AMPERE - Laboratoire AMPERE (UMR5005)

Directeur de thèse: Xavier Brun

Président de jury : Philippe LUTZ

Composition du jury : Pierre CASTAGNA, Philippe LUTZ, Khalid KOUISS, Mady GUILLEMOT, Minh Tu PHAM, Xavier BRUN