Thèse de doctorat

Pour obtenir le grade de Docteur de l'Université de

VALENCIENNES ET DU HAINAUT-CAMBRESIS

Discipline, spécialité selon la liste des spécialités pour lesquelles l'Ecole Doctorale est accréditée :

Automatique et Informatique Industrielle

Présentée et soutenue par Guillaume, DEMESURE.

Le 08/12/2016, à Valenciennes

Ecole doctorale : Sciences Pour l'Ingénieur (SPI)

Equipe de recherche, Laboratoire : Laboratoire d'Automatique, de Mécanique et d'Informatique Industrielles et Humaines (LAMIH, CNRS UMR 8201)

COORDINATION ET PLANIFICATION DE SYSTÈMES MULTI-AGENTS DANS UN ENVIRONNEMENT MANUFACTURIER

JURY

Président du jury

- M. LOISEAU, Jean-Jacques	Directeur de recherche CNRS, École Centrale de Nantes

Rapporteurs

- M. LOISEAU, Jean-Jacques	Directeur de recherche CNRS, École Centrale de Nantes
- M. SIARRY, Patrick	Professeur, Université Paris-Est Créteil

Examinateurs

- M. LAGHROUCHE, Salah	Maître de Conférences, SET, Belfort
- M. MOULAY, Emmanuel	Chargé de recherche CNRS, XLIM, Poitiers
- M. TRENTESAUX, Damien	Professeur, Université de Valenciennes et du Hainaut-Cambrésis

Directeur de thèse

- M. DJEMAÏ, Mohamed	Professeur, Université de Valenciennes et du Hainaut-Cambrésis

Encadrants

- M. BEKRAR, Abdelghani	Maître de Conférences, Université de Valenciennes et du Hainaut-Cambrésis
- M. DEFOORT, Michael	Maître de Conférences, Université de Valenciennes et du Hainaut-Cambrésis

Coordination et planification de systèmes multi-agents dans un environnement manufacturier.

RESUME : Cette thèse porte sur la navigation d'agents dans un environnement manufacturier. Le cadre général du travail relève de la navigation d'AGVs (véhicules autoguidés), transportant librement et intelligemment leur produit. L'objectif est de proposer des outils permettant la navigation autonome et coopérative d'une flotte d'AGVs dans des systèmes de production manufacturiers où les contraintes temporelles sont importantes. Après la présentation d'un état de l'art sur chaque domaine (systèmes manufacturiers et navigation d'agents), les impacts de la mutualisation entre ceux-ci sont présentés. Ensuite, deux problématiques, liées à la navigation d'agents mobiles dans des environnements manufacturiers, sont étudiées. La première problématique est centrée sur la planification de trajectoire décentralisée où une fonction d'ordonnancement est combinée au planificateur pour chaque agent. Cette fonction permet de choisir une ressource lors de la navigation afin d'achever l'opération du produit transporté le plus tôt possible. La première solution consiste en une architecture hétérarchique où les AGVs doivent planifier (ou mettre à jour) leur trajectoire, ordonnancer leur produit pour l'opération en cours et résoudre leurs propres conflits avec les agents à portée de communication. Pour la seconde approche, une architecture hybride à l'aide d'un superviseur, permettant d'assister les agents durant leur navigation, est proposée. L'algorithme de planification de trajectoire se fait en deux étapes. La première étape utilise des informations globales fournies par le superviseur pour anticiper les collisions. La seconde étape, plus locale, utilise les données par rapport aux AGVs à portée de communication afin d'assurer l'évitement de collisions. Afin de réduire les temps de calcul des trajectoires, une optimisation par essaims particulaires est introduite. La seconde problématique se focalise sur la commande coopérative permettant un rendez-vous d'agents non holonomes à une configuration spécifique. Ce rendez-vous doit être atteint en un temps donné par un cahier des charges, fourni par le haut-niveau de contrôle. Pour résoudre ce problème de rendez-vous, nous proposons une loi de commande à temps fixe (i.e. indépendant des conditions initiales) par commutation permettant de faire converger l'état des AGVs vers une resource. Des résultats numériques et expérimentaux sont fournis afin de montrer la faisabilité des solutions proposées.

MOTS-CLES: Navigation décentralisée d'agents, Systèmes manufacturiers flexibles, Ordonnancement, systèmes multi-agents, planification de trajectoire, rendez-vous d'agents.

Coordination and motion planning of multi-agent systems in manufacturing environment

ABSTRACT: This thesis is focused on agent navigation in a manufacturing environment. The proposed framework deals with the navigation of AGVs (Automated Guided Vehicles), which freely and smartly transport their product. The objective is to propose some tools allowing the autonomous and cooperative navigation of AGV fleets in manufacturing systems for which temporal constraints are important. After presenting the state of the art of each field (manufacturing systems and agent navigation), the impacts of the cross-fertilization between these two fields are presented. Then, two issues, related to the navigation of mobile agents in manufacturing systems, are studied. The first issue focuses on decentralized motion planning where a scheduling function is combined with the planner for each agent. This function allows choosing a resource during the navigation to complete the ongoing operation of the transported product at the soonest date. The first proposed approach consists in a heterarchical architecture where the AGVs have to plan (or update) their trajectory, schedule their product and solve their own conflict with communicating agents. For the second approach, hybrid architecture with a supervisor, which assists agents during the navigation, is proposed. The motion planning scheme is divided into two steps. The first step uses global information provided by the supervisor to anticipate the future collisions. The second step is local and uses information from communicating agents to ensure the collision avoidance. In order to reduce the computational times, a particle swarm optimization is introduced. The second issue is focused on the cooperative control, allowing a rendezvous of nonholomic agents at a specific configuration. This rendezvous must be achieved in a prescribed time, provided by the higher level of control. To solve this rendezvous, a fixed time (i.e. independent of initial conditions) switching control law is proposed, allowing the convergence of agent states towards a resource configuration. Some numerical and experimental results are provided to show the feasibility of the proposed methods.

KEYWORDS: Decentralized agent navigation, Flexible manufacturing systems, Scheduling, Multi-agent systems, Motion planning, Agent rendezvous.

Table des matières

Remerciements

Le travail présenté dans ce mémoire a été effectué au laboratoire LAMIH à l'Université de Valenciennes et du Hainaut-Cambrésis sous la direction de Mohamed Djemaï, encadré par Abdelghani Bekrar et Michael Defoort, en collaboration avec Damien Trentesaux. Je tiens à les remercier très amicalement, avec ma plus grande reconnaissance, pour l'accompagnement dont ils ont fait preuve lors de ces trois années de thèse. Leurs différents conseils m'ont permis d'achever ce travail de thèse dans des conditions optimales. Ils ont fait preuve de rigueur et de recul pour ce sujet de thèse ayant des thématiques variées.

Je tiens aussi à remercier Monsieur Patrick Siarry et Monsieur Jean-Jacques Loiseau d'avoir accepté d'être rapporteur de cette thèse. Leurs lectures précises et minutieuses ainsi que leurs questions en fin de soutenance ont, en plus d'ajouter leur propre expertise, donner une très bonne valeur ajoutée à ces travaux de thèse. Je remercie aussi Monsieur Jean-Jacques Loiseau d'avoir présider le Jury de thèse.

Qu'il me soit permis de remercier Monsieur Salah Laghrouche et Monsieur Emmanuel Moulay pour l'honneur qu'ils m'ont fait en acceptant d'être examinateurs du jury de ma thèse, tout en apportant diverses questions soulignant leur intérêt pour ce sujet de thèse.

Mes remerciements vont aussi à tous les membres du LAMIH pour leur sympathie. Je remercie Marie-Pierre, sans qui je n'aurais pas pu obtenir des résultats expérimentaux. Un grand merci à tous les doctorants ou jeunes docteurs que j'ai côtoyés (Pipit, Mohamed, Gabriel, José, Zahir, Thomas, Salvatore, Ben, Amir, Quentin et tous les autres). Enfin, une dédicace particulière à Monsieur Thierry-Marie Guerra, alias TMG, directeur du LAMIH, sans qui je n'aurais jamais choisi la filière de l'automatique. Je le remercie aussi de m'avoir accepté au sein du laboratoire.

Mes derniers remerciements sont pour ma famille et ma compagne pour m'avoir supporté lors de ces trois années de thèse.

Notations

Notations générales :

$\mathbb{R}$: ensemble des nombres réels.

$\mathbb{R}_+$: ensemble des nombres réels positifs ou nuls.

$\mathbb{R}_+^*$: ensemble des nombres réels strictement positifs.

$\mathbb{R}^n$: espace vectoriel de dimension n construit sur le corps des réels.

$\mathbb{N}$: ensemble des nombres entiers naturels.

$[a, b]$: intervalle fermé de $\mathbb{R}$ d'extrémités a et b.

$]a, b[$: intervalle ouvert de $\mathbb{R}$ d'extrémités a et b.

$[a, b[$: intervalle semi–ouvert de $\mathbb{R}$ d'extrémités a et b.

$t \in \mathbb{R}$: variable temporelle.

$T_{init} \in \mathbb{R}$: instant initial fixé.

$t_{fin} \in \mathbb{R}$: instant final.

$\dot{x} = \frac{dx}{dt}$: dérivée de la variable x par rapport au temps.

$\ddot{x} = \frac{d^2x}{dt^2}$: seconde dérivée de la variable x par rapport au temps.

$x^{(j)} = \frac{d^jx}{dt^j}$: $j^{ème}$ dérivée de la variable x par rapport au temps.

x^T : transposé du vecteur x.

$x \in \mathbb{R}^n$: vecteur de composantes x_j avec $1 \leq j \leq n$.

$|.|$: valeur absolue d'un nombre réel.

$\|.\|$: norme euclidienne sur $\mathbb{R}^n$.

$\det(A)$: déterminant de la matrice A.

$\|A\|$: norme euclidienne de la matrice A.

I_n : matrice identité de $\mathbb{R}^{n\times n}$.

0_n : matrice nulle de $\mathbb{R}^{n\times n}$.

Notations communes aux chapitres 3 **et** 4 :

$\mathcal{R}$: ensemble des ressources (machines) $\mathcal{R} = \{1, ..., N_c\}$.

$\mathcal{A}$: ensemble des AGVs (agents) $\mathcal{A} = \{1, ...N_i\}$.

$\mathcal{J}$: ensemble des produits $\mathcal{J} = \{1, ..., N_l\}$.

$\mathcal{O}_l$: ensemble séquencé d'opérations $\mathcal{O}_l = \{1, ..., N_n\}$ à effectuer par le produit $l \in \mathcal{J}$.

$\mathcal{R}_{nl}$: ensemble de ressources pouvant effectuer l'opération n du produit l.

opt_{nl} : temps de traitement de l'opération n du produit l (à une ressource $b \in \mathcal{R}_{nl}$).

odd_{nl} : échéance de l'opération n du produit l.

ct_{nlb} : temps pour achever l'opération n du produit l à la ressource $b \in \mathcal{R}_{nl}$.

q_b : position $(x_b, y_b)^T$ de la ressource $b \in \mathcal{R}$ dans l'environnement manufacturier.

w_b : temps d'attente actuel à la ressource $b \in \mathcal{R}$.

q_i : position actuelle de l'agent i.

v_i : vitesse actuelle de l'agent i.

IP_i : indice actuel de performance (individuel) de l'agent i (pour l'opération n de son produit l).

$\mathcal{N}_i$: ensemble des voisins actuel de l'agent i (agents qui communiquent avec celui–ci).

$\mathcal{HP}_i$: ensemble des voisins actuel de l'agent i ayant un niveau de priorité plus haut que celui–ci.

R_{com} : portée de communication pour tous les agents $i \in \mathcal{A}$.

d_{safe} : distance de sécurité à respecter pour éviter les collisions inter–agents.

$T_{i,init}$: instant initial lorsque l'agent i démarre (ou a démarré) de la ressource.

$T_{i,fin}$: instant final actuel que l'agent i planifie pour atteindre la prochaine ressource.

T_{i,upt_i} : instant où la trajectoire de l'agent i est mise à jour.

Notations spécifiques au chapitre 3 :

upt_i : nombre de mises à jour de la trajectoire de l'agent i.

$\mathcal{HPD}_i$: ensemble d'agents voisins de niveau de priorité plus haut que l'agent i et se dirigeant vers la même ressource que celui–ci, $\mathcal{HPD}_i \subset \mathcal{HP}_i$.

$q_{ib}(t, T_{i,upt_i})$: trajectoire de l'agent i planifiée à l'instant $T_{i,upt}$ vers la ressource $b \in \mathcal{R}_{nl}$.

TT_{ib} : temps de transport actuel pour atteindre la ressource $b \in \mathcal{R}_{nl}$

$q_i(t, T_{i,upt_i})$: trajectoire finale de l'agent i planifiée à l'instant $T_{i,upt}$.

$cr_i(upt_i)$: ressource choisie par l'agent i à la mise à jour upt_i.

$\mathcal{G}_i$: ensemble (ou groupe) actuel des agents connectés avec l'agent i par liens de communication, directs (par le voisinage) et indirects (par le biais des voisins).

OP_i : performances comparatives actuelles de l'agent i par rapport à son groupe $\mathcal{G}_i$.

sf_i : variable représentant le niveau de priorité actuel de l'agent i.

Notations spécifiques au chapitre 4 :

wp_{ibk} : temps d'attente supplémentaire (fictif) dû aux agents $j \in \mathcal{HP}_i$ se dirigeant vers la même ressource b que l'agent i.

T_c : période de mise à jour de la trajectoire.

τ_k : instant de (i.e. lorsque les trajectoires des agents sont) mise(s) à jour $\tau_k = \tau_0 + k \cdot T_c, k \in \mathbb{N}$, τ_0 est l'instant de départ de la production.

m : nombre minimum de mises à jour (des trajectoires des agents) entre deux ordonnancement successifs $m \in \mathbb{N}^*$.

sk_i : nombre de mises à jour depuis que l'agent i a ordonnancé son produit $sk_i \in \mathbb{N}$.

sch_{ik} : variable binaire mise à 1 si l'agent i est autorisé à ordonnancer son produit à l'instant de mise à jour de trajectoire τ_k, 0 sinon.

$q^*_{ib}(t, \tau_k)$: trajectoire directe de l'agent i vers la resource $b \in \mathcal{R}_{nl}$ à l'instant τ_k.

$\widetilde{q}_{ib}(t, \tau_k)$: trajectoire intuitive de l'agent i vers la resource $b \in \mathcal{R}_{nl}$ à l'instant τ_k.

$q_i(t, \tau_k)$: trajectoire finale de l'agent i planifiée à l'instant τ_k.

cr_{ik} : resource choisie par l'agent i à l'instant τ_k, $cr_{ik} \in \mathcal{R}_{nl}$.

TT^*_{ibk} : temps de transport correspondant à la trajectoire directe de l'agent i (i.e. $q^*_{ib}(t, \tau_k)$).

$\widetilde{TT}_{ik}$: temps de transport correspondant à la trajectoire intuitive de l'agent i (i.e. $\widetilde{q}_i(t, \tau_k)$).

TT_{ik} : temps de transport correspondant à la trajectoire finale planifiée par l'agent i (i.e. $q_i(t, \tau_k)$).

$\mathcal{CF}_{ik}$: ensemble de conflits (de collision) de l'agent i à l'instant τ_k (i.e. $\mathcal{CF}_{ik} = \{\mathcal{C}_{ijb}\}$).

$\mathcal{C}_{ijb}$: conflit que l'agent i doit éviter avec l'agent j lorsque la ressource b est choisie (i.e. $\mathcal{C}_{ijb} = (j, b, I_{ij}, cd_{ij})$).

I_{ij} : intervalle de temps du conflit $\mathcal{C}_{ijb}$.

cd_{ij} : degré du conflict $\mathcal{C}_{ijb}$, $cd_{ij} \in \mathbb{N}^*$.

Notations spécifiques à la commande coopérative :

N : nombre d'agents suiveurs dans la flottille.

$\mathcal{G}$: graphe des communications inter–agents qui consiste à un ensemble de noeuds $\mathcal{V} = \{1, 2, \cdots, N\}$ et de lien $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$.

A : matrice adjacente du graphe $\mathcal{G}$.

L : matrice Laplacienne du graphe $\mathcal{G}$.

$q_n(t)$: configuration du robot suiveur n pour le modèle non holonome.

$u_n(t)$: entrée du robot n, $u_n(t) = (v_n(t), w_n(t))^T$ où $v_n(t)$ et $w_n(t)$ sont respectivement les vitesses linéaire et angulaire.

$q_0(t)$: configuration cible correspondant au meneur virtuel 0.

ξ_n : configuration du robot n sous forme chaînée.

ν_n : entrée du robot n sous forme chaînée.

ξ_0 : configuration du meneur 0 sous forme chaînée.

ξ_0^d : configuration intermédiaire correspondant à la configuration q_0^d définie à l'aide de L'Eq. (5.5.1).

T_1, T_2 : temps de commutation de la commande ν_n, indépendants des conditions initiales.

T_{max} : temps de stabilisation désiré, indépendant des conditions initiales.

sign(a) : fonction signe réelle définie par :

$$\text{sign}(a) = \begin{cases} -1 & \text{si} \quad a < 0 \\ 1 & \text{si} \quad a > 0 \end{cases}$$

Par extension, sign(x) où $x = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$ sera le vecteur de composantes sign(x_j).

Acronymes

- AGV(s) : Véhicule(s) autoguidé(s) (*Automated Guided Vehicle*).
- FMS : Système manufacturier flexible (*Flexible manufacturing system*).
- PSO : Optimisation par essaim particulaire (*Particle Swarm Optimization*).
- G.O.R.P : Gestion des Opérations et des Ressources de Production.

Introduction générale

Ce travail de doctorat a été préparé au sein du département automatique du Laboratoire d'Automatique, de Mécanique et d'Informatique Industrielles et Humaines (LAMIH, UMR CNRS 8201). Ce département est divisé en deux thèmes : RObustesse et Complexité (ROC) et Systèmes Intelligents Coopérants (SIC). Mon travail de thèse s'inscrit dans l'intersection de ces deux thèmes. En effet, l'étude des systèmes multi–agents se situe à la fois dans un cadre orienté contrôle, mais aussi dans une optique où les agents doivent coopérer. Avec le mot "agent", nous désignons n'importe quel système ayant des capacités de perception, de calcul et de communication, tel que les robots mobiles, appelés aussi véhicules autoguidés (de l'anglais "Automated Guided Vehicles") dans les applications manufacturières. Le contexte auquel mon travail est appliqué se rapporte aux systèmes de production manufacturiers où les contraintes temporelles sont importantes. De ce fait, nous aimerions souligner la pluridisciplinarité de ces travaux, où des aspects de contrôle des systèmes multi–agents, ainsi que la recherche opérationnelle, sont étudiés. De plus, l'intérêt majeur est aussi d'arriver à une mutualisation entre les deux domaines de recherche, en combinant leurs outils respectifs. Il est donc primordial d'introduire le contexte manufacturier, où les agents seront plus contraints temporellement par rapport à d'autres contextes.

Contexte

Les systèmes de production manufacturiers sont de plus en plus étudiés du fait de l'évolution grandissante des besoins industriels. De nos jours, ces besoins deviennent de plus en plus stricts, car on recherche, à court terme, une réactivité suffisante face aux événements inattendus, et à long terme, des moyens pour s'adapter à l'évolution des besoins du marché. Pour répondre à ces besoins, les industriels peuvent être amenés à reconsidérer leurs systèmes de production en appliquant de nouvelles méthodologies. Parmi ces méthodologies, il y a plusieurs intérêts à considérer les véhicules autoguidés (AGVs) dans les systèmes manufacturiers. En effet, l'évolution technologique dans les domaines de la mécatronique, de l'ingénierie informatique et des technologies d'information et de communication (ICT) permet d'utiliser ces AGVs dans des applications manufacturières, tout en réduisant leur coût de déploiement. Cette évolution amène les industries à considérer ces AGVs comme une solution possible pour améliorer la réactivité et la flexibilité de leur production. Par exemple, l'entreprise

FIGURE 1 – Les AGVs Kiva [Wurman et al., 2008], utilisés dans les entrepôts de l'entreprise Amazon

Amazon [Gilmour, 2003] utilise des AGVs dans certains entrepôts comme le montre la Fig. 1. En effet, les AGVs permettent, par nature, d'élargir le cadre des comportements réactifs dans le routage dynamique, lorsqu'ils naviguent librement dans les ateliers de production.

Dans ce travail de thèse, on suppose que les AGVs (robots ou agents mobiles) naviguent librement dans le système manufacturier. Dans ce but, les efforts de recherche sur les systèmes manufacturiers doivent être intensifiés, afin de pouvoir considérer cette navigation libre. De plus, le système de production doit fournir aux AGVs des missions qu'ils sont capables de remplir dans des délais imposés, et ce quelques soient les incertitudes rencontrées (ordre de fabrication urgent, panne d'une ressource...). De nos jours, les AGVs deviennent de plus en plus complexes, intégrant des capacités de perception, de communication et d'adaptation à diverses situations, que l'on peut rencontrer dans les systèmes de production. Les AGVs visent aussi des exigences plus importantes en termes de robustesse, d'ergonomie, de sécurité et peuvent coopérer, afin de résoudre plus efficacement et plus rapidement leur mission respective. Cependant, les capacités des AGVs restent limitées, forçant le système de production à prendre en compte ces limites (faisabilité, embarquabilité...).

Objectifs et contributions de la thèse

Les objectifs liés aux AGVs et au système de production manufacturier sont étroitement liés. En effet, l'objectif principal des recherches effectuées sur les systèmes manufacturiers consiste à améliorer les performances globales de la production à l'aide d'outils de contrôle haut-niveau. Dans cette optique, le système manufacturier peut agir sur les décisions des AGVs. D'un point de vue local, les AGVs ont pour objectif d'accomplir les tâches assignées par le haut–niveau du système manufacturier. Ils doivent aussi coopérer entre eux pour éviter les différents conflits qu'ils peuvent rencontrer durant leur mission. D'une manière générale, il est possible de traiter les objectifs séparément, car les recherches sur les systèmes de production sont plutôt centrées sur le contrôle haut–niveau. A l'inverse, l'étude des AGVs est commune en robotique mobile et est focalisée sur du contrôle de plus bas niveau. La Fig. 2 montre clairement ce positionnement, ainsi que les efforts de recherche sur lesquels ces domaines se focalisent.

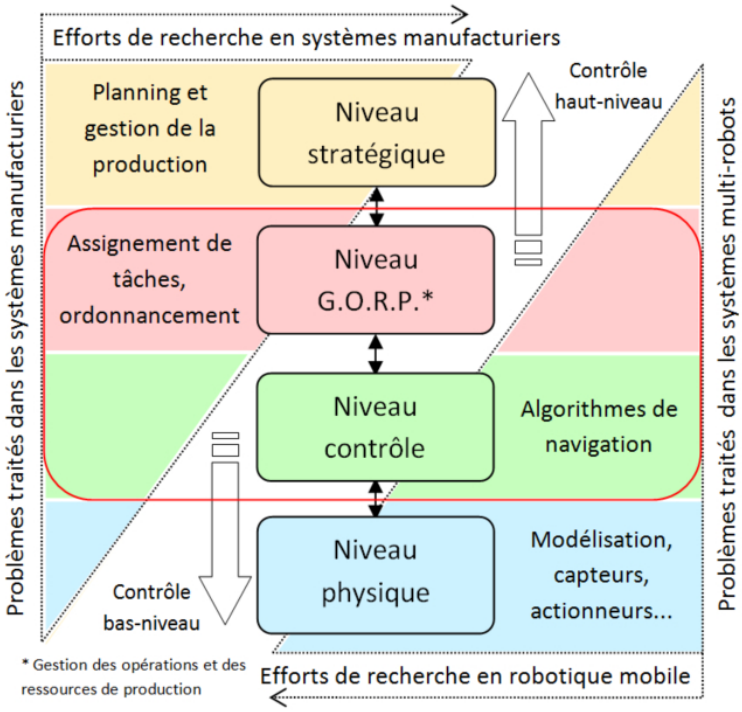


FIGURE 2 – Efforts de recherche dans les domaines de robotique mobile et de systèmes manufacturiers

Dans ce mémoire, nous serons amenés à mettre en place des outils permettant la navigation autonome et coopérative d'une flotte d'AGVs dans un environnement manufacturier. Par conséquent, il est nécessaire de mentionner les spécificités des systèmes manufacturiers traités, mais aussi celles des AGVs auxquels on s'intéresse. Les spécificités liées aux systèmes manufacturiers sont :

- On s'intéresse aux systèmes de productions flexibles. La notion de flexibilité sera introduite dans la Section 1.1.1.
- Un cahier des charges est fourni pour chaque produit à effectuer. Chaque cahier des charges ne sera pas remis en cause et sera supposé faisable.
- Comme les approches seront focalisées sur la navigation en elle-même, on s'intéressera aux moyens permettant de compléter une seule tâche (ou opération) plutôt qu'un ensemble de tâches.
- Il n'y a pas d'obstacles (fixes) dans l'environnement de production. Il sera supposé que les ressources sont bien positionnées, afin que les AGVs ne doivent qu'éviter les collisions entre eux.
- Les incertitudes du système, telles que les pannes de ressources, ne seront pas considérées. Cependant, plusieurs pistes sur les moyens de les prendre en compte seront discutées.

Les spécificités liées aux AGVs sont les suivantes :

- Les AGVs ne transportent qu'un produit à la fois. De plus, un produit n'est transporté que par un seul AGV.
- Les AGVs sont soumis à des contraintes physiques (vitesse limitée, par exemple).
- Les AGVs peuvent échanger des informations sur leur position et/ou leurs intentions.
- La puissance disponible des AGVs en termes de communication et de calcul est limitée.
- Selon l'architecture de pilotage choisie, les conflits entre les AGVs peuvent être résolus localement ou globalement.
- La fonction de planification de trajectoire est résolue de manière décentralisée : chaque AGV doit calculer sa propre trajectoire.
- Les trajectoires planifiées sont supposées être parfaitement suivies par les agents.

La plupart des spécificités mentionnées ci–dessus permettent de simplifier le problème. En effet, chaque partie est individuellement complexe, donc traiter l'ensemble du problème l'est encore plus. De plus, certaines fonctionnalités de la navigation seront combinées avec des outils liés à la navigation des AGVs. Ainsi, ces spécificités permettent de se focaliser sur les interactions entre les AGVs et le système de production, mais aussi de combiner les outils de manière plus simple.

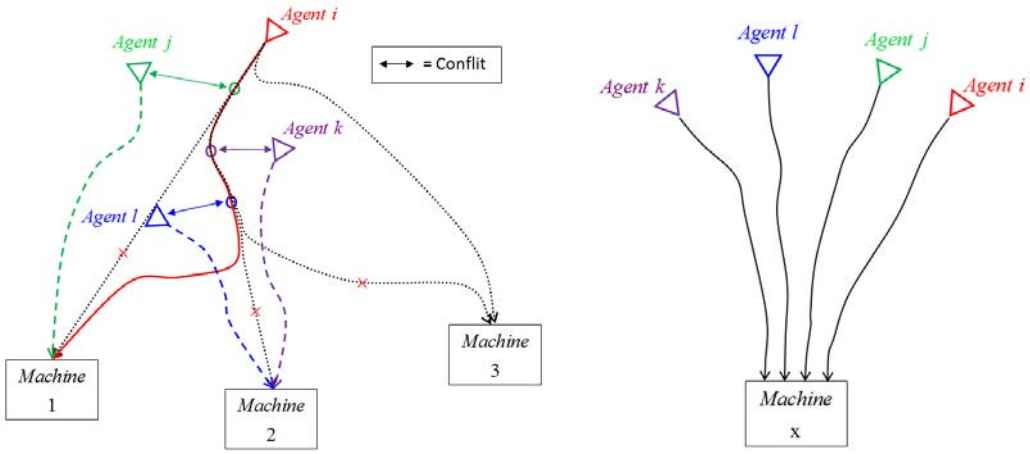
Afin de répondre à leur spécificités, les AGVs doivent être capables de calculer ou de mettre à jour leur trajectoire durant la navigation, tant que la tâche assignée n'est pas accomplie. De plus, la mise à jour de la trajectoire peut aussi dépendre d'autres fonctions de l'AGV en relation avec la production (par exemple, l'ordonnancement) du produit qu'il transporte. Le cadre général du travail proposé dans ce mémoire relève donc de la navigation d'AGVs, transportant librement et intelligemment leur produit. Cependant, l'accent doit aussi être mis sur les spécificités des approches proposées, impliquant des problématiques liées à la mutualisation et qui nécessitent la conception de méthodes combinant les outils de chaque domaine de recherche.

L'objectif de cette thèse est principalement focalisé sur la navigation des AGVs dans un système de production manufacturier. Plus précisément, on s'intéresse aux moyens pour atteindre une mutualisation entre les deux domaines de recherche mentionnés auparavant. On sera plutôt focalisé sur les interactions entre le haut niveau correspondant au système de production et le bas niveau correspondant au contrôle des AGVs (partie encadrée en rouge sur la Fig. 2). De plus, certains outils de chaque domaine pourront être combinés de façon cohérente pour atteindre un certain niveau d'intelligence des AGVs, pouvant ainsi améliorer leur performance.

Dans cette thèse, les contributions portent sur deux problèmes de navigation, illustrés par la Fig. 3 :

1. La génération de trajectoire pour les agents mobiles, en respectant les contraintes liées à un environnement manufacturier, où une fonction d'ordonnancement est incluse dans l'algorithme de navigation. Cette fonction permettra de choisir la ressource lors de la navigation, afin de faire face à différents conflits (Fig. 3–(a)).

2. La commande coopérative permettant un rendez-vous d'agents à un endroit spécifique. Ce rendez–vous devra être atteint en un temps donné par un cahier des charges, fourni par le haut–niveau (Fig. 3–(b)).



(a) Planification de trajectoire (b) Commande coopérative

FIGURE 3 – Illustration des contributions de la thèse

Organisation du mémoire de thèse

Le chapitre 1 donne un état de l'art sur les systèmes de production manufacturiers et les outils de navigation des AGVs. Les principaux concepts nécessaires à cette thèse seront introduits pour chaque domaine. Pour les systèmes manufacturiers, les architectures de pilotage seront présentées avec leurs spécificités. Pour les aspects de robotique sur les systèmes multi–agents, on y retrouvera les concepts de commande coopérative et de planification de trajectoire. Ensuite, les impacts et les problématiques liés à la mutualisation seront énoncés.

Dans le chapitre 2, le problème de planification de trajectoire des AGVs et la paramétrisation des trajectoires seront présentés afin d'introduire les approches traitées dans les chapitres 3 et 4. Pour le problème de planification de trajectoire, nous introduirons l'objectif des AGVs, ainsi que les principales hypothèses. La combinaison de la planification de trajectoire avec une fonction d'ordonnancement sera

introduite et explicitée. De plus, les architectures de pilotage utilisées seront présentées. Enfin, les courbes splines, utilisées pour mettre les trajectoires sous forme paramétrique dans les chapitres 3 et 4, seront introduites.

Dans le chapitre 3, on trouvera une première solution du problème de planification de trajectoire des AGVs dans un système manufacturier flexible. Pour cette solution, une architecture hétérarchique est utilisée, où les AGVs devront planifier une trajectoire, ordonnancer leur produit pour l'opération en cours et résoudre les conflits par eux–mêmes. Les fonctions de planification de trajectoire et d'ordonnancement seront combinées afin de remplir l'objectif individuel de chaque AGV d'une manière efficace. La mise à jour des trajectoires des AGVs ne se fera qu'en présence d'évènements tels que les conflits (pour éviter les collisions par exemple). Pour résoudre les conflits entre AGVs, une négociation par approche sociale sera adaptée, afin de répondre aux spécificités de notre problème.

Dans le chapitre 4, une seconde solution est proposée en utilisant une architecture hybride, à l'aide d'un superviseur permettant d'assister les agents dans leurs tâches. Après avoir décrit le problème et les rôles du superviseur, un algorithme en deux étapes est introduit pour chaque AGV. La première étape utilise des informations globales fournies par le superviseur pour anticiper les collisions. La seconde étape, plus locale, utilise les données par rapport aux AGVs à portée de communication, afin de permettre l'évitement de collisions. Afin de réduire le temps de calcul des trajectoires, nécessaire au vu d'une planification de trajectoires en deux étapes, une résolution du problème de planification à l'aide de méta heuristiques est introduite.

Le chapitre 5 présente une commande coopérative où les AGVs se donnent rendez-vous à une certaine configuration. Ce rendez–vous devra se faire en un temps donné, c'est–à–dire à une date imposée par un cahier des charges de production fourni par le haut–niveau du système manufacturier. Dans ce chapitre, les robots (AGVs) seront représentés par des modèles non holonomes. L'objectif est de faire converger l'état des AGVs vers une configuration représentée par une ressource en un temps fixe. On y énoncera le concept de commande à temps fixe, où le rendez-vous doit avoir lieu à une certaine date, indépendamment des conditions initiales des AGVs, permettant de répondre au cahier des charges de production, qui impose des contraintes temporelles.

Dans le chapitre 6, les résultats numériques des chapitres 3, 4 et 5 seront fournis avec les explications adéquates, permettant de souligner les avantages de chaque approche proposée. On y trouvera également une comparaison entre les deux solutions proposées dans les chapitres 3 et 4. Cette comparaison permet notamment de montrer comment les inconvénients de l'approche du chapitre 3 sont comblés par celle du chapitre 4. De plus, des résultats expérimentaux seront fournis, afin de montrer la faisabilité des solutions proposées. Enfin, une discussion sera proposée sur les éléments requis permettant de déployer des AGVs en industrie. Cette discussion permettra aussi d'orienter les principales perspectives de cette

thèse.

Ce mémoire se termine par une conclusion générale, rappelant toutes les contributions pour les techniques proposées sur la planification de trajectoire et la commande coopérative. De plus, suite à la discussion introduite dans le chapitre 6, les perspectives à court terme et à plus long terme seront énoncées et explicitées.

Production scientifique personnelle

Conférences nationales et internationales avec comité de lecture :

- **G. Demesure**, M. Defoort, A. Bekrar, D. Trentesaux & M. Djemaï. *Cooperation mechanisms in Multi-Agent Robotic Systems and their use in distributed manufacturing control : Issues and literature review.* In Industrial Electronics Society, IECON 2014-40th Annual Conference of the IEEE, pp. 2538-2543, Oct. 2014, Dallas, USA.

- **G. Demesure**, M. Defoort, A. Bekrar, D. Trentesaux & M. Djemaï. *Planification de trajectoire et ordonnancement d'agents mobiles : application aux systèmes de production.* JDMACS'15, Juin 2015, Bourges, France.

Revues internationales avec comité de lecture :

- M. Defoort, **G. Demesure**, Z. Zuo, A. Polyakov & M. Djemaï. *Fixed-time stabilization and consensus of nonholonomic systems.* IET Control Theory & Applications, 2016.

- **G. Demesure**, M. Defoort, A. Bekrar, D. Trentesaux & M. Djemaï. *Navigation Scheme with Priority-Based Scheduling of Mobile Agents : Application to AGV-Based Flexible Manufacturing System.* Journal of Intelligent & Robotic Systems, 82(3-4), pp. 495-512, 2016.

- M. Defoort, A. Polyakov, **G. Demesure**, M. Djemai & K. Veluvolu. *Leader-follower fixed-time consensus for multi-agent systems with unknown non-linear inherent dynamics.* IET Control Theory & Applications, 9(14), pp. 2165-2170, 2015.

Chapitre 1

Problématique et état de l'art

1.1 Introduction

Le déploiement des systèmes autoguidés (AGVs) en environnement industriel devient de plus en plus commun. Les AGVs sont des systèmes autonomes (sans conducteur), permettant un transport de matières (produits, outils, palettes, conteneurs, ...) dans des environnements intérieurs (entrepôts, usines de fabrication ...) ou extérieurs (ports, gares, ...). Leur utilisation dans des systèmes de production manufacturiers apporte de nombreux avantages. Cependant, il est nécessaire de considérer des outils, permettant notamment leur navigation, qui doivent être adaptés au contexte industriel. La plupart des outils de navigation, tels que la planification de trajectoire ou les commandes coopératives, sont étudiés dans le domaine de la robotique mobile. Afin d'atteindre une mutualisation entre les domaines de la robotique mobile et de l'étude des systèmes manufacturiers (recherche opérationnelle), les outils de chaque domaine doivent impérativement s'harmoniser, voire se combiner. De ce fait, il est nécessaire de présenter les deux domaines pour introduire les différents concepts et outils. L'objectif principal de ce chapitre est donc d'introduire ces concepts pour chaque domaine, tout en présentant les différents travaux proposés dans la littérature. Nous nous intéresserons ensuite à la mutualisation entre ces domaines, en présentant les impacts de cette mutualisation, qui impliquent différentes problématiques.

1.1.1 Les systèmes manufacturiers

De nos jours, les entreprises ont de fortes contraintes liées aux besoins industriels. L'évolution des besoins industriels a de nombreux impacts sur l'étude des systèmes de production manufacturiers, qui doivent améliorer leurs performances, tout en étant soumis à ces contraintes. Afin de suivre ce contexte industriel changeant, une feuille de route a été définie par la communauté IMS2020 [Rolstadas, 2010] concernant les systèmes manufacturiers intelligents (IMS) du futur [Cardin et al., 2016].

Sur cette feuille de route, trois prescriptions ont été mises en avant, afin d'orienter les recherches sur les systèmes de production :

- **Production rapide et adaptative, centrée sur utilisateur :** Cette prescription stipule que les systèmes de production doivent avoir la capacité de répondre aux besoins des clients, en termes de délais et de personnalisation des produits et/ou service qu'ils demandent. Ce qui, de façon implicite, inclut la capacité d'être réactif et de s'adapter aux besoins du marché.

- **Chaîne de valeur hautement flexible permettant d'organiser les systèmes de production de différentes manières :** Cette prescription impose que les interactions entre les différentes entités du système doivent permettre une auto-organisation. Ceci implique que les systèmes doivent intégrer un certain niveau de flexibilité, en termes d'architecture de pilotage, mais aussi en termes d'infrastructure physique, pour pouvoir fabriquer de nouveaux types de produits.

- **Production durable prenant en compte les changements culturels des industries :** Cette dernière prescription est en relation avec l'énergie, mais aussi avec le rapport entre l'industrie et son environnement culturel.

Afin de suivre ces prescriptions, les entreprises doivent reconsidérer leurs stratégies d'un point de vue financier, mais aussi en termes de planification de production. Cette reconsideration a de grandes répercussions sur les décisions prises par les niveaux de contrôle inférieurs. La Fig. 1.1, se basant sur la norme ISA95 [Bolton et Tyler, 2008], positionne ces différents niveaux et les interactions entre ceux-ci, afin d'illustrer ces propos.

Notons que, dans cette figure, le niveau pilotage englobe les niveaux tactique et opérationnel. Comme mentionné préalablement, l'étude sera focalisée sur les niveaux pilotage et contrôle. Plus particulièrement, on s'intéressera aux fonctions d'ordonnancement, de conduite, de routage et aussi aux architectures de pilotage supportant ces fonctions. Par conséquent, nous allons, dans un premier temps, introduire les rôles de ces fonctions et étudier les architectures de pilotage existantes.

Le pilotage des systèmes de production manufacturiers

Le pilotage des systèmes de production a été défini à plusieurs reprises [Wyns, 1998, Trentesaux, 2002, Leitao, 2004]. La définition du pilotage de [Trentesaux, 2002] est ici retenue et adaptée au contexte actuel :

> *Le **pilotage** consiste à décider dynamiquement des commandes pertinentes à donner à un système soumis à l'incertitude, pour atteindre un objectif donné, décrit en termes de maîtrise de performances.*

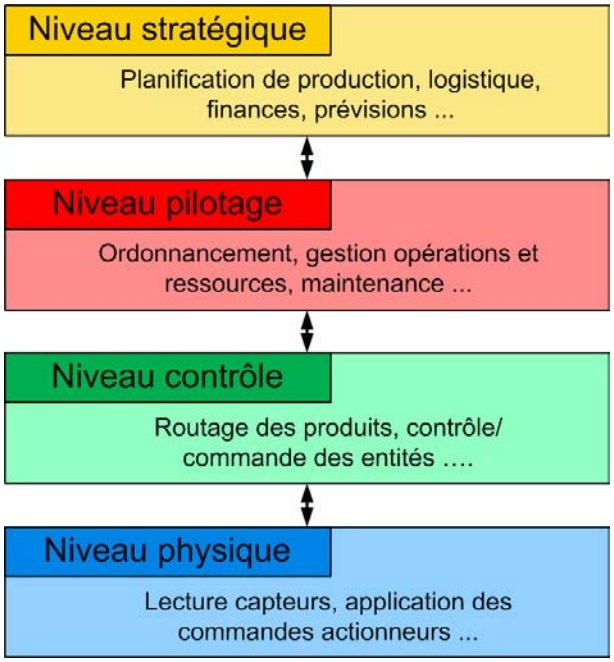


FIGURE 1.1 – Les niveaux de contrôle d'un système de production

Initialement, le terme *perturbation* était utilisé à la place d'*incertitude*. Cependant, le terme *incertitude* est plus générique par rapport au contexte changeant, afin de considérer en plus des perturbations (e.g. les pannes machine), les contraintes de la production (e.g. personnalisation des produits) imposées par le niveau stratégique. Dans cette définition, le terme "système" concerne les êtres biologiques tels que les hommes, mais aussi les êtres artificiels, tels que les robots, machines, réseaux d'ordinateurs. Le pilotage est étudié dans plusieurs domaines, mais sera étudié ici dans le domaine des systèmes de production.

Le niveau de pilotage permet de piloter le système par rapport aux stratégies de l'entreprise. Il comporte une multitude de fonctions telles que l'ordonnancement, la conduite, la maintenance, la gestion des stocks, et autres [Scholten, 2007]. La fonction d'ordonnancement est directement liée aux performances globales et à la réactivité du système de production. Dans la suite de cette thèse, cette seule fonction sera étudiée, car son rôle a plus d'importance dans la production par rapport aux autres fonctions [Graves, 1981]. Celle–ci a été définie dans la littérature dans de nombreux travaux [Pinedo, 2012, Conway et al., 2012, Tangour, 2007]. En se basant sur ces définitions, nous proposons la nôtre, qui sera plus adaptée au contexte de cette thèse, comme suit :

> *L'**ordonnancement** consiste à déterminer la date à laquelle une ou plusieurs tâches seront effectuées et à leur allouer des ressources, afin d'optimiser un ou plusieurs objectifs, en respectant les contraintes de production.*

En comparant les définitions du pilotage et de l'ordonnancement, on remarque bien le rôle majeur de la fonction d'ordonnancement sur le pilotage du système de production. On peut agir sur l'ordonnancement afin d'optimiser la production selon des critères choisis, et ainsi améliorer l'efficacité du système. L'ordonnancement peut être statique ou dynamique. Lorsqu'il est statique, l'ensemble des tâches, séquencé ou non, doit être ordonnancé, c'est-à-dire qu'il faut trouver les dates de début de chaque tâche et allouer les ressources pour celles–ci avant que la production de ce produit ne commence. A l'inverse, il est dynamique lorsque l'on doit ordonnancer les tâches une à une, quand celles–ci se présentent [Fattahi et Fallahi, 2010], ce qui permet d'inclure des variations de certains paramètres (par exemple, la date d'arrivée à une ressource) ou des changements de l'objectif de production.

L'ordonnancement de produit est un problème très complexe et a été défini comme NP–difficile [Lenstra et Kan, 1981] d'une manière générale. Une multitude de travaux ont été proposés sur l'optimisation de cet ordonnancement avec des types de systèmes de production spécifiques ou avec des objectifs différents. Les objectifs de production souvent rencontrés sont :

- Minimiser le temps pour compléter le(s) produit(s), appelé *makespan*. Cet objectif permet d'achever la production le plus rapidement possible [Liu et al., 2005, Rajendran et Ziegler, 2004].
- Compléter le produit juste à temps, ce qui permet de considérer la limitation de la zone de stockage des produits dans les entrepôts de l'entreprise [Cho et Lazaro, 2010, Rey et al., 2013].
- Minimiser l'utilisation des machines pour plusieurs raisons, notamment la consommation d'énergie [Pach et al., 2014b].

D'ailleurs, certains travaux considèrent plusieurs objectifs de façon simultanée pour améliorer les performances de production [Adibi et al., 2010, Ho et Tay, 2008, Ishibuchi et Murata, 1998]. Le pilotage des systèmes de production dépend de son architecture, qui intègre les prises de décision de chacune de ses fonctions. Il est donc préférable de présenter ces architectures de pilotage.

Dans le cadre de cette thèse, l'objectif de production sera de minimiser le temps pour effectuer les opérations des produits. Ce choix se justifie par les impacts associés à la mutualisation. En effet, nous verrons dans la Section 1.2 que l'avantage majeur est de réduire les temps de transport, afin d'améliorer la cadence de production. De plus, nous verrons qu'il est difficile d'achever le produit juste à temps, au vu des différents conflits rencontrés lors de la navigation. Par conséquent, l'objectif annoncé semble plus cohérent pour les problèmes traités.

Les architectures de pilotage

Les architectures de pilotage ont été définies comme une description de la composition et de la structure d'un système de pilotage [Senehi et Kramer, 1998, Bongaerts et al., 1995, Leitao, 2004]. Ces architectures sont de deux types : centralisée et décentralisée (ou distribuée). Les architectures **centralisées** considèrent la production d'un point de vue global. Cependant, à cause de l'évolution

de la complexité des systèmes manufacturiers, il devient difficile, voire impossible, de trouver une station centrale suffisamment puissante afin de piloter tous les éléments du système. De plus, cette complexité augmente exponentiellement par rapport au nombre d'éléments à piloter et le système a de grandes difficultés à traiter les incertitudes à cause de sa faible réactivité. Les architectures **distribuées**, bien que plus complexes en structure et en organisation, sont plus prometteuses, car les capacités décisionnelles sont divisées entre certains éléments du système. Dans les systèmes de production flexibles, la fonction d'**ordonnancement** est souvent supportée par des éléments proches du système physique, comme les robots, produits actifs et/ou ressources actives [Sallez et al., 2010]. Les éléments qui ont des capacités de pilotage sont appelés *entités*, définies par [Trentesaux, 2009] de la manière suivante :

> *Le terme **entité** est un terme générique faisant référence à une unité autonome capable de communiquer, prendre des décisions et agir.*

Au vu des inconvénients d'une structure centralisée, les architectures distribuées sont plus viables afin de répondre aux prescriptions IMS2020 préalablement discutées. Ces architectures regroupent l'ensemble des entités du système et les relations entre elles, et ont été divisées en trois classes par [Trentesaux, 2009], comme le montre la Fig. 1.2 extraite de cette source.

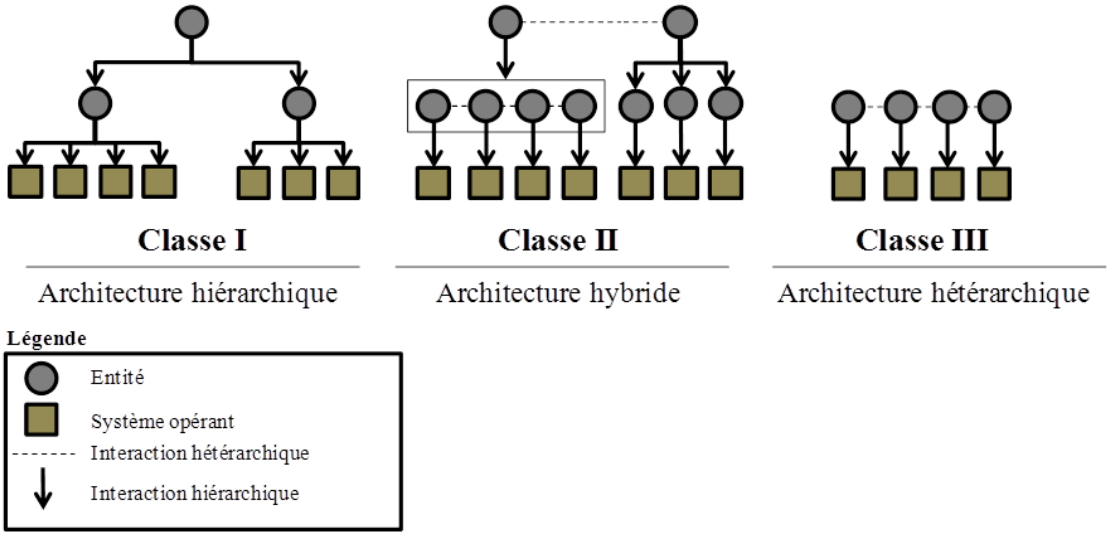


FIGURE 1.2 – Les trois classes d'architectures de pilotage selon [Trentesaux, 2009]

La **classe I** représente les architectures **hiérarchiques**, où les entités sont liées par des relations maître–esclave. Cette hiérarchie a été très étudiée [Buzacott et Yao, 1986, Scattolini, 2009] (concept de *Computer-Integrating Manufacturing*) et est très répandue et déployée en industrie depuis les années 1970. Les décisions de pilotage sont, dans ce type d'architecture, prises par l'entité de plus haut niveau, qui doit nécessairement posséder toutes les informations suffisantes à la prise de décisions permettant de bonnes performances globales. Cependant, ce type d'architecture a un grand manque de réactivité.

En effet, il y a souvent un grand laps de temps entre le moment où un évènement inattendu est détecté et le moment où la décision est prise par l'entité de plus haut niveau, par la suite propagée vers les entités de bas niveau qui l'appliquent.

La **classe III** représente les architectures **hétérarchiques**, où les entités ont le même niveau de hiérarchie et peuvent interagir pour influencer les décisions prises par les autres entités [Trentesaux, 2002]. Ainsi, chaque entité peut être esclave d'une autre à un certain moment, puis inversement à un autre moment. Afin d'éviter les possibles conflits entre les entités lors de décisions contradictoires, des notions de coopération ou de négociation ont été définies [Camalot, 2000, Taghezout et Zaraté, 2008]. Les entités étant directement liées à leur système, elles peuvent directement agir sur celui–ci, ce qui apporte une haute réactivité. Les décisions étant prises localement, un "phénomène" lié au manque de performances globales du système manufacturier se produit. Ce phénomène a été appelé *myopie* et se caractérise par un manque de visibilité sur les états futurs des entités [Trentesaux, 2009]. Le terme myopie est issu du domaine médical, mais a d'autres définitions dans des domaines différents (pour plus de détails sur ces domaines, voir [Zambrano Rey, 2014]). Dans le domaine des systèmes de production, la myopie est définie de la façon suivante :

> *La **myopie** représente le manque d'informations d'une entité sur son futur et sur celui des autres entités du système.*

En partant de cette définition, [Zambrano Rey, 2014] a défini deux types de myopie : sociale et temporelle. La myopie sociale, pour une entité, représente son manque d'information par rapport aux autres entités et est souvent rencontrée en robotique mobile [Mataric, 1992], où la communication est limitée [Defoort, 2007]. La myopie temporelle est liée à la limitation des connaissances du futur des entités, qui n'ont pas la possibilité de prévoir et d'agir sur les événements à long terme. La myopie est très présente dans les architectures hétérarchiques, où les entités prennent les décisions localement, à l'aide d'un nombre réduit d'informations. A l'inverse, si les entités ont beaucoup d'informations à traiter, la myopie est réduite, ce qui permet d'améliorer les performances globales, mais cela empêche les entités de réagir rapidement.

Les architectures de **classe II**, dites hybrides (ou semi–hétérarchiques), peuvent permettre de garder les avantages des deux autres classes (performances et réactivité), tout en réduisant la myopie. En effet, ces architectures sont une combinaison de hiérarchie et d'hétérarchie et dépendent de deux critères. Le premier critère est le dynamisme du pilotage, représentant l'évolution des interactions entre entités. Par exemple, une architecture dite dynamique a la possibilité de basculer entre hiérarchie et hétérarchie [Pach, 2013, Jimenez et al., 2016], permettant ainsi de coupler les contrôles proactif/réactif [Cardin et al., 2015]. Le second critère est l'homogénéité, c'est-à-dire la façon dont le pilotage s'applique sur les entités. Selon les critères, quatre sous–classes des architectures de classe II ont été identifiées et sont données en Annexe A avec les principales références.

Dans le cadre de cette thèse, nous nous intéresserons aux architectures de pilotage hybrides et hétérarchiques des systèmes de production flexibles. Nous allons donc nous intéresser à ces notions de flexibilité, tout en donnant différents types d'ateliers de production.

Flexibilité des systèmes de production

Afin de suivre la deuxième prescription d'IMS2020, les systèmes de production doivent intégrer une haute flexibilité. Du fait de l'absence d'accord sur sa définition, il y a eu beaucoup de discussions sur cette flexibilité. Dans [Petkova et van Wezel, 2006], pas moins de 141 définitions ont été référencées, où 49 types de flexibilité sont donnés pour les systèmes de production. Les définitions les plus récentes se basent sur la notion d'adaptation aux incertitudes, où la flexibilité est plutôt définie [ElMaraghy, 2005] comme suit :

> *La **flexibilité** est la capacité d'un système de production de pouvoir changer et prendre en charge des positions ou états différents, pour répondre aux changements des besoins, avec peu de pénalités en termes de temps, d'effort, de coût ou de performances.*

De cette définition, on remarque que la flexibilité est le moyen de faire face aux changements des activités de production d'une manière efficace et performante. Les principaux types de flexibilité sont les suivants :

- **Flexibilité de ressource** : Multitude de types d'opérations qu'une ressource (ou machine) peut effectuer à faible temps de transition.
- **Flexibilité d'opération** : Permet à un produit d'être effectué de plusieurs façons différentes, en agissant sur la séquence des opérations.
- **Flexibilité du processus** : Ensemble de produits pouvant être effectués sans modification majeure sur le système.
- **Flexibilité de séquence machine** : Capacité d'effectuer une opération d'un produit à différentes machines.
- **Flexibilité de manutention** : Moyens mis en oeuvre afin d'acheminer n'importe quel produit (ou ensemble de produits) en fonction du volume ou du poids du produit, par exemple.
- **Flexibilité du routage** : Ensemble de chemins [1] faisables pouvant être utilisés pour effectuer un produit.

Ces types de flexibilité peuvent être différents selon le type d'atelier de fabrication. En effet, les structures physiques des ateliers peuvent changer certaines flexibilités. Par exemple, le routage est différent si le système de transport est guidé ou non.

1. Le terme "chemin" n'a de sens que si le transport des produits est guidé (par exemple : système de convoi). Sinon on parlera de "trajectoire" (voir détails dans la Section 1.1.2).

Les systèmes de production sont donc classifiés par rapport à l'aménagement des ateliers :

- **Atelier à machine unique** (*Single machine shop*) : C'est l'atelier le plus simple, où une seule machine fait toutes les opérations des produits. La machine opère de façon continuelle, jusqu'à ce qu'une nouvelle configuration soit requise [Coffman Jr et al., 1990].

- **Atelier à machines parallèles** (*Parallel machine shop*) : Cet atelier est similaire au précédent, car toutes les machines sont identiques et opèrent sur toutes les opérations des produits. De ce fait, celui-ci est un atelier à machine unique avec redondance de machines, permettant notamment d'accélérer la production, mais aussi d'empêcher un arrêt de production, lorsqu'une machine tombe en panne [Sankar et al., 2005].

- **Atelier à cheminement unique** (*Flow shop*) : Les différentes ressources de cet atelier sont organisées par rapport aux séquences des produits. Les produits doivent visiter toutes les machines, où chaque opération se fait à une unique machine, dans un ordre spécifique. Le flux de produit est ici unidirectionnel. Cet atelier est flexible lorsque les machines sont placées en parallèle sur au moins une étape du cheminement [Linn et Zhang, 1999].

- **Atelier à cheminements multiples** (*Job shop*) : Dans ce type d'atelier, les ressources ayant les mêmes capacités sont regroupées. La flexibilité du processus et des ressources est élevée car les machines peuvent accomplir plusieurs opérations différentes et peuvent supporter différents types de produits. Chaque produit se rend aux ressources selon sa séquence d'opérations, puisqu'il y a une allocation d'opérations aux ressources à priori [Jones et al., 1999].

- **Atelier à production "lineless"** : Dans ce type de production, les produits et ressources sont capables de naviguer librement dans l'environnement manufacturier. Ainsi, la flexibilité de routage est très élevée, mais cela nécessite de prendre en compte d'autres contraintes liées aux déplacements des entités [Ueda et al., 2001]. Ce type d'atelier est une généralisation des ateliers "ouverts" (*Open shop*), où la flexibilité des opérations est améliorée [Sha et Hsu, 2008].

- **Atelier à projets** : Cet atelier est un premier cas particulier du précédent. Aussi connu comme atelier à position fixe, il a la particularité d'avoir un produit immobile (de grande taille par exemple) et les différentes ressources sont mobiles, afin d'opérer sur le produit [Bertrand et Wortmann, 1992].

- **Atelier flexible basé sur AGVs** : Ici, les produits sont transportés par des AGVs pour augmenter la flexibilité. Ils peuvent naviguer en suivant des chemins prédéfinis, de manière bi-directionnelle, ou naviguer librement dans l'environnement [Banaszak et Krogh, 1990]. Les ressources sont immobiles comme dans les ateliers à cheminements multiples. Lorsque les AGVs naviguent librement, il y a de nombreux avantages et impacts, discutés dans la Section 1.2.

Dans le cadre de cette thèse, nous nous intéresserons principalement aux ateliers basés sur des AGVs où la navigation est libre, permettant une haute flexibilité. Les sujets de recherche sur les systèmes de production basés sur robots/AGVs sont une partie intégrante des systèmes manufacturiers flexibles (c.f. **RT3.09**, Section 6 de [Rolstadas, 2010]). Par conséquent, il y a de nombreux intérêts à considérer ces AGVs pour répondre aux prescriptions de IMS2020. Cependant, l'utilisation d'AGVs nécessite de considérer les outils et concepts nécessaire à leur navigation.

1.1.2 Navigation des systèmes multi–agents

La navigation d'agents dans un environnement commun est souvent traitée dans le domaine de la robotique mobile. On y recherche des algorithmes de planification et/ou de stratégies de commande pour un ensemble de véhicules. L'automatisation de ces véhicules, au vu de leurs évolutions technologiques, permet maintenant de les considérer dans un plus grand nombre d'applications. Cependant, la coordination de l'ensemble des agents implique de nouvelles problématiques, qui sont différentes selon l'application. Notons que le terme robot sera utilisé au lieu d'agent ou d'AGV, afin d'améliorer la compréhension des différents concepts introduits. Nous nous intéresserons principalement à la planification de trajectoire et aux commandes coopératives des systèmes multi–agents. Le problème de suivi de trajectoire d'un seul robot, décrit dans l'Annexe B, ne sera pas traité.

Planification de trajectoire

La planification de trajectoire a été beaucoup étudiée depuis que les travaux proposés dans [Lozano-Perez, 1983] ont introduit le concept d'espace des configurations. Cette configuration désigne l'ensemble des paramètres qui caractérisent le robot dans son environnement, d'une manière unique. L'ensemble des configurations du robot est noté $\mathcal{Q}$. En utilisant cette notion de configuration, une trajectoire peut donc être définie de la façon suivante [Defoort, 2007] :

> *Une* ***trajectoire*** *est une fonction continue de* $[t_{initial}, t_{final}] \subset \mathbb{R}$ *dans* $\mathcal{Q}$ *qui, à toute valeur* $t \in [t_{initial}, t_{final}]$*, associe une configuration* $q(t)$ *:*

$$\begin{array}{rccc} q: & [t_{initial}, t_{final}] & \rightarrow & \mathcal{Q} \\ & t & \mapsto & q(t) \end{array}$$

Une trajectoire est dite ***admissible*** si elle respecte, pour des conditions initiale et finale données, la dynamique du robot considéré, ainsi que ses contraintes physiques. Les contraintes physiques, que la trajectoire doit respecter, se définissent en termes d'entrée (vitesse, accélération bornées) mais aussi en termes de sortie. En effet, selon l'environnement, le contexte et l'objectif de l'agent, certaines configurations doivent être évitées, par exemple, pour éviter les collisions avec des obstacles stationnaires ou mobiles (autres robots). De ces propos, on peut définir la planification de trajectoire pour un robot comme suit :

*La **planification de trajectoire** est le calcul d'une trajectoire admissible et sans collision pour un robot entre une configuration de départ et une configuration d'arrivée données.*

Dans le cadre de la planification de trajectoire des systèmes multi–agents, d'autres problématiques sont à traiter, telles que la coordination ou l'évitement de collisions inter-robots. Afin que les actions des robots soient cohérentes, il est nécessaire de mettre en avant les principaux mécanismes de coordination. Trois mécanismes de coordination ont été identifiés [Mintzberg, 1989] et sont retranscrits ci-dessous selon notre contexte :

- **Leadership** : La coordination se fait de manière hiérarchique : certains robots ont des décisions qui influencent les autres robots.
- **Ajustement mutuel** : Les robots s'accordent pour partager des ressources pour atteindre un objectif commun. Pour ce mécanisme, aucun robot n'a de contrôle sur les autres.
- **Standardisation** : Des procédures sont préalablement définies pour traiter des situations particulières, en fixant des règles qui limitent les conflits, par exemple.

Tout comme les architectures de pilotage discutées précédemment, les mécanismes de coordination sont centralisés ou décentralisés. Lorsqu'ils sont centralisés, l'ensemble des trajectoires des robots sont calculées par une unité centrale à l'aide de problèmes d'optimisation à grande échelle. Cependant, la complexité en termes de calcul et le manque de réactivité empêchent la centralisation pour la plupart des applications avec des flottilles de plus de cinq robots.

Dans les approches proposées, nous nous intéresserons plutôt aux mécanismes décentralisés, bien plus rentables en termes de calcul et de réactivité, face aux événements inattendus. La décentralisation nécessite un flux de communication élevé, pour transmettre des informations aux autres robots. Chaque robot génère sa propre trajectoire en prenant en compte les autres robots (leurs intentions, activités, configuration, ...). En conséquence, les robots doivent être capables de mettre à jour leur trajectoire en ligne régulièrement, afin d'éviter les collisions inter-robots et ce, jusqu'à l'achèvement de leur(s) mission(s).

Méthodes de résolution du problème de planification

La plupart des problèmes de planification de trajectoire sont complexes et ont été définis comme NP–difficiles [Canny et Reif, 1987]. Afin de prouver cette complexité, de nombreux problèmes de planification ont été étudiés dans [Canny, 1988], comme le problème du plus court chemin, le problème de planification à vitesse bornée ou le problème d'évitement de collisions. On remarque que même les problèmes les plus simples de planification sont considérés comme complexes à résoudre.

Afin de résoudre ces problèmes, un ensemble de méthodes classiques ont été proposées [Masehian et Sedighizadeh, 2007]. Ces méthodes de résolution sont basées sur des approches basiques, telles que les feuilles de route (*roadmaps*) [Canny, 1988], les décompositions en cellules [Latombe, 2012], les champs

de potentiel [Gazi et Passino, 2004] ou la programmation séquentielle quadratique [Defoort et al., 2007]. Beaucoup de problèmes de planification peuvent être résolus en se basant sur ces approches basiques ou en combinant celles-ci. Par exemple, les méthodes de graphe de visibilité [Asano et al., 1985] ou de partition de Voronoi [Choset, 1996] sont basées sur les feuilles de route. Dans [Defoort et al., 2011], une technique d'optimisation pour planifier les trajectoires des robots est combinée avec des champs de potentiels pour assurer l'évitement de collisions. Certaines approches consistent à décomposer le problème de planification en deux étapes, où la première permet d'éviter les obstacles stationnaires, et la seconde permet d'éviter les collisions inter-robots [Guo et Parker, 2002].

Les approches classiques mentionnées ont cependant plusieurs inconvénients, comme des temps de calcul élevés ou des problèmes de minimums locaux. Afin de rendre la planification de trajectoire plus efficace, tout en éliminant ces inconvénients, d'autres méthodes, outils ou algorithmes ont été développés. Les algorithmes probabilistes (par exemple, *probabilistic roadmaps* [Kavraki et al., 1996] ou *rapidly–exploring Random trees* [LaValle et Kuffner, 2001]) permettent une implémentation très rapide [Sucan et al., 2012]. Les méthodes de planification auxquelles nous nous intéressons sont basées sur la programmation mathématique (ou technique d'optimisation), car elles semblent plus appropriées au contexte des systèmes de production, où l'on cherche à optimiser des performances.

Afin de pouvoir calculer et mettre à jour en ligne les trajectoires, plusieurs travaux sur les techniques d'optimisation ont été proposés. Par exemple, dans la planification à horizon glissant [Defoort et al., 2009a], les trajectoires planifiées sont partielles, afin de réduire les temps de calcul. Pour améliorer le temps de calcul, mais aussi éviter les minimums locaux, des outils méta heuristiques ont été adaptés à la planification de trajectoire du fait de leur convergence rapide vers des solutions proches de l'optimale. Par exemple, les algorithmes génétiques [Ahuactzin et al., 1993], les colonies de fourmis [Brand et al., 2010] ou d'abeilles [Lin et al., 2009], la recherche tabou [Masehian et Amin-Naseri, 2008] ou l'optimisation par essaim particulaire (PSO) [Zhengxiong et Xinsheng, 2010] ont été appliqués à des problèmes de planification de trajectoire. Les concepts de ces méta heuristiques, ainsi que leur application sur le problème de planification de trajectoire, sont détaillés dans l'annexe C.

Commande coopérative

La navigation des robots ne se limite pas à la planification de trajectoire. En effet, les stratégies de commande distribuée et coopérative des robots peuvent aussi permettre leur navigation. Dans cette partie, nous présenterons une grande partie des ces stratégies distribuées, en introduisant les notions de rendez–vous et de contrôle de formation.

L'étude de commandes distribuées d'un ensemble de robots est issue des travaux en informatique distribuée [Lynch, 1996] et en science de gestion [DeGroot, 1974]. Dans la communauté de contrôle des systèmes, les travaux pionniers sont [Tsitsiklis et al., 1984], où un problème de prise de décision distribuée a été étudié. Depuis ces travaux, beaucoup d'algorithmes ont été proposés sur la commande distribuée [Cao et al., 2013]. L'objectif de ces stratégies de commande est de faire coopérer l'ensemble

des robots à travers des protocoles distribués. La coopération se fait dans ce cas par des relations inter–robots, où l'échange d'informations joue un rôle essentiel. Le contrôle distribué a plusieurs avantages, comme le faible coût opérationnel, une haute réactivité, une bonne robustesse, ...

Afin d'élaborer ces stratégies, il est nécessaire d'introduire la théorie des graphes. En effet, la commande dépend du réseau de communication de la flottille de robots. On parle alors de topologie du graphe, qui dépend de l'ensemble des robots et des liens de communication entre ceux–ci. Le graphe est dit dirigé lorsque les communications sont unidirectionnelles. Sinon le graphe est non–dirigé. De plus, la topologie peut varier au cours du temps, on dira alors que la topologie commute, sinon, elle sera dite fixe.

Parmi les différentes commandes coopératives, on distingue trois grandes directions, qui ont des objectifs différents, mais sont basées sur le même concept de la théorie des graphes. Ces trois directions sont les suivantes :

- **Le rendez-vous** : Le problème du rendez–vous, souvent appelé *consensus*, consiste à faire converger tout ou partie de l'état des robots vers une certaine configuration (ou état). Comme cette configuration est non définie, les agents se déplacent vers la moyenne pondérée de la configuration de leurs voisins.

- **Le rendez–vous vers un meneur** : Ce problème, appelé *leader–follower consensus*, est similaire au précédent. Cependant, au lieu de faire converger l'ensemble des robots vers une configuration unique, les robots doivent suivre un meneur. Ce meneur peut être physique ou virtuel. Lorsqu'il est virtuel, le problème est similaire au suivi de trajectoire, où l'ensemble des robots doivent converger vers cette trajectoire, puis la suivre. De plus, le meneur peut ne pas se déplacer, la stratégie revient alors à faire converger la flottille vers une position fixe.

- **Le contrôle de formation** : Ici, l'idée est de faire converger les robots vers une certaine formation ou faire suivre une trajectoire (meneur virtuel) en formation. Ce problème est appelé *formation control* ou *formation tracking* lorsqu'il s'agit de faire suivre une trajectoire. Beaucoup de sous–problèmes ont émergé par la suite, comme la fusion de groupes d'agents, la division d'un groupe en plusieurs autres ou encore l'étude de la rigidité de la formation.

Un ensemble conséquent de travaux ont été proposés, avec des modèles dynamiques ou des topologies de graphe différentes, pour chacune de ces directions. Le tableau 1.1 propose des exemples de travaux sur chacune des directions. La colonne 1 montre sur quel modèle dynamique de robot la stratégie de commande a été appliquée. La colonne 2 donne le type de convergence souhaité et la colonne 3 donne les références associées. Notons qu'il existe beaucoup de références et que seules les principales ont été données. De plus, d'autres travaux ont été proposés avec des modèles à temps discrets [Cao et Ren, 2010b], en présence de retards liés à la communication [Olfati-Saber et Murray, 2004], ou sur les ruptures momentanées de communication [Taousser et al., 2016].

Parmi tous les travaux sur la commande coopérative distribuée, on remarque que la convergence à temps fixe reste peu étudiée. En effet, les travaux sur la commande à temps fixe sont plus récents. Ce type de commande, introduit par [Polyakov, 2012a], a pour but de faire converger l'état en un temps fixe, c'est–à–dire en un temps fini indépendant des conditions initiales. La notion de temps fixe est rappelée dans la Section 5.4 du Chapitre 5. Dans ce même chapitre, nous proposerons une stratégie de commande meneur–suiveur à temps fixe, pour des robots non–holonomes.

Tableau 1.1 – Les principaux travaux proposés sur la commande coopérative

	Modèle dynamique	**Convergence souhaitée**	**Référence**
Rendez–vous	Simple intégrateur	Asymptotique	[Olfati-Saber et Murray, 2004] [Ren et al., 2005]
	Double intégrateur	Asymptotique	[Cao et Ren, 2010a] [Qin et al., 2011]
	Simple intégrateur	Temps fini	[Cortés, 2006] [Jiang et Wang, 2009]
	Double intégrateur	Temps fini	[Li et al., 2011] [Wang et Hong, 2008]
	Non holonome	Temps fini	[Wang et al., 2012] [Zoghlami, 2014]
	Simple intégrateur	Temps fixe	[Parsegov et al., 2013]
	Double intégrateur	Temps fixe	[Zuo et al., 2014]
Meneur–suiveur	Simple intégrateur	Asymptotique	[Yong-Zheng et Jiong, 2008] [Ren, 2008]
	Double intégrateur	Asymptotique	[Yang et al., 2012] [Song et al., 2010]
	Simple intégrateur	Temps fini	[Yong et al., 2012] [Cao et Ren, 2012]
	Double intégrateur	Temps fini	[Khoo et al., 2009] [Zhang et Yang, 2013]
	Non holonome	Temps fini	[Shi et al., 2015] [Shi et al., 2016]
	Simple intégrateur	Temps fixe	[Defoort et al., 2015]
	Double intégrateur	Temps fixe	[Zuo et al., 2014]
Contrôle de formation	Simple intégrateur	Asymptotique	[Olfati-Saber, 2006] [De Gennaro et Jadbabaie, 2006]
	Double intégrateur	Asymptotique	[Olfati-Saber et Murray, 2002] [Lu et al., 2012]
	Non holonome	Asymptotique	[Lin et al., 2005] [Do et Pan, 2007]
	Non holonome	Temps fini	[Defoort et al., 2008] [Shi-Cai et al., 2007]
	Simple intégrateur	Temps fini	[Xiao et al., 2009] [Cao et al., 2010]

Les principaux outils de navigation ont été introduits. Nous avons vu une multitude de travaux sur la planification de trajectoire et sur la commande coopérative. Cependant, afin d'appliquer ces outils pour faire naviguer des AGVs dans un système de production, d'autres aspects doivent être introduits, afin d'atteindre un certain niveau de mutualisation.

1.2 Mutualisation entre robotique et système manufacturier

1.2.1 La mutualisation

L'utilisation d'AGVs dans un système de production peut se faire de plusieurs manières différentes. Dans beaucoup de travaux, la navigation des AGVs se fait par chemins virtuels [Bocewicz et al., 2014,Erol et al., 2012,Srivastava et al., 2008] à l'aide de guidage optique ou magnétique, par exemple [Ganesharajah et al., 1998]. Ce type de guidage peut réduire l'interblocage en ajoutant des outils permettant d'éviter les collisions [Digani et al., 2014]. Dans ce cas, uniquement les outils de suivi de trajectoire avec évitement de collisions sont nécessaires et la mutualisation est faible, car les outils requis ont peu d'impact sur la production des produits.

Dans notre étude, nous considérons une navigation des AGVs leur permettant de se mouvoir librement, c'est-à-dire sans qu'ils soient contraints à suivre des chemins. Ceci implique des outils de navigation plus complexes, qui doivent s'harmoniser et/ou se combiner avec les fonctions de pilotage utiles à la production. C'est dans cette notion "d'harmoniser et/ou se combiner" que la mutualisation est ici définie [Demesure et al., 2014].

Faire naviguer librement les AGVs a plusieurs avantages, qui seront, pour les plus importants, illustrés à l'aide de la Fig. 1.3.

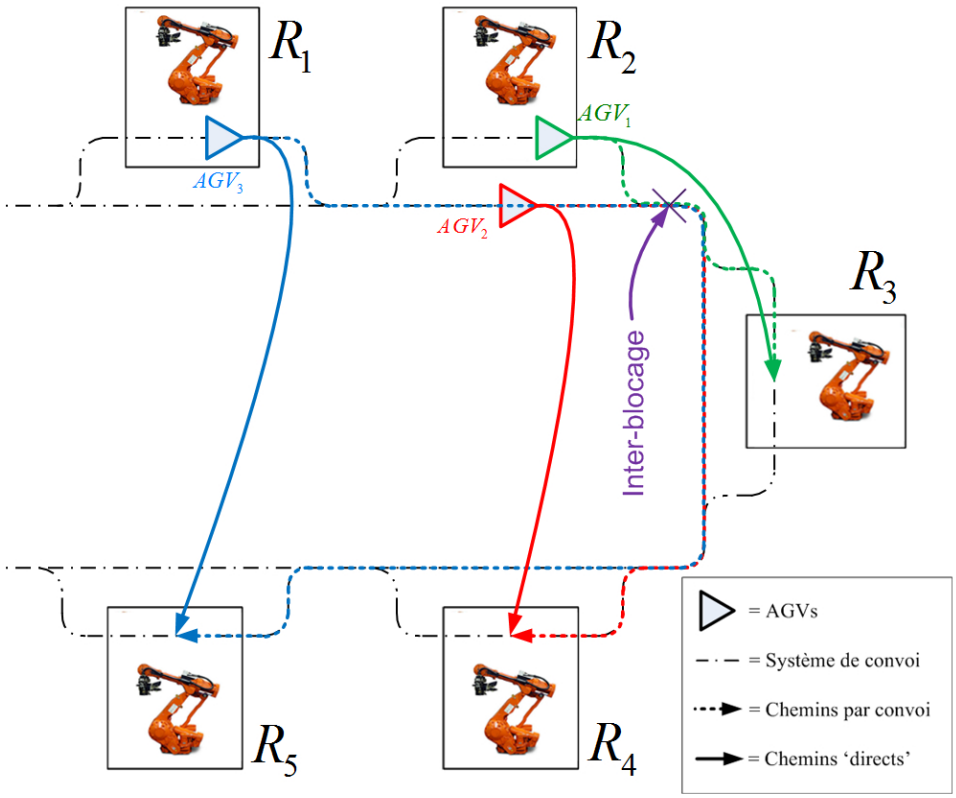


FIGURE 1.3 – Mise en évidence des avantages de la navigation libre.

D'abord, la flexibilité dans le routage des produits transportés par les AGVs est augmentée, permettant de supprimer les problèmes d'interblocage dans le cheminement des produits et de réduire la saturation du système. L'interblocage [Girault et al., 2013] dans un système de convoi se produit lorsque deux AGVs, ou plus, convoitent le même chemin, comme le montre la Fig. 1.3, avec les AGVs 1 et 2. La saturation correspond au nombre d'AGVs maximum qu'un système de production peut tolérer. Elle est réduite, car sans chemin, les AGVs sont libres de se déplacer dans l'atelier.

En plus de la flexibilité sur le routage, la flexibilité de manutention peut aussi être améliorée. Il est possible de transporter des produits de plus gros volume (ou des lots de produits sur une palette) en utilisant plusieurs AGVs. Ces AGVs devront maintenir une certaine formation lors du transport.

L'avantage majeur de la navigation non guidée est de réduire les temps de transport entre les ressources. En effet, les AGVs peuvent planifier des trajectoires directes, permettant d'atteindre les ressources désirées plus rapidement. La Fig. 1.3 illustre très bien cet avantage, car l'AGV 3 peut directement atteindre sa ressource de destination, sans entrer en conflit avec les AGVs 1 et 2. Les temps de transport étant réduits, il est possible d'achever les opérations des produits plus rapidement, ce qui réduit le temps de séjour des produits dans le système. Cet avantage permet aussi de considérer la production urgente d'un produit. En donnant une très haute priorité à un produit urgent, il pourra achever l'ensemble de ses opérations de manière directe, alors que les autres produits devront s'adapter.

1.2.2 Les impacts de la mutualisation et les problématiques résultantes

L'intégration d'AGVs dans un système de production a de nombreux impacts, qui peuvent être bénéfiques, ou non, au système manufacturier. Il est nécessaire de s'interroger sur ces principaux impacts, ainsi que les problématiques qui en résultent. De quel manière faut-il piloter le système de production en incluant la navigation des AGVs ? Quelle architecture de pilotage permet d'atteindre un niveau de performance élevé ? A quel niveau de myopie les AGVs sont-ils soumis ? Quels moyens pourraient permettre de traiter les conflits entre AGVs, en maintenant un niveau de performance suffisant ? Quels outils de navigation sont les plus efficaces pour traiter un problème de production avec contraintes temporelles ?

Afin de répondre à ces problématiques, nous proposons d'introduire certains points sur lesquels les impacts sont importants, car ils peuvent devenir des inconvénients, selon les solutions proposées pour les prendre en compte. De ce fait, il est nécessaire de les présenter, afin de mieux comprendre les enjeux de la mutualisation, ce qui permet aussi de montrer la manière dont nos recherches ont été orientées.

Le pilotage, défini préalablement dans la section 1.1.1 de ce chapitre, est principalement centré sur l'ordonnancement et la conduite. L'idée est de savoir sur quelle ressource une opération doit être effectuée. D'une manière abstraite, on pourrait dire que la fonction de navigation est semblable à une fonction de conduite. En effet, la conduite [Aissani, 2010] a été définie comme la mise en oeuvre de l'ordonnancement, qui dans notre cas, revient à planifier une trajectoire permettant d'atteindre la ressource allouée par la fonction d'ordonnancement. De plus, tout comme la conduite, la navigation

permet aussi d'assurer une flexibilité quotidienne dans le routage pour faire face aux fluctuations du système. Ainsi, la navigation peut être considérée comme une fonction de pilotage car, comme la conduite, elle est étroitement liée à la fonction d'ordonnancement.

La navigation est différente lorsque les AGVs se déplacent dans un système de production. D'abord, il est nécessaire de mentionner les contraintes temporelles pour le transport des produits. Les produits transportés par les AGVs peuvent avoir des échéances qu'il faut prendre en compte dans la navigation des AGVs. De plus, même sans la présence de contraintes temporelles, il faut faire en sorte que le produit soit achevé le plus tôt possible. Cependant, beaucoup de conflits peuvent se produire durant la navigation, comme les collisions à éviter. Il faut donc s'interroger sur la manière de résoudre ces conflits afin, d'une part, de ne pas trop réduire les performances globales, et d'autre part, de respecter les contraintes temporelles définies sur chaque produit transporté par leur AGV.

Plusieurs outils de coopération ou de négociation ont été proposés pour résoudre ces conflits de manière directe ou non. Lorsque la résolution de conflit est faite indirectement, les entités coopèrent, sans pour autant se consulter les unes les autres, comme par exemple avec les champs de potentiel [Zbib et al., 2012], le *blackboard* [Hayes-Roth, 1985], le protocole Contrat–Net [Sousa et Ramos, 1999], [Kadera et Tichy, 2009] ou la stigmergie [Sallez et al., 2009]. S'ils doivent se consulter, les entités doivent résoudre le conflit entre–elles à l'aide de négociation par réseaux de Pétri [Chen et al., 1999], par approche basée sur la théorie des jeux [Yu et al., 2011], par approche sociale [Zambrano Rey, 2014], ou autres [Kraus, 2001].

Les performances locales et globales ne dépendent pas uniquement de la navigation et de la manière dont les conflits sont résolus. L'architecture de pilotage a, dans ce type de problème, une grande influence. En effet, la navigation est une fonction répartie de manière hétérarchique, car chaque AGV décide lui–même de sa trajectoire. De plus, il faut prendre en compte l'inconvénient majeur de ce type d'architecture : la myopie. La fonction de navigation est très sensible à cette myopie, car elle est à la fois sociale et temporelle. Sociale, car, à chaque moment, un AGV ne connaît que les informations des AGVs à proximité. Temporelle, car il est difficile de prévoir les évènements qui vont se produire lors de la navigation avec exactitude. Cette myopie rend l'ordonnancement plus complexe, car il devient difficile de savoir si la ressource allouée est la meilleure à chaque moment. Par conséquent, les temps de transport entre les ressources peuvent être fortement variables, ce qui rend l'optimisation globale beaucoup plus complexe.

Comme il devient difficile d'ordonnancer globalement le système, on s'interroge sur la possibilité de rendre cet ordonnancement plus dynamique, permettant de choisir une ressource lors de l'opération en cours (*work–in–progress*). Ce dynamisme pourrait permettre de changer de destination lors de la navigation pour diverses raisons : afin d'éviter des conflits compliqués ou insolubles, car la ressource choisie tombe en panne, ou parce que l'AGV n'a plus assez d'énergie pour l'atteindre. Ce dynamisme aura pour effet d'améliorer la réactivité du système où chaque AGV peut s'adapter à toute éventualité durant le trajet.

Dans ce sens, les outils de planification de trajectoire sont bénéfiques car ils peuvent permettre d'avoir une indication sur le temps d'arrivée à la ressource. Ils doivent aussi permettre d'éviter les obstacles et surtout les collisions inter–robots. En effet, la manière dont les collisions sont évitées a de l'influence sur la production, car les AGVs déviant de leur trajectoire retardent l'arrivée à leur destination (ressource).

Les techniques d'optimisation semblent plus pratiques pour traiter ce type de problème. En effet, on peut minimiser un ou plusieurs critères, tels que les performances de production désirées, tout en considérant les contraintes physiques, temporelles de l'AGV. La fonction de planification de trajectoire doit aussi permettre de mettre à jour la trajectoire suite à la résolution des conflits rencontrés.

La planification de trajectoire n'est pas nécessairement le seul outil impacté par la mutualisation des domaines. En effet, les commandes coopératives entre robots doivent aussi respecter le cahier des charges fourni par le haut–niveau. Ce qui implique une convergence plus rapide par rapport à d'autres applications. En effet, la convergence en temps fini ou en temps fixe permettrait de respecter un tel cahier des charges. Cependant, au vu des limitations physiques des AGVs, il serait nécessaire que le cahier des charges soit bien élaboré afin que le rendez–vous à une ressource dans un délai imposé soit réalisable.

Pour finir, notons qu'il reste encore plusieurs problématiques non énumérées dans ce chapitre telles que la notion d'énergie ou les capacités limitées des AGVs. Ces problématiques seront discutées dans la section 6.3.2.

1.2.3 Conclusion sur la mutualisation

Nous avons vu dans cette section les avantages de la mutualisation lorsque le transport des produits se fait sans système de convoi. De plus, plusieurs impacts venant de cette mutualisation ont été présentés. En regardant ces avantages et ces impacts, on pourrait s'interroger sur la faisabilité de l'utilisation d'outils de navigation par rapport au problème d'embarquabilité. Cependant, beaucoup d'industries utilisent ces AGVs dans leurs entrepôts. Par exemple, nous avions mentionné que l'entreprise Amazon a déployé des AGVs dans ses entrepôts [Gilmour, 2003] où les AGVs ont plusieurs fonctions, dont la navigation, qui est traitée par une implémentation de l'algorithme A^*. Il existe beaucoup de fournisseurs d'AGVs pour les entreprises qui utilisent des algorithmes de navigation, comme Savant Automation Inc.[1], JBT Corporation[2], Transbotics Corporation[3], ou American in Motion[4]. De plus, beaucoup de chercheurs contribuent à l'utilisation d'AGVs dans des milieux industriels [Herrero-Perez et Martinez-Barbera, 2008, Herrero-Perez et Matinez-Barbera, 2008, Martínez-Barberá et Herrero-Pérez, 2010], à l'aide par exemple de l'algorithme $D*$. Enfin, d'autres contribuent à leur façon à l'application de la navigation en milieu manufacturier [Watanabe et al., 2001], mais sans application industrielle.

Ces travaux ne montrent pas uniquement la faisabilité, mais aussi que les AGVs sont de nos jours

1. http://www.agvsystems.com/, 2. http://www.jbtc-agv.com/, 3. https://www.transbotics.com/,
4. http://www.weareaim.com/

plutôt déployés dans des entrepôts. On remarque notamment le manque de travaux sur le problème de production. En effet, l'utilisation d'AGVs en entrepôt est moins contrainte que dans le cas du problème de production, où l'entreprise doit répondre à des besoins.

1.3 Conclusion

Dans ce chapitre, un ensemble de concepts, d'outils et de méthodes ont été introduits sur les systèmes de production et sur la navigation. Après avoir présenté la mutualisation de ces domaines, nous avons montré les difficultés rencontrées lorsque des AGVs naviguent dans un système manufacturier.

Le domaine des systèmes manufacturiers est très riche, en termes d'outils et de méthodes permettant d'améliorer les performances globales et la réactivité. Après avoir introduit le pilotage, nous avons vu que les inconvénients des architectures hiérarchiques sont le manque de réactivité. A l'inverse, les architectures hétérarchiques souffrent du phénomène de myopie, réduisant les performances globales du système. Dans ce but, les architectures hybrides peuvent permettre de combler ces inconvénients.

En énumérant les différents types d'atelier, nous en avons conclu que faire naviguer les produits librement dans l'atelier à l'aide d'AGVs permet d'obtenir une haute flexibilité, mais cela nécessite des outils de navigation. Ces outils, tels que la planification de trajectoire et la commande coopérative, devront permettre la navigation des AGVs entre les différentes ressources. De plus, il est nécessaire de planifier les trajectoires de manière décentralisée et de pouvoir les mettre à jour en ligne, pour éviter les collisions ou autres conflits de manière réactive.

L'utilisation d'AGVs naviguant librement dans un système de production a de nombreux avantages, mais nécessite une mutualisation entre les domaines de la robotique mobile et des systèmes de production. Cette mutualisation a un ensemble conséquent de problématiques liées au pilotage, à la navigation mais aussi aux capacités des AGVs. La littérature nous a montré que l'utilisation d'AGVs en industrie est faisable, mais que beaucoup de contributions sont nécessaires dans le domaine de la production.

Afin de combler ce manque, nous introduirons des méthodologies permettant la navigation d'AGVs (ou robots/agents mobiles) afin de proposer un point de départ pour la mutualisation entre les domaines. Au vu des avantages de la planification de trajectoire, nous proposerons une combinaison de cette fonction avec celle d'ordonnancement comme contribution principale. Cette combinaison sera appliquée sur deux types d'architectures de pilotage : hétérarchique et hybride avec superviseur. Nous y verrons notamment les effets de la myopie résultante, en comparant les deux approches.

En termes de commande coopérative, une stratégie de commande permettant à une flottille d'AGVs d'atteindre une ressource par rendez–vous sera proposée. De plus, cette stratégie permettra de respecter un cahier des charges, où la flottille devra atteindre la ressource spécifiée en un temps donné, et ce, quelles que soient les conditions initiales des AGVs.

Chapitre 2

Formulation du problème dans un environnement manufacturier flexible

2.1 Introduction

Dans le chapitre précédent, nous avons vu que la fonction d'ordonnancement est étroitement liée à la navigation. Cependant, calculer l'ordonnancement d'une manière statique est difficilement réalisable à cause du phénomène de myopie car on ne peut prévoir à priori la manière dont l'agent va résoudre les conflits qu'il va rencontrer lors de la navigation. Nous avons aussi vu que la navigation par planification de trajectoire est mieux adaptée, permettant d'avoir une connaissance sur le temps d'arrivée à la ressource, ce qui est utile pour le calcul de l'ordonnancement. De plus, les techniques d'optimisation semblent appropriées pour traiter un problème en relation avec les performances du produit transporté par l'agent.

Ces propos nous mènent à envisager une combinaison entre un planificateur de trajectoire avec une fonction d'ordonnancement, permettant une haute réactivité lors de la navigation. En effet, la formulation du problème de planification de trajectoire sous forme de problème d'optimisation peut facilement intégrer les contraintes liées au produit. De plus, le choix du critère d'optimisation à minimiser nous permet de mettre en relation la navigation avec l'objectif de production. Comme la fonction de planification de trajectoire est appliquée individuellement, cela nous amène à considérer des architectures hétérarchiques (ou partiellement hétérarchiques).

Au vu de l'hétérarchie, il y aura nécessairement des conflits à résoudre entre les AGVs dont l'évitement de collisions. Une approche de coordination par leadership est préférable car de cette manière, un produit urgent n'a pas à dévier de sa trajectoire optimale. Par conséquent, la résolution de conflit devra dépendre des performances des produits transportés, mais aussi de leurs contraintes temporelles. Ceci permettra une coordination en rapport avec la production, où des AGVs ayant plus de temps pour achever leur produit, utiliseront ce temps pour éviter les autres AGVs qui en ont moins.

Ce chapitre va nous permettre d'introduire les deux approches présentées dans les chapitres 3 et 4 qui traitent le même problème d'une manière différente. La formulation de ce problème, l'objectif des AGVs et les hypothèses communes aux deux approches seront énumérés ci–dessous. Notons que pour chaque approche, les courbes splines, utilisées pour mettre les problèmes d'optimisation sous forme paramétrique, seront introduites dans ce chapitre avec leurs spécificités.

2.2 Idée générale et objectif des véhicules autoguidés

Considérons un système manufacturier flexible avec N_c ressources, N_l produits et N_i AGVs (pouvant être désignés par agents par commodité) où N_c, N_l, $N_i \in \mathbb{N}^*$. Le niveau de pilotage haut est appelé niveau de Gestion des Opérations et des Ressources de Production (G.O.R.P). Les notations, adaptées d'un problème de production comme [Trentesaux et al., 2013] et données à la page 7, sont rappelées à l'aide du tableau 2.1 afin de faciliter la compréhension du problème.

Tableau 2.1 – Notations des variables et paramètres

$\mathcal{R}$:	ensemble des ressources (machines) $\mathcal{R} = \{1, ..., N_c\}$.
$\mathcal{A}$:	ensemble des AGVs (agents) $\mathcal{A} = \{1, ...N_i\}$.
$\mathcal{J}$:	ensemble des produits $\mathcal{J} = \{1, ..., N_l\}$.
$\mathcal{O}_l$:	ensemble séquencé d'opérations $\mathcal{O}_l = \{1, ..., N_n\}$ à effectuer par le produit $l \in \mathcal{J}$.
$\mathcal{R}_{nl}$:	ensemble de ressources pouvant effectuer l'opération n du produit l.
opt_{nl} :	temps de traitement de l'opération n du produit l (à une ressource $b \in \mathcal{R}_{nl}$).
odd_{nl} :	échéance de l'opération n du produit l.
ct_{nlb} :	temps pour achever l'opération n du produit l à la ressource $b \in \mathcal{R}_{nl}$.
q_b :	position $(x_b, y_b)^T$ de la ressource $b \in \mathcal{R}$ dans l'environnement manufacturier.
w_b :	temps d'attente actuel à la ressource $b \in \mathcal{R}$.
q_i :	position actuelle de l'agent i.
v_i :	vitesse actuelle de l'agent i.
$q_i(t)$:	Trajectoire de l'agent i.
$T_{i,init}$:	Instant initial lorsque l'agent i démarre (ou a démarré) de la ressource.
$T_{i,fin}$:	Instant final que l'agent i planifie pour atteindre la prochaine ressource.
T_{i,upt_i} :	Instant (actuel) où la trajectoire de l'agent i est mise à jour.
TT_{ib} :	Temps de transport vers la ressource b.
$\mathcal{N}_i$:	Ensemble des voisins de l'agent i (agents qui communiquent avec celui–ci).
$\mathcal{HP}_i$:	Ensemble des voisins de l'agent i ayant un niveau de priorité plus haut que celui–ci.
R_{com} :	Portée de communication pour tous les agents $i \in \mathcal{A}$.
d_{safe} :	Distance de sécurité à respecter pour éviter les collisions inter–agents.

Dans ce système manufacturier, les AGVs ne doivent pas uniquement transporter leur produit entre les différentes ressources. En effet, ils doivent aussi prendre dynamiquement des décisions afin d'ordonnancer leur produit selon un cahier des charges comme le montre la Fig. 2.1.

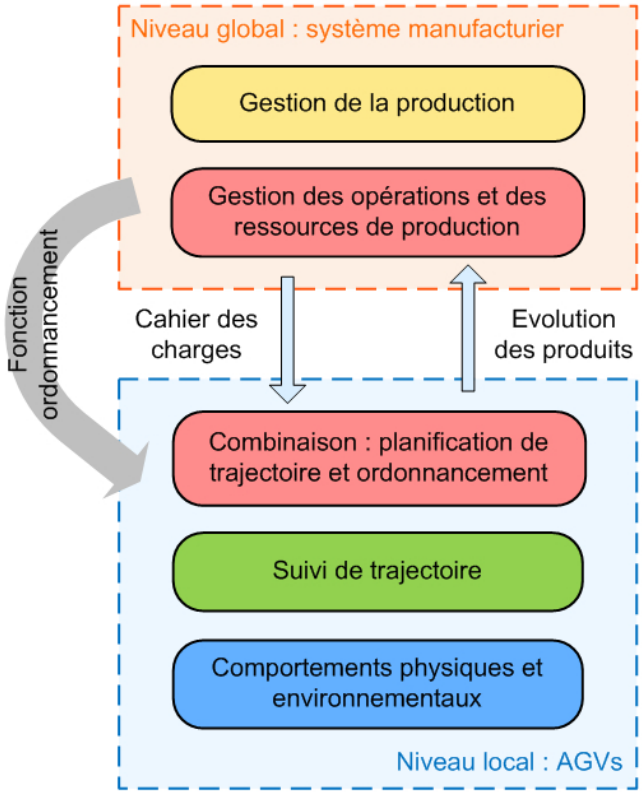


FIGURE 2.1 – Interactions entre le niveau global et les AGVs

La première idée est de descendre la fonction d'ordonnancement au niveau local des AGVs dans l'algorithme de navigation. La fonction permettant d'élaborer le cahier des charges reste au niveau global dans la gestion des opérations et des ressources de production (G.O.R.P) et ne sera pas traitée dans les approches proposées. Le cahier des charges correspond à l'assignement d'une date, appelée échéance, pour laquelle une opération d'un produit doit être achevée sans préciser explicitement sur quelle ressource l'opération doit être achevée. L'ordonnancement signifie l'affectation d'une ressource pour chaque opération en respectant le cahier des charges.

En plus de descendre la fonction d'ordonnancement au niveau local, la seconde idée est de faire une combinaison entre l'ordonnancement et la planification de trajectoire. Cette idée repose sur le fait que ces fonctions sont étroitement liées (voir chapitre 1). En effet, la planification de trajectoire permet d'avoir une information sur la date à laquelle l'AGV arrivera à la ressource en considérant l'évitement de collisions. En la combinant à la fonction d'ordonnancement, on peut sélectionner la ressource par rapport à l'information donnée par la trajectoire (temps d'arrivée à la ressource). Cependant, il convient de noter qu'il est difficile voire impossible, pour deux agents ou plus, d'ordonnancer leur produit d'une manière simultanée, c'est-à-dire en parallèle. Ce problème est illustré par un exemple à l'aide de la Fig. 2.2 où deux agents doivent ordonnancer leur produit en même temps tout en planifiant leur trajectoire.

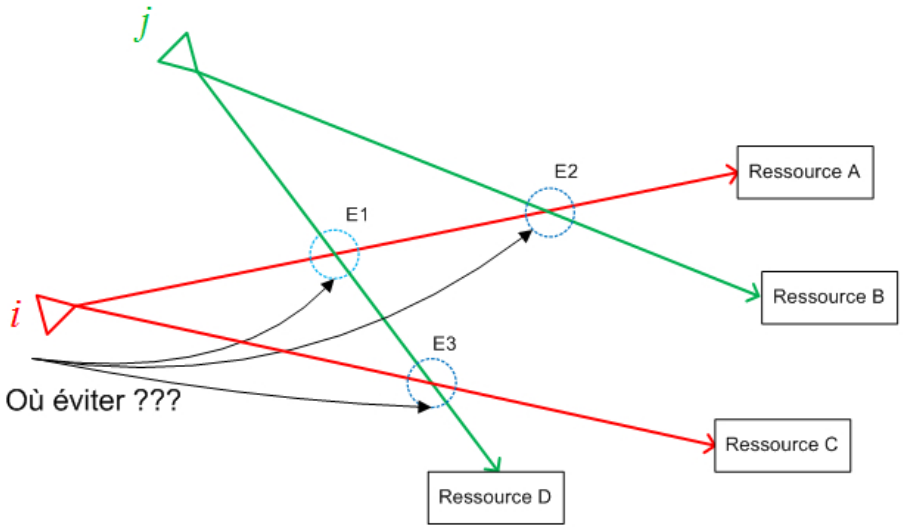


FIGURE 2.2 – Exemple du problème de l'ordonnancement combiné avec la planification

Pour cet exemple, l'agent i a un niveau de priorité plus bas et doit éviter l'agent j (ou un conflit de collision avec celui–ci). Pour cela, il faut impérativement que l'agent i ait la connaissance du choix de ressource de l'agent j. En effet, ne connaissant pas le choix de ressource de l'agent j (car la planification est simultanée), l'agent i ne peux savoir s'il doit éviter la collision en $E1$ ou en $E2$ s'il se dirige vers la ressource A (voir Fig. 2.2). De la même façon, il ne sait pas s'il doit éviter la collision en $E3$ avec l'agent j s'il se dirige vers la ressource C. De ces propos, on remarque qu'il faut mettre en oeuvre des outils permettant d'éviter ce problème.

Pour cela, les deux approches possibles sont énumérées ci–dessous :

- La première approche, appliquée dans le chapitre 3, consiste à ne pas mettre à jour la trajectoire et la ressource de plusieurs agents d'une manière simultanée. Cependant, cela peut engendrer des répercussions car un agent doit attendre que tous les autres ont mis à jour leur trajectoire pour savoir vers quelles ressources ils se dirigent.

- La seconde approche, appliquée dans le chapitre 4, permet de mettre à jour la trajectoire de plusieurs agents d'une manière simultanée. Cependant, la ressource n'est pas mise à jour à chaque fois que la trajectoire l'est car tous les agents ne sont pas autorisés à ordonnancer leur produit à chaque mise à jour de trajectoire. Ainsi, certains agents ne se dirigent que vers une seule ressource, permettant aux autres d'ordonnancer leur produit en étant assurés du choix de ressources des agents qu'ils doivent éviter.

L'objectif pour chaque AGV est, en démarrant de la ressource $a \in \mathcal{R}_{(n-1)l}$, de minimiser le temps pour compléter l'opération n en cours, ce qui revient à achever l'opération le plus rapidement possible.

Autrement dit, il faut choisir la *meilleure* ressource appropriée $b \in \mathcal{R}_{nl}$ (ordonnancement) tout en planifiant une trajectoire $q_i(t)$ vers celle–ci (conduite). La *meilleure* ressource signifie celle qui permet d'achever le produit dans le plus court délai. Ceci revient à minimiser le temps pour achever l'opération ct_{nlb} comme suit :

$$\min_{q_i(t),\, b} ct_{nlb} = T_{i,init} + TT_{ib} + w_b + opt_{nl} \tag{2.2.1}$$

où $T_{i,init}$ est l'instant de départ de la ressource $a \in \mathcal{R}_{(n-1)l}$ ayant effectuée l'opération $n-1$, TT_{ib} est le temps de transport entre la position initiale $q_i(T_{i,init}) = q_a$ de l'agent i et la position finale q_b de la ressource $b \in \mathcal{R}_{nl}$. w_b est le temps d'attente à la ressource et opt_{nl} est le temps de traitement de l'opération. De ce critère, on remarque que le choix de ressource se fait par rapport au temps de transport et au temps d'attente de la ressource. En effet, le temps de traitement opt_{nl} est le même quelque soit la ressource pouvant effectuer l'opération. Ainsi, si le temps d'attente à une ressource est élevé, il y a plus de chance que l'agent choisisse une autre ressource. De la même manière, si l'agent met plus de temps à arriver à une ressource (car elle est éloignée ou parce qu'il y a beaucoup de collisions à éviter), une autre ressource pourrait être choisie.

2.3 Principales hypothèses

Les deux approches introduites dans les chapitres suivants sont soumises à un certain nombre d'hypothèses. La Fig. 2.3 est proposée pour expliquer certaines de ces hypothèses afin d'améliorer la compréhension du problème. Les hypothèses sont de deux types : celles en relation avec les différentes entités du système manufacturier et celles propres aux AGVs.

Notons que les approches proposés dans les chapitres 3 et 4 sont centrées sur l'achèvement d'une seule opération et non du produit transporté. Ceci est principalement dû au fait que les travaux de cette thèse se focalisent sur les interactions entre le haut et le bas niveau plutôt que sur la production des produits en elle-même.

Le système manufacturier : hypothèses et contraintes

Étant dans un système de production flexible, il est préférable de mentionner l'ensemble des hypothèses liées aux produits $l \in \mathcal{J}$, à ses différentes opérations $n \in \mathcal{O}_l$ et aux ressources $b \in \mathcal{R}$.

D'abord, chaque agent i ne transporte qu'un seul produit l, d'une ressource $a \in \mathcal{R}_{(n-1)l}$ ayant effectuée son opération précédente $n-1$, vers une autre ressource $b \in \mathcal{R}_{nl}$ permettant d'accomplir son opération en cours n. De plus, un produit l n'est transporté que par un agent i uniquement (les produits volumineux, transportés par plusieurs AGVs, ne sont pas traités) jusqu'à ce qu'il soit achevé. Rappelons que le transport des produits est libre et qu'il n'y a aucun chemin physique ou virtuel imposé.

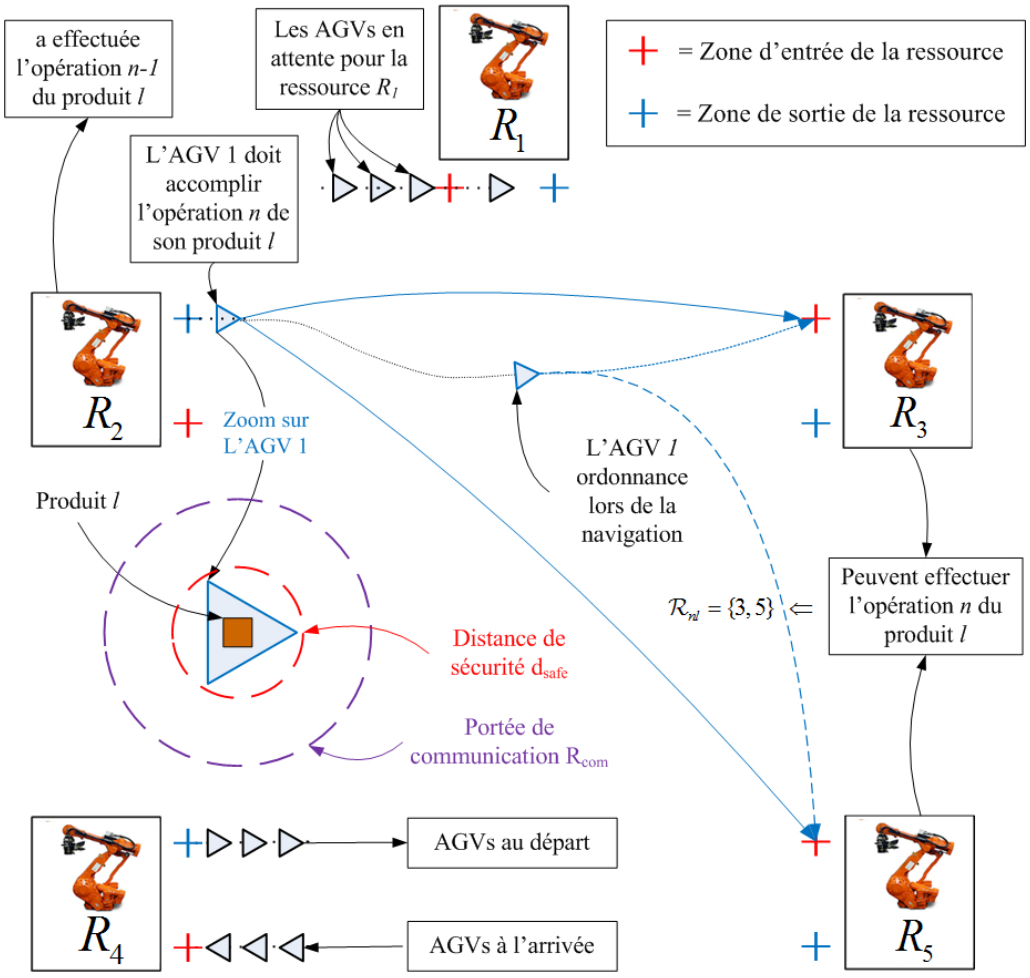


FIGURE 2.3 – Explications des variables et hypothèses pour le problème considéré

Lorsqu'un agent $i \in \mathcal{A}$ est disponible, le niveau G.O.R.P lui assigne un produit $l \in \mathcal{J}$. Chaque produit l a une séquence d'opérations à suivre $\mathcal{O}_l$, où les opérations sont traitées une à une. Pour chaque opération $n \in \mathcal{O}_l$ d'un produit assigné l, un cahier des charges de production est donné par le niveau G.O.R.P. Ce cahier des charges inclut un ensemble de ressources $\mathcal{R}_{nl}$ où l'opération n peut être effectuée et une échéance odd_{nl} pour laquelle l'opération doit être achevée préalablement. De plus, le cahier des charges est supposé faisable et n'est pas modifié sur l'intervalle de temps considéré, c'est–à–dire que l'ensemble de ressources et l'échéance ne peuvent être changés lors de la navigation.

Afin de respecter le cahier des charges de production, l'agent doit choisir une ressource $b \in \mathcal{R}_{nl}$ en considérant la contrainte temporelle liée à l'échéance odd_{nl} de l'opération n du produit l. Il faut donc que l'opération soit achevée avant l'échéance de celle-ci, ce qui se formule par la contrainte suivante :

$$ct_{nlb} \leq odd_{nl}, \qquad \forall n \in \mathcal{O}_l, \forall l \in \mathcal{J}, b \in \mathcal{R}_{nl} \tag{2.3.1}$$

Notons que cette échéance odd_{nl} permet d'empêcher un produit de rester trop longtemps dans la production ou de changer de ressource lorsque celle–ci est surchargée (temps d'attente élevé). Le terme échéance est ici utilisé par abus de langage car il n'est en aucun cas lié à l'échéance du produit utilisée en production dite "juste à temps". L'échéance représente plutôt un temps pour lequel on voudrait achever l'opération, c'est–à–dire un objectif (ou une stratégie) que l'entreprise se fixe et non une contrainte provenant de son ou ses clients. Par conséquent, on peut caractériser cette échéance comme les performances minimales désirées par l'entreprise en termes de temps pour achever l'opération.

Toutes les ressources $b \in \mathcal{R}$, pouvant être considérées comme des machines effectuant les opérations des produits, sont supposées immobiles. Lorsqu'une opération est achevée par une ressource, l'agent démarre de cette ressource par sa zone de sortie. Inversement, l'arrivée des agents à une ressource se fait par sa zone d'entrée, qui peut être vue comme une zone de stockage où les agents attendent la disponibilité de la ressource. Pour chaque ressource, les zones d'entrée et de sortie sont à des positions différentes afin d'éviter les collisions entre les agents entrant et sortant. Les files d'attente des ressources sont supposées être à capacité infinie, c'est–à–dire que le nombre d'agents pouvant attendre à une ressource n'est pas limité. Le temps d'attente à une ressource w_b correspond au temps pour lequel un agent doit attendre à la ressource $b \in \mathcal{R}$ avant de commencer l'opération du produit transporté. Les perturbations liées aux ressources, telles que les pannes machines ou opérations de maintenance, ne sont pas considérées dans les approches proposées.

Lorsque plusieurs agents se dirigent vers une même ressource, il est nécessaire de savoir l'ordre d'arrivée des agents à cette ressource. Cet ordre d'arrivée se fait à l'aide de l'ensemble $\mathcal{HP}_i$ qui représente l'ensemble $\mathcal{N}_i$ des voisins j qui ont un niveau de priorité plus haut que l'agent i. Cet ensemble est calculé différemment selon l'approche considérée et sera introduit dans chaque chapitre lorsque nous en aurons besoin. L'ensemble des voisins $\mathcal{N}_i$ sera défini plus bas à l'aide de l'équation (2.3.9). Par conséquent, l'arrivée d'un agent i est contraint de la manière suivante :

$$T_{i,fin} > T_{j,fin}, \forall j \in \mathcal{HP}_i : cr_i(upt_i) = cr_j(upt_j) \tag{2.3.2}$$

où $T_{i,fin}$ est l'instant d'arrivée de l'agent i, qui dépend du temps de transport vers la ressource choisie TT_{ib} induit par la trajectoire générée $q_i(t)$ de l'agent i. De la même façon, $T_{j,fin}$ est l'instant d'arrivée de l'agent j à cette même ressource b. L'égalité $cr_i(upt_i) = cr_j(upt_j)$ indique le choix identique de la ressource b par les agents i et j aux instants de mises à jour respectifs. Ainsi, un agent i est contraint d'arriver après chaque voisin j se dirigeant vers la ressource choisie à chaque moment où l'agent i met à jour sa trajectoire. En plus de permettre une mise en place de l'ordre d'arrivée aux ressources, cette contrainte est très utile pour l'ordonnancement. En effet, si beaucoup d'agents se dirigent vers une même ressource, les agents avec un niveau de priorité plus bas pourront choisir une autre ressource le cas échéant.

Les agents : hypothèses et contraintes

Les agents (ou AGVs) sont aussi soumis à un certain nombre d'hypothèses afin de prendre en compte l'aspect dynamique de la navigation. D'abord, il est supposé que tous les agents sont identiques en termes de capacités des dispositifs de calcul, en termes de comportement physique et ont tous la même taille et portée de communication. De plus, la gestion de l'énergie des batteries des agents en termes de charge et de décharge n'est pas traitée. Notons que comme on considère le traitement d'une seule opération, la navigation de chaque agent i se fait sur un intervalle restraint $[T_{i,init}, T_{i,fin}]$ où $T_{i,init}$ est l'instant initial où l'agent i démarre de la ressource $a \in \mathcal{R}_{(n-1)l}$ et $T_{i,fin}$ est l'instant d'arrivée à la ressource $b \in \mathcal{R}_{nl}$ pour effectuer l'opération en cours n. Ainsi, toutes les contraintes dynamiques données ci–dessous devront être respectées sur cet intervalle de temps $[T_{i,init}, T_{i,fin}]$. Les comportements dynamiques des agents sont régis par l'équation, $\forall i \in \mathcal{A}$:

$$\dot{q_i}(t) = v_i(t), \quad \forall t \in [T_{i,init}, T_{i,fin}] \tag{2.3.3}$$

où $q_i = (x_i, y_i)^T \in \mathbb{R}^2$ est la position physique de l'agent i et $v_i = (V_{x_i}, V_{y_i})^T \in \mathbb{R}^2$ sa vitesse. Notons que l'utilisation de ce modèle permet de simplifier les problèmes d'optimisation qui ont plus de contraintes à cause du problème de production. Ceci permet notamment de réduire la complexité des problèmes et surtout le temps de calcul pour les résoudre. Par conséquent, les trajectoires risquent, dans certains cas, de ne pas respecter les contraintes de non holonomie [Defoort, 2007]. Cependant, il est possible de suivre cette trajectoire à l'aide d'une commande robuste [Defoort et al., 2005], mais le suivi sera moins précis sur les zones (de la trajectoire) où ces contraintes ne sont pas respectées.

La vitesse de chaque agent i est bornée comme suit :

$$\|v_i(t)\| \leq v_{max}, \quad \forall t \in [T_{i,init}, T_{i,fin}] \tag{2.3.4}$$

où $v_{max} \in \mathbb{R}$ est la vitesse maximum tolérée pour chaque agent et $\|v_i(t)\|$ est la norme euclidienne de la vitesse définie par :

$$\|v_i(t)\| = \sqrt{(V_{x_i})^2 + (V_{y_i})^2}$$

Chaque agent connaît sa position initiale de départ $q_i(T_{i,init}) = q_{i,init}$, sa position actuelle q_i à chaque instant $t \in [T_{i,init}, T_{i,fin}]$ et les positions finales de chaque ressource q_b ($b \in \mathcal{R}_{nl}$) qui dépendent de l'opération n en cours de production. De la même manière, chaque agent démarre de sa position initiale et arrive à sa position finale à vitesse nulle et connaît sa vitesse actuelle v_i à chaque instant. Ceci se traduit par les conditions aux limites suivantes :

$$q_i(T_{i,init}) = q_a, \quad a \in \mathcal{R}_{(n-1)l} \tag{2.3.5}$$

$$\dot{q_i}(T_{i,init}) = 0 \tag{2.3.6}$$

$$q_i(T_{i,fin}) = q_b, \quad b \in \mathcal{R}_{nl} \tag{2.3.7}$$

$$\dot{q}_i(T_{i,fin}) = 0 \tag{2.3.8}$$

En termes de communication, la position des agents et la portée limitée des émetteurs/receveurs imposent la topologie de communication. Celle–ci varie dans le temps et peut être exprimée par un graphe de communication non dirigé. Ce graphe est constitué d'un ensemble de noeuds $\mathcal{V} \subset \mathcal{A}$ et un ensemble de liens $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Un lien $\varepsilon_{ij} \in \mathcal{E}$ correspond à une liaison de communication entre les agents i et j. Pour chaque agent, on définit l'ensemble actuel des voisins $\mathcal{N}_i$ à proximité, c'est–à–dire où il y a une liaison entre l'agent i et les agents j à l'instant actuel, de la manière suivante :

$$\mathcal{N}_i = \{j \in \mathcal{A}, \|q_i - q_j\| < R_{com}\} \tag{2.3.9}$$

où $\|q_i - q_j\| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ est la distance entre les agents i et j à l'instant actuel et R_{com} représente le rayon de la portée de communication, centrée en q_i.

Pour finir, comme un cadre multi-agents est considéré, il faut éviter les collisions entre les agents. Cet évitement se fait à l'aide d'une coordination par leadership où chaque agent i doit éviter les collisions avec ses voisins j ayant un niveau de priorité plus haut, i.e. $j \in \mathcal{HP}_i$. Ceci se traduit par l'inégalité suivante :

$$\|q_i(t) - q_j(t)\| \geq d_{safe}, \qquad \forall j \in \mathcal{HP}_i,\ \forall t \in [T_{i,init}, T_{i,fin}] \tag{2.3.10}$$

où d_{safe} est la distance entre les agents à respecter, dépendant de la forme géométrique de l'AGV. La raison pour laquelle une coordination par leadership est choisie est due au contexte manufacturier. En effet, l'évitement de collision d'un agent le fait dévier de sa trajectoire optimale, ce qui peut retarder l'arrivée à la ressource qu'il choisit. Ce retard peut avoir un impact conséquent sur les performances du produit. De plus, comme un cahier des charges doit être respecté pour chaque produit, plusieurs déviations consécutives pourrait empêcher le respect de celui–ci. Avec une coordination par leadership, le "leader" n'a pas besoin de dévier de sa trajectoire et peut suivre sa trajectoire optimale. Par conséquent, il est évident qu'une coordination par leadership est plus appropriée pour notre contexte. La manière dont le "leader" est déterminé a une grande importance sur le problème et devra donc dépendre des performances (en terme de temps pour achever l'opération) du produit transporté et sur le cahier des charges imposé à l'agent.

Notons que l'évitement des collisions avec des obstacles, autres que des agents, n'est pas considéré dans notre approche. Ce choix est basé sur deux faits. D'une part, comme les agents naviguent toujours dans la même zone (de production), les obstacles stationnaires serait connus à l'avance. De plus, on pourrait éviter de mettre des obstacles stationnaires "en plein milieu" de la zone de navigation lors de la conception d'un système de production. Pour un exemple de conception de ce type de système, on pourrait mettre toutes les ressources autour de la zone de production comme sur la Fig. 2.3.

D'autre part, les obstacles non stationnaires seront principalement des humains. Comparé à l'utilisation des AGVs dans les entrepôts où les humains se déplacent régulièrement et librement, les humains sont principalement présents lors d'opérations de maintenance des AGVs dans un contexte de production. Ainsi, il n'y a aucune raison qu'un humain soit présent en fonctionnement "normal". Comme les incertitudes ne sont pas considérées, les obstacles non stationnaires (humains), présents dans la zone de navigations, ne sont pas considérés.

Synthèse sur le problème et sa formulation

Suite à l'objectif des agents et aux hypothèses données, le problème à résoudre pour chaque agent i est le suivant : en démarrant de la ressource $a \in \mathcal{R}_{(n-1)l}$ ayant effectuée la dernière opération $n-1$, déterminer la trajectoire $q_i(t)$ et la ressource $b \in \mathcal{R}_{nl}$ permettant d'achever l'opération en cours n du produit l transporté par l'agent i le plus tôt possible. Ce problème se formule pour chaque agent i de la manière suivante :

$$\min_{q_i(t),\, b} ct_{nlb} = T_{i,init} + TT_{ib} + w_b + opt_{nl} \tag{2.3.11}$$

sous les contraintes de production :

$$ct_{nlb} \leq odd_{nl}, \quad \forall n \in \mathcal{O}_l, \forall l \in \mathcal{J}, b \in \mathcal{R}_{nl} \tag{2.3.12}$$

$$T_{i,fin} > T_{j,fin}, \quad \forall j \in \mathcal{HP}_i : cr_i(upt_i) = cr_j(upt_j) \tag{2.3.13}$$

et les contraintes liées aux agents :

$$\dot{q_i}(t) = v_i(t), \quad \forall t \in [T_{i,init}, T_{i,fin}] \tag{2.3.14}$$

$$\|v_i(t)\| \leq v_{max}, \quad \forall t \in [T_{i,init}, T_{i,fin}] \tag{2.3.15}$$

$$q_i(T_{i,init}) = q_a, \quad a \in \mathcal{R}_{(n-1)l} \tag{2.3.16}$$

$$\dot{q}_i(T_{i,init}) = 0 \tag{2.3.17}$$

$$q_i(T_{i,fin}) = q_b, \quad b \in \mathcal{R}_{nl} \tag{2.3.18}$$

$$\dot{q}_i(T_{i,fin}) = 0 \tag{2.3.19}$$

$$\|q_i(t) - q_j(t)\| \geq d_{safe}, \quad \forall j \in \mathcal{HP}_i, \forall t \in [T_{i,init}, T_{i,fin}] \tag{2.3.20}$$

Notons que le problème ci–dessus est formulé d'une manière déterministe où tout est connu à l'avance. Cependant, il est important de noter que ce problème sera traité dynamiquement, c'est–à–dire que les décisions d'ordonnancement et de génération de trajectoire changent lors de la navigation. Par conséquent plusieurs modifications seront faites pour prendre en compte ce dynamisme. D'abord, l'instant initial $T_{i,init}$ sera remplacé par l'instant actuel (i.e. moment de mise à jour) T_{i,upt_i} dans le critère (2.3.12) et pour les contraintes (2.3.15)–(2.3.18) et (2.3.20). Les contraintes (2.3.17) et (2.3.18) deviennent, respectivement, $q_i(T_{i,upt_i}) = q_i$ et $\dot{q}_i(T_{i,upt_i}) = v_i$ où q_i est la position actuelle et v_i est la vitesse actuelle.

Ensuite, les contraintes (2.3.14) et (2.3.20) dépendent de l'ensemble $\mathcal{HP}_i$ représentant les agents j ayant un niveau de priorité plus haut que l'agent i. Notons que cet ensemble est variable dans le temps, c'est–à–dire qu'il dépend des voisins à l'instant actuel T_{i,upt_i}. Enfin, les trajectoires étant mises à jour lors de la navigation, un second argument T_{i,upt_i} est spécifié pour chaque trajectoire $q_i(t)$ permettant de savoir à quel instant la trajectoire est mise à jour. La trajectoire de l'agent i sera notée $q_i(t, T_{i,upt_i})$ où t désigne le temps et T_{i,upt_i} l'instant de mise à jour.

Le problème étant formulé, il est important de noter que les trajectoires des agents sont mises sous forme paramétrique à l'aide de courbes splines présentées dans la section ci–dessous.

2.4 Mise sous forme paramétrique des trajectoires

Afin de transformer les problèmes de planification de trajectoire en problèmes de programmation non linéaire, il est préférable de paramétriser les trajectoires. Il existe plusieurs types de courbes pouvant être utilisées afin de spécifier une trajectoire telles que les polynômes, les séries de Fourier, etc. Ces courbes doivent permettre de représenter, de manière précise et avec un nombre raisonnable de paramètres à déterminer, les éléments de la solution du problème de planification de trajectoire. De plus, il est préférable que le choix de la courbe n'engendre pas un ajout de contraintes supplémentaires voire que ce choix permette de retirer une ou plusieurs contraintes afin de réduire la complexité du problème.

Comme les polynômes sont différentiables et facilement évaluables, ils peuvent palier une partie de ces difficultés. Cependant, ils nécessitent d'utiliser d'un grand nombre de paramètres lorsque l'horizon de planification devient grand. De plus, les paramètres sont dépendants les uns des autres, ce qui rend l'ajustement complet de la courbe mauvais s'il l'est localement. Ces inconvénients nous empêchent de trouver un polynôme adéquat en un temps de calcul très court, rendant une implementation en ligne plus difficile voire impossible dans certains cas.

Pour les problèmes considérés, nous avons besoin de courbes qui dépendent d'un nombre suffisant de paramètres afin de satisfaire les différentes contraintes. Il est aussi nécessaire que les effets des paramètres de la courbe soient facilement estimables et que ces paramètres aient peu d'influence les uns sur les autres sur une zone réduite. Ceci permettrait de générer plus rapidement la courbe correspondant à la solution des problèmes d'optimisation de planification de trajectoire. Par conséquent, une implémentation en ligne deviendrait envisageable.

Une solution permettant de répondre à ce besoin est d'utiliser des courbes splines pour représenter les trajectoires. Ces courbes sont construites par combinaison linéaire de fonctions splines de base, appelées B–splines. Une courbe spline est un ensemble de polynômes continus par morceaux correspondant au raccordement de courbes de Bézier. L'avantage principal est de pouvoir gérer la continuité aux points de raccordement entre les B–splines afin que la courbe spline ait le degré de continuité voulu. Une étude exhaustive des fonctions splines est proposée dans [De Boor et al., 1978].

Base des B–splines

Les B–splines dépendent d'une suite de noeuds $\{\text{nod}_0 \leq \ldots \leq \text{nod}_{N_{nod}}\}$. Cette suite non décroissante de réels positifs contient l'ensemble des points de raccordement entre les différentes B–splines. Les B–spline d'ordre $d = 1$ (c.f. Fig. 2.4–(A)) sont, pour tout $a = \{0, \ldots, N_{nod} - 1\}$, définies par :

$$B_{a,1}(t) = \begin{cases} 1 & \text{si } \text{nod}_a \leq t < \text{nod}_{a+1} \\ 0 & \text{sinon} \end{cases} \tag{2.4.1}$$

Les B–splines d'ordre supérieur, i.e. $d \geq 2, d \in \mathbb{N}$, se déduisent par relation de récurrence de la manière suivante, pour tout $a = \{0, \ldots, N_{nod} - d\}$:

$$B_{a,d}(t) = \frac{t - \text{nod}_a}{\text{nod}_{a+d+1} - \text{nod}_a} B_{a,d-1}(t) + \frac{\text{nod}_{a+d} - t}{\text{nod}_{a+d} - \text{nod}_{a+1}} B_{a+1,d-1}(t) \tag{2.4.2}$$

Notons d'une manière générale que certains noeuds peuvent être confondus. Dans ce cas et par convention, on annule les $B_{j,d}$ qui présentent un dénominateur nul dans leur définition. On obtient ainsi des discontinuités dans le graphe des B-splines [Defoort, 2007]. Le nombre d'occurrence d'un point de raccordement est appelé multiplicité du point de raccordement. Avec les mêmes notations, les B–splines sont assujetties aux deux propriétés suivantes :

- $B_{a,d}(t) \geq 0$ sur $[\text{nod}_a, \text{nod}_{a+d}]$ et $B_{a,d}(t) = 0$, $\forall\, t \notin [\text{nod}_a, \text{nod}_{a+d}]$.
- Les B-splines d'ordre d sont des fonctions polynomiales par morceaux de degré au plus $d - 1$. Si les noeuds sont tous distincts, ce sont des fonctions de classe C^{d-2}. Lorsqu'un noeud est de multiplicité $1 \leq r \leq d - 1$, la B-spline n'est que $d - r - 1$ fois continûment dérivable en ce point. Si un noeud est de multiplicité d, il apparaît alors une discontinuité dans le graphe de la B-spline.

Lorsqu'elles existent, les dérivées successives des B–splines peuvent être obtenues par des équations de récurrence, où la multiplicité des noeuds est prise en compte, comme suit :

$$\dot{B}_{a,d}(t) = (d-1)\left(\frac{B_{a,d-1}(t)}{\text{nod}_{a+d-1} - \text{nod}_a} - \frac{B_{a+1,d-1}(t)}{\text{nod}_{a+d} - \text{nod}_{a+1}}\right) \tag{2.4.3}$$

Afin d'illustrer les B–splines, nous proposons quelques exemples, mis en évidence sur la Fig. 2.4, sur un cas où les noeuds sont entiers et distincts (uniformes), c'est–à–dire l'ensemble de noeuds $\{0, 1, 2, \ldots\}$.

Pour ce cas particulier, il y a des propriétés d'invariance par translation (i.e. $B_{a,d}(t) = B_{0,d}(t-a)$) et de symétrie (i.e. $B_{0,d}(t) = B_{0,d}(d+1-t)$). Les exemples de B–splines de cette figure sont les suivants :

- La B–spline de degré 1 (Fig. 2.4–(B)) est continue, non dérivable et composée de polynômes linéaires par morceaux :

$$B_{0,2}(t) = \begin{cases} t & \text{pour } 0 \leq t \leq 1 \\ 2 - t & \text{pour } 1 \leq t \leq 2 \end{cases}$$

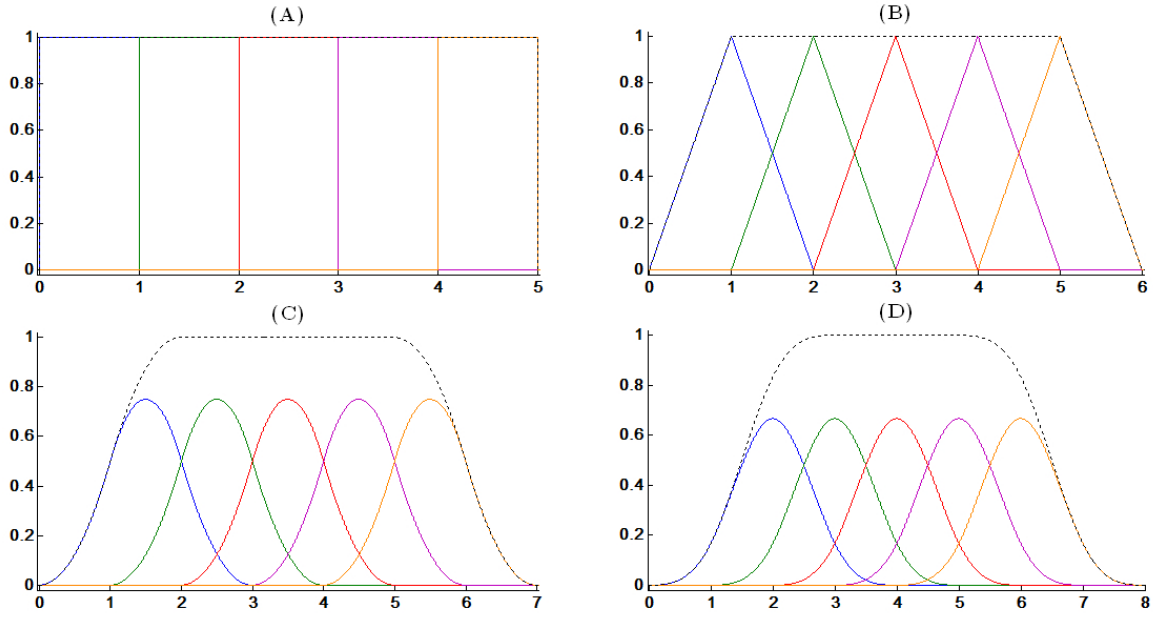


FIGURE 2.4 – Exemples de B–splines avec noeuds distincts uniformes : (A) degré 0, (B) degré 1, (C) degré 2, (D) degré 3

- La B–spline de degré 2 (Fig. 2.4–(C)) est de classe C^1 et composée de polynômes quadratiques par morceaux :

$$B_{0,3}(t) = \begin{cases} \frac{t^2}{2} & \text{pour } 0 \leq t \leq 1 \\ \frac{-2t^2+6t-3}{2} & \text{pour } 1 \leq t \leq 2 \\ \frac{(3-t)^2}{2} & \text{pour } 2 \leq t \leq 3 \end{cases}$$

- La B–spline de degré 3 (Fig. 2.4–(D)) est de classe C^2 et composée de polynômes cubiques par morceaux :

$$B_{0,4}(t) = \begin{cases} \frac{t^3}{6} & \text{pour } 0 \leq t \leq 1 \\ \frac{-3t^3+12t^2-12t+4}{6} & \text{pour } 1 \leq t \leq 2 \\ \frac{3t^3-24t^2+60t-44}{6} & \text{pour } 2 \leq t \leq 3 \\ \frac{(4-t)^3}{6} & \text{pour } 3 \leq t \leq 4 \end{cases}$$

Les courbes splines

Une courbe spline (notée ici $S_d(t)$) est définie par une combinaison linéaire de B–splines d'ordre d, pondérée par $p+1$ points $P_0, \ldots, P_p$, appelés points de contrôle. Soit $t_0 \leq t_1 \leq \ldots \leq t_m, m \geq p+1+d$, la courbe spline est décrite par :

$$S_d(t) = \sum_{a=0}^{p} P_a B_{a,d}(t) \tag{2.4.4}$$

où p est défini par la relation suivante :

$$p = n_{\text{knot}}(d-1) - \sum_{k=1}^{n_{\text{knot}}-1} s_k$$

avec

- n_{knot} : le nombre d'intervalles $[\text{nod}_a, \text{nod}_{a+1}]$ $(a = 1, \ldots, M-1)$ de longueur non nulle,
- s_k $(k = 1, \ldots, n_{\text{knot}} - 1)$: le degré de continuité de chaque point de raccordement intérieur (i.e. dans l'intervalle ouvert $(\text{nod}_0, \text{nod}_{N_{nod})}$).

Les courbes splines sont sujettes à l'ensemble de propriétés suivantes :

- Dans la plupart des applications, les points intérieurs de raccordement ont le même degré de continuité s, ce qui implique :
$$p = n_{\text{knot}}(d-1-s) + s.$$
- La somme des B–splines est toujours égale à 1 sur l'intervalle $[t_0, t_m]$ du paramètre t, de telle sorte que $S_d(t)$ soit le barycentre à poids positifs des points de contrôle P_a. Ainsi, la courbe spline se situe dans l'enveloppe convexe de son S–polygone (voir Fig. 2.5–(B)). Un S–polygone ou polygone spline est la ligne brisée joignant les points de contrôle $\{P_0, \ldots, P_p\}$.
- Un point de la courbe ne dépend que d'au plus $d+1$ points de contrôle. Si $t_b \leq t < t_{b+1}$, la courbe $S_d(t)$ ne dépend que des points P_a pour lesquels $b - p \leq a \leq b$. Inversement, le point P_b n'influence la courbe $S_d(t)$ que pour les points tels que $t_b \leq t < t_{b+d+1}$.
- En général, la courbe spline ne passe pas par les points de contrôle P_a. Ceci est le cas lorsque tous les noeuds sont distincts, c'est–à–dire de multiplicité 1. Inversement, la courbe passe par les points de contrôles associés à des noeuds de multiplicité supérieure ou égale à 2, ce qui peut être très utile si l'on désire qu'une trajectoire passe par certains points.

Intérêts des courbes splines pour la génération de trajectoires

Soient un entier $d > 1$ et $\{\eta_0 = T_i < \eta_1 < \ldots < \eta_{N_{nod}} = T_f\}$ une subdivision uniforme de l'intervalle $[T_i, T_f]$ où N_{nod} est un entier fixé et non nul. A partir de cette subdivision, on construit la suite de noeuds, sur laquelle les B-splines sont raccordées, de la manière suivante :

$$\{\text{nod}_0 = \ldots = \text{nod}_{d-1} = \delta_0 < \text{nod}_d = \delta_1 < \ldots < \text{nod}_{d-1+N_{\text{nod}}} = \ldots = \text{nod}_{2d-2+n_{\text{knot}}} = \delta_{n_{\text{knot}}}\} \tag{2.4.5}$$

La trajectoire $q(t)$ peut alors s'écrire comme une combinaison linéaire des B–splines d'ordre d :

$$q(t) = \sum_{a=0}^{d+N_{\text{nod}}-2} P_a B_{a,d}(t) \tag{2.4.6}$$

où les points $P_a \in \mathbb{N}^m$ sont des vecteurs de taille m ayant un impact local. Lorsqu'on recherche une trajectoire, on recherche dans un premier temps l'ensemble des points de contrôle. De plus, la variable T_f représente le temps final pour lequel on atteint la position $q(T_f)$. Si l'on recherche ce temps final, on change aussi la suite de noeud (2.4.5).

D'une part, on remarque que l'ordre des B–splines d et la suite de noeuds (2.4.5) sont choisis par rapport au nombre de variables à optimiser, des contraintes à respecter et du degré de continuité des dérivées voulu. Il faut donc bien choisir l'entier N_{nod} afin de réduire le temps de calcul sans pour autant augmenter l'erreur d'approximation.

D'autre part, on peut voir que les noeuds aux temps initiaux et finaux sont de multiplicité supérieure à 2. On dit dans ce cas que la courbe spline est vissée aux extrémités. Ceci a un grand intérêt car la trajectoire résultante passe par les points de contrôle P_0 et $P_{d+N_{\mathrm{nod}}-2}$. Par conséquent, cette propriété permet de prendre en compte les contraintes aux limites d'une manière implicite.

Pour illustrer ces propos, nous proposons un exemple avec 9 points de contrôle, des B–splines d'ordre 4 (ou de degré 3) et l'ensemble de noeud pour l'intervalle $[T_i = nod_0, T_f = nod_{12}]$ suivant :

$$\{\mathrm{nod}_0 = \ldots = \mathrm{nod}_3 < \mathrm{nod}_4 < \ldots < \mathrm{nod}_9 = \ldots = \mathrm{nod}_{12}\} \tag{2.4.7}$$

La trajectoire désirée $q(t)$ est définie par

$$q(t) = \sum_{a=0}^{8} P_a B_{a,d}(t)$$

où les B–splines sont données par la Fig. 2.5–(A) pour l'ensemble de noeuds donné et P_a sont les points de contrôle .

En termes de conditions aux limites, la configuration initiale est celle actuelle $q(T_i) = q$ avec une vitesse actuelle $v(T_i) = v$ et la configuration finale à atteindre est $q(T_f) = q_{des}$ à vitesse nulle (i.e. $v(T_f) = 0$). Les points de contrôle aux extrémités sont fixés comme suit :

$$P_0 = q \tag{2.4.8}$$
$$P_1 = P_0 + \gamma \cdot v \cdot (T_f - T_i) \tag{2.4.9}$$
$$P_7 = q_{des} \tag{2.4.10}$$
$$P_8 = P_7 \text{ (car } v(T_f) = 0) \tag{2.4.11}$$

où γ est un paramètre dont la valeur est obtenue à l'aide de simulations. On remarque que la position initiale dépend uniquement du point de contrôle initial P_0. La vitesse initiale dépend des points P_0 et P_1 ainsi que de l'intervalle $[T_i, T_f]$. Ceci est respectivement valable pour les conditions terminales (position et vitesse finales) qui dépendent des points de contrôle finaux (ici P_7 et P_8).

La Fig. 2.5–(B) permet de souligner le respect des conditions aux limites données par les équations précédentes, avec pour exemple $q = 2$, $v = 0.8$ et $q_{des} = 3$ et des points de contrôle $P_a, a = \{2, \ldots, 6\}$ choisis aléatoirement :

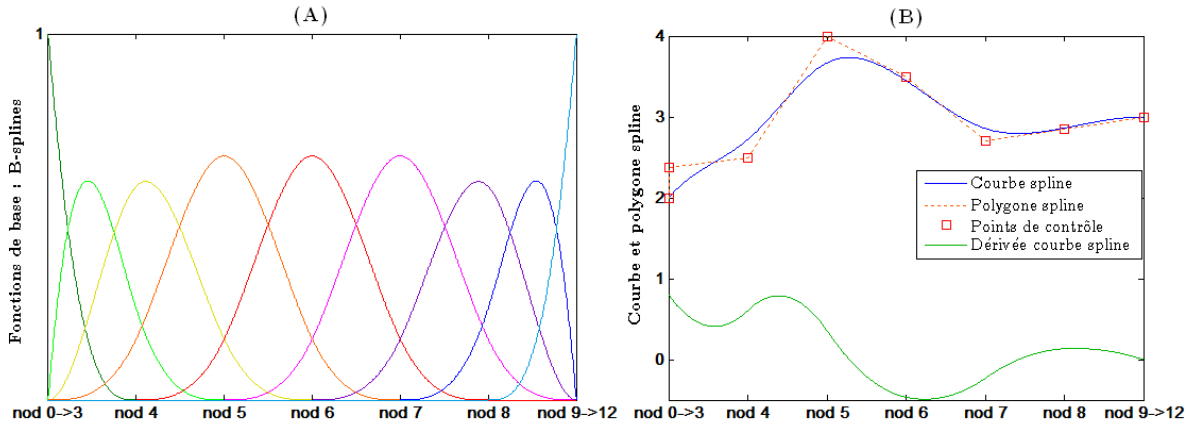


FIGURE 2.5 – (A) B–splines d'ordre 4 pour l'ensemble de noeuds donnés (B) Exemple de courbe et polygone spline

Pour conclure, les problèmes de recherche de trajectoires n'ont pas besoin d'inclure les contraintes de condition aux limites qui sont fixées par les points de contrôle des extrémités. En effet, pour générer une trajectoire, il ne reste qu'à déterminer l'ensemble des points de contrôle *centraux* (i.e. ceux non fixés par les conditions aux limites) ainsi que le temps final T_f pour lequel on atteindra la configuration désirée.

2.5 Conclusion

Ce chapitre nous a permis de formuler le problème de planification de trajectoire de chaque agent. Cette formulation sera utile pour la compréhension des approches proposées dans les chapitres 3 et 4. Nous avons vu que la fonction d'ordonnancement, permettant de sélectionner une ressource, est incluse dans l'algorithme de navigation. Plus précisément, cette fonction est combinée à la fonction de planification de trajectoire. Dans la suite de ce mémoire, lorsque nous mentionnons planification de trajectoire, cela inclut implicitement la fonction d'ordonnancement.

Afin de montrer la vision générale du système manufacturier dans lequel les AGVs naviguent, un ensemble d'hypothèses a été fourni. De plus, l'objectif de production consistant à achever l'opération du produit transporté par l'AGV a été décrit. Rappelons que les approches présentées dans les deux prochains chapitres ne traitent qu'une opération seulement. Ceci permet de montrer d'une part la difficulté du problème pour achever cette unique opération mais aussi de voir le challenge pour traiter l'ensemble des opérations de chaque produit. Ce challenge sera discuté dans les perspectives car il dépend d'autres notions (comme la gestion de la batterie des AGVs, non traitée dans cette thèse),

utiles pour l'implémentation de telles approches en industrie.

Enfin, les trajectoires calculées à l'aide des algorithmes proposés dans les chapitres suivants seront paramétrisées à l'aide de courbes splines dont les avantages ont été données. L'utilisation de ces courbes permet de simplifier la résolution des problèmes de planification de trajectoire dû aux propriétés des B–splines et des courbes résultantes. En effet, uniquement les points de contrôle centraux sont à déterminer et les contraintes de conditions aux limites sont respectées d'une manière implicite.

Pour les deux prochains chapitres, deux approches différentes, permettant de résoudre le problème présenté dans ce chapitre, seront introduites. Ces différences sont principalement les suivantes :

- Les architectures de pilotages sont différentes dans chaque approche. Dans le chapitre 3, une architecture hétérarchique est proposée où chaque AGV doit résoudre les conflits par lui–même. Dans le chapitre 4, un superviseur est inclut dans l'architecture, permettant de se rapprocher d'une architecture hybride. Ce superviseur permet d'assister les AGVs pour les différents conflits, ce qui réduit la myopie comparé à une architecture hétérarchique.

- Le planificateur de trajectoire, combiné avec la fonction d'ordonnancement diffère en deux points.

  – Pour chaque approche, la mise à jour des trajectoire et de l'ordonnancement se fait d'une manière différente, permettant d'empêcher une mise à jour simultanée de ressource par des agents voisins. Dans le chapitre 3, une mise à jour de la trajectoire et de l'ordonnancement par évènement est proposée. Chaque agent ne peut pas mettre à jour cette combinaison (i.e. planification de trajectoire et choix de machine) en même temps qu'un autre et doit attendre que ses voisins ayant un niveau de priorité plus haut aient mis à jour leur combinaison. Dans le chapitre 4, tous les agents mettent à jour leur trajectoire aux mêmes instants. Cependant, ils ne sont pas tous autorisés à ordonnancer leur produit à chaque instant de mise à jour.

  – En termes d'algorithme, le chapitre 3 propose pour chaque agent un planificateur de trajectoire en une seule étape. Chaque agent planifie sa trajectoire sans collision et met à jour sa ressource par rapport à ses voisins. Cependant, il n'y a aucune prévision des intentions des autres agents non voisins dû au phénomène de myopie, ce qui peut mener à des conflits difficiles à résoudre. Dans le chapitre 4, une planification de trajectoire en deux étapes est proposée. La première étape permet de planifier l'intention des robots, en termes d'évitement et de choix de resource (si l'ordonnancement est autorisé), à l'aide d'informations fournies par le superviseur, ce qui réduit la myopie. La seconde étape utilise ces intentions pour planifier une trajectoire sans collision vers la ressource spécifiée à la première étape.

Dans la suite de ce mémoire, ces deux approches sont présentées dans les deux chapitres suivants. Nous verrons ces différences qui se caractérisent principalement en termes d'architecture, de résolution des conflits et d'algorithme de planification de trajectoire.

Chapitre 3

Planification de trajectoire : approche hétérarchique

3.1 Introduction

Le chapitre 2 nous a permis d'introduire une vue d'ensemble du problème de planification de trajectoire d'un système multi–agents en milieu manufacturier. L'objectif est, pour chaque agent, de compléter l'opération en cours le plus tôt possible. Pour cela, ce chapitre va nous permettre d'introduire une approche où l'architecture est complètement hétérarchique. Ce qui signifie que chaque agent, en plus de planifier sa trajectoire et ordonnancer son produit, devra résoudre lui–même les conflits en coopérant avec les agents à portée de communication. Comme nous l'avons vu, cette hétérarchie est fortement sensible à la myopie d'une manière temporelle et sociale.

En plus des principales hypothèses introduites dans le chapitre 2, cette architecture hétérarchique stipule qu'aucune communication n'est faite avec le niveau G.O.R.P pendant qu'un agent navigue entre deux ressources. Inversement, les informations d'un produit transporté par un agent sont renvoyées au niveau G.O.R.P uniquement lorsque celui–ci atteint une ressource. Ce qui signifie que les agents n'ont aucune connaissance des temps d'attente aux ressources et que ces temps d'attente ne peuvent être prédits. On se limite donc aux systèmes manufacturiers dans lesquels les temps de traitement des opérations sont très courts. Plus précisément, ces courts temps de traitement rendent les temps d'attente aux ressources négligeables par rapport aux temps de transport entre les ressources. Par conséquent, les files d'attente ne seront pas considérée dans cette approche (ou les agents considérerons des temps d'attente nuls).

Dans ce chapitre, nous donnerons dans un premier temps quelques généralités sur l'architecture proposée ainsi que sur les fonctions des agents. La résolution hétérarchique des conflits par négociation entre agents sera présentée. Cette négociation devra prendre en compte un maximum d'agents pour réduire au maximum le problème de myopie dans la résolution des conflits. Suite à cette négociation, le

problème de planification de trajectoire sera introduit où chaque agent élaborera sa propre trajectoire en fonction de ses voisins et de la prise de décision face aux conflits.

3.2 Architecture proposée et algorithme de navigation : généralités

Un algorithme de planification de trajectoire hétérarchique est proposé pour chaque agent afin d'effectuer l'opération de son produit. Pour chaque agent, l'objectif est de planifier une trajectoire sans collision vers une ressource qui doit être sélectionnée afin que l'opération soit achevée le plus tôt possible. Pour cela, les agents doivent prendre des décisions pour atteindre cet objectif en fonction du cahier des charges fourni. Les prises de décision se font aussi lors de la navigation, impliquant une ou plusieurs mises à jour des trajectoires des agents. Notons que pour cette approche, l'ordonnancement, qui permet de sélectionner la ressource, est appliqué à chaque mise à jour de trajectoire, permettant de changer de ressource selon les collisions à éviter vers chaque ressource. Cet évitement de collisions se fait par échange de trajectoire entre voisins à chaque mise à jour. Dans cette approche, la négociation joue un rôle essentiel dans la prise de décisions pour résoudre les conflits pouvant se produire lors de la navigation. Cette négociation est faite en comparant les performances d'un groupe connecté d'agents. Une vue d'ensemble de l'architecture est proposée dans la Fig. 3.1.

Notons que le niveau G.O.R.P ne communique avec les agents que lorsque ceux–ci sont aux ressources. Ce qui permet de leur fournir les informations nécessaires à l'opération qui devra être traitée. Ces informations incluent le cahier des charges comme mentionné dans le chapitre précédent. Ainsi, Les agents devront faire en sorte que ce cahier des charges soit respecté puisqu'ils n'auront plus la possibilité d'obtenir d'autres informations lors de la navigation.

L'algorithme de navigation, (encadré en vert sur la Fig. 3.1) ne traite qu'une opération à la fois et combine les fonctions suivantes :

- Le *planificateur de trajectoire avec ordonnancement* utilise les contraintes physiques et temporelles pour sélectionner la ressource optimale de manière à achever l'opération le plus tôt possible. Une trajectoire optimale sans collision est aussi générée pour chaque agent. Les contraintes temporelles permettent d'apparier la planification de trajectoire avec l'aspect d'ordonnancement.

- Le *suivi de trajectoire* est utilisé pour que l'agent suive d'une manière robuste la trajectoire planifiée en dépit des perturbations externes et des anomalies intrinsèques entre le robot et son modèle [Defoort et al., 2009b].

- Le *processus de négociation* permet de concevoir un mécanisme de priorité entre les agents connectés dans un même groupe. Un groupe connecté d'agents signifie que tous les agents du groupe communiquent d'une manière directe (avec les voisins) et indirecte (par le biais des voisins). Ceci est possible uniquement lorsque le nombre d'informations à transmettre est limité.

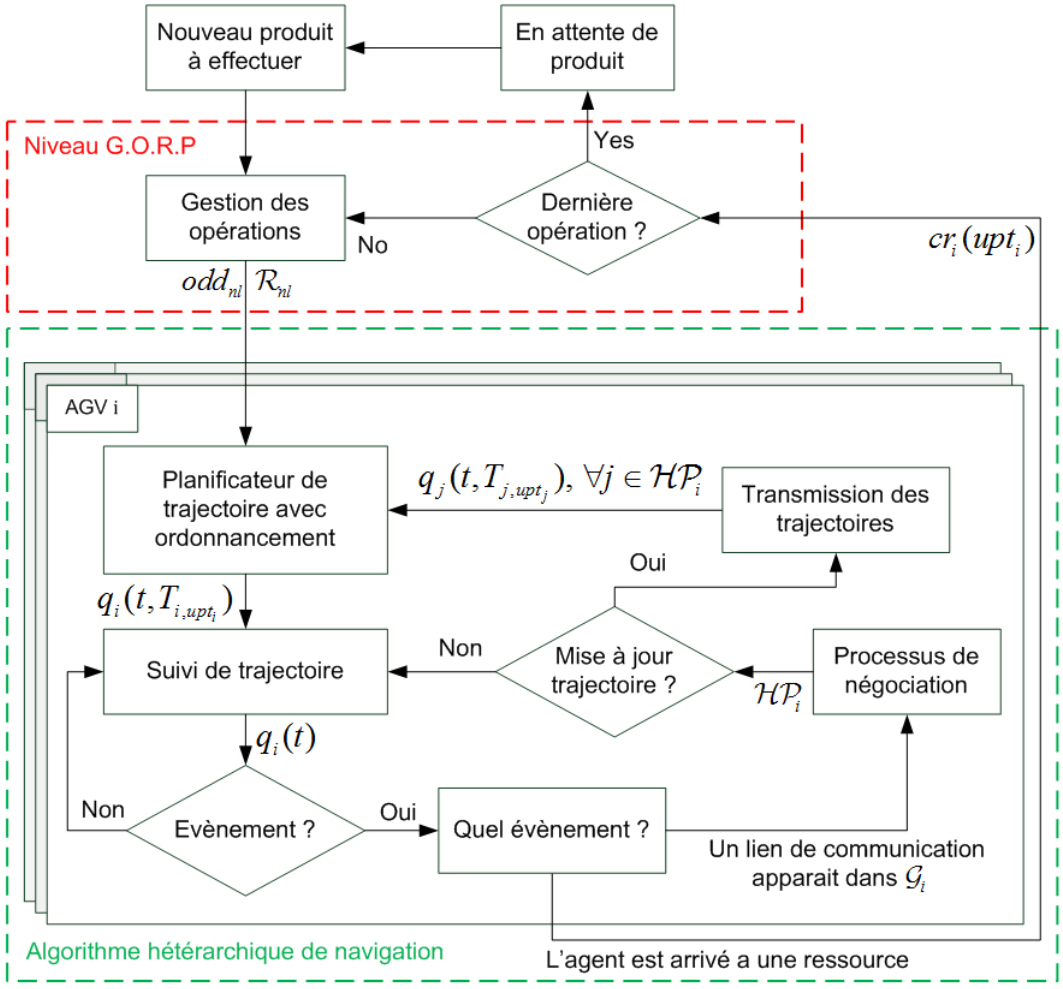


FIGURE 3.1 – Vue d'ensemble de l'architecture hétérarchique et de l'algorithme de navigation

- Le bloc *mise à jour trajectoire* applique le mécanisme de priorité pour savoir s'il doit mettre à jour sa trajectoire. Il y a deux cas pour lesquels il doit la mettre à jour :

  1. Lorsqu'il y a un nouvel agent voisin ayant un niveau de priorité plus haut.
  2. Lorsqu'un agent plus prioritaire a mis (ou doit mettre) à jour sa trajectoire.

  Sinon, l'agent peut continuer son suivi robuste de trajectoire.

- Le bloc *transmission des trajectoires* permet à l'agent i, lorsqu'il doit mettre à jour sa trajectoire suite à une décision du bloc *mise à jour trajectoire*, de recevoir les trajectoires $q_j(t, T_{j,upt_j})$ des agents voisins $j \in \mathcal{HP}_i$ ayant un niveau de priorité plus haut. Après avoir reçu ces trajectoires, l'agent i devra mettre à jour la sienne en évitant les collisions avec tous les agents $j \in \mathcal{HP}_i$.

- Les blocs *évènement* et *quel évènement* permettent de détecter et de réagir à deux types d'évènements. Tant qu'il n'y en a pas, l'agent suit sa trajectoire de manière robuste. Le premier

évènement est lorsque l'agent atteint une de ses ressources $b \in \mathcal{R}_{nl}$. Dans ce cas, il indique au niveau G.O.R.P son arrivée en spécifiant la ressource choisie $cr_i(upt_i)$ (i.e. la ressource choisie à la dernière mise à jour). Le second évènement dépend du groupe connecté $\mathcal{G}_i$ de l'agent i. En effet, si un nouveau lien de communication apparaît entre deux agents (ou plus), un (ou plusieurs) agent aura un niveau de priorité plus bas et devra mettre à jour sa trajectoire. Celui–ci ne sera pas forcément l'agent i, cela peut aussi être un de ses voisins. Dans ce cas, la décision de mise à jour de la trajectoire de l'agent i se fera par le bloc *mise à jour trajectoire* après négociation. Notons qu'il n'y a pas d'évènement si un lien de communication disparaît dans le groupe $\mathcal{G}_i$ car les collisions et les conflits ont été résolus avant que ce lien ne disparaisse.

Les autres blocs sont en relation avec la gestion de la production et ont peu d'impact sur la résolution de notre problème puisque l'algorithme de navigation ne traite qu'une opération à la fois. Nous nous focaliserons donc sur l'algorithme de navigation en lui-même. De plus, la fonction de suivi de trajectoire ne sera pas traitée car il relève du cadre mono–robot qui est hors contexte de cette thèse.

Pour l'algorithme de navigation que l'on vient de présenter, trois points importants sont à souligner :

1. Comme mentionné, la mise à jour de trajectoire de l'agent i se fait par évènement. Si aucun évènement ne se produit ou si les évènements rencontrés n'implique pas de mise à jour car aucun agent j n'est plus prioritaire (i.e. $\nexists j \in \mathcal{HP}_i$), l'agent i peut d'une manière directe atteindre sa ressource de destination. Ceci permet à un agent avec de mauvaises performances d'atteindre sa ressource sans se soucier de l'évitement de collisions inter–robots. Inversement, un agent avec de bonnes performances aura plus de temps pour éviter les collisions, il devra donc éviter les agents avec un niveau de priorité plus haut à chaque évènement.

2. Lorsqu'un agent j avec un niveau de priorité plus haut que l'agent i met à jour sa trajectoire, l'agent i doit attendre que l'agent j ait généré sa trajectoire avant de mettre à jour la sienne. En effet, il est difficile voire impossible de planifier des trajectoires de deux agents simultanément en présence de la fonction d'ordonnancement comme nous l'avons vu dans le chapitre 2.

3. Bien que la communication ne se fasse qu'uniquement avec les voisins, il est possible de passer des informations d'une manière indirecte par le biais de ceux–ci. Ainsi, chaque agent ne dispose pas uniquement de l'information locale des voisins, afin de détecter des évènements se produisant lors de la navigation ainsi que d'appliquer le processus de négociation entre agents connectés. Ceci est possible car les informations requises sont uniquement les indicateurs de performances $IP_i, i \in \mathcal{G}_i$, qui peuvent être rapidement transmis. Par contre, l'échange des trajectoires, qui nécessite plus de temps pour transmettre, se fait uniquement entre voisins.

Les généralités sur l'algorithme de navigation ayant été présentées, nous allons maintenant nous intéresser à la négociation entre agents et au problème de planification de trajectoire.

3.3 Négociation hétérarchique entre agents

Lors de la navigation, deux types de conflits peuvent se produire :

- Le premier conflit concerne le problème de collisions. Dans ce cas, les agents doivent décider quel agent peut suivre sa trajectoire planifiée et quel(s) autre(s) agent(s) doit(doivent) mettre à jour sa(leur) trajectoire pour éviter la ou les collisions.

- Le second conflit se produit lorsque plusieurs agents, étant voisins, ont l'intention de se diriger vers la même ressource. Par conséquent, ils doivent décider quel agent atteint la ressource en premier. D'une manière générale, ils doivent décider l'ordre d'arrivée à la ressource pour éviter tout interblocage.

Afin de résoudre ces conflits, des mécanismes de coopération ou une négociation deviennent obligatoires, spécialement dans un environnement contraint temporellement où des performances individuelles et globales sont demandées. Ces mécanismes permettent notamment de définir un ordre de priorité entre les agents voisins. Cet ordre de priorité sera utilisé pour résoudre les problèmes d'optimisation (du planificateur de trajectoires) d'une manière séquentielle. De ce fait, il est difficile, voire impossible, de donner une justification formelle de l'absence d'interblocage dû au couplage entre résolution de conflits et planification de trajectoire, qui sont tous deux des problèmes complexes [Reveliotis et Roszkowska, 2010].

L'approche de coopération considérée pour notre algorithme de navigation hétérarchique est basée sur une négociation par approche sociale [Rey et al., 2013]. D'autres approches, comme la négociation par réseau de pétri, ont été présentées dans le chapitre 1. Cependant, cette approche sociale a les avantages d'être simple, de pouvoir être mise en relation avec les performances des produits transportés et aussi d'être calculée et transmise très rapidement. Cependant, elle nécessite une adaptation face à notre problème comme nous le verrons plus bas. Avant de l'adapter, nous allons dans un premier temps présenter cette approche sociale dans le contexte auquel elle a été appliquée.

3.3.1 Négociation par approche sociale : généralités

La négociation par approche sociale a été introduite par [Rey et al., 2013] pour des systèmes manufacturiers guidés, c'est–à–dire lorsque les produits sont transportés à l'aide d'un système de convoi (navettes par exemple) où les temps de transport entre ressources sont fixes. De plus, comme les entités transportant les produits ne sont pas limitées par la portée de communication, les informations de tous les produits en cours de production sont connues.

Approximativement parlant, le concept de cette négociation permet de trouver de "bonnes" solutions pour le système global, qui sont aussi "bonnes" pour les objectifs individuels des agents. L'avantage de cette approche sociale est que les produits (agents) n'ont pas besoin de connaître les objectifs

individuels des autres ; juste un indicateur de performance leur donne des informations sur l'efficacité pour atteindre leur but.

Pour une architecture avec N_i agents (ou produits), qui communiquent tous les uns avec les autres, on définit le facteur social actuel d'un agent i comme suit :

$$sf_i = \frac{1 + \cos\left[\pi \cdot \left(1 - \frac{|IP_i - OP_i|}{\max(|IP_i|,|OP_i|)}\right)\right]}{2} \tag{3.3.1}$$

où IP_i et OP_i sont respectivement les performances individuelles et comparatives actuelles de l'agent i définies par :

$$IP_i = \frac{dd_i - ct_i}{dd_i} \tag{3.3.2}$$

$$OP_i = \frac{1}{N-1} \sum_{\substack{j=1 \\ j \neq i}}^{N} IP_j \tag{3.3.3}$$

La variable dd_i représente l'échéance du produit (et non de l'opération en cours). La variable ct_i est le temps de traitement actuel du produit, ce qui peut être caractérisé par la position (temporelle) du produit, en termes d'opérations, par rapport à son évolution dans la production. Comme définies dans [Rey et al., 2013], les performances sont appelées "bonnes" quand $IP_i < 0.3$, "mauvaises" lorsque $IP_i > 0.7$ et "moyennes" sinon.

Le facteur social, permettant de fixer l'ordre de priorité parmi les agents, a trois catégories de comportements selon sa valeur :

***Égoïste* :** Ce comportement, représenté par $sf_i < 0.25$ permet aux agents d'avoir un haut–niveau de priorité lorsqu'ils sont plus loin de leur objectif comparativement aux autres.

***Altruiste* :** Inversement au comportement égoïste, l'altruisme se traduit par $sf_i > 0.75$. Il permet aux agents proches de leur objectif, d'avoir un niveau de priorité plus bas que les autres.

***Coopératif* :** Lorsque le comportement d'un agent n'est ni égoïste, ni altruiste, il est dit coopératif.

Afin de fixer le mécanisme de priorité, un agent i a un niveau de priorité plus haut qu'un ou plusieurs autres agents j s'il a un facteur social plus bas (i.e. $sf_i < sf_j$).

Notons qu'il peut paraître étrange de parler d'altruisme pour des agents (ou robots). Ce terme est utilisé ici comme antonyme d'égoïste et permet de décrire un comportement qui permet à l'agent de prendre des décisions qui sont bénéfiques pour les objectifs des autres agents, même si cela n'a pas d'avantages apparents pour son propre objectif. Néanmoins, les comportements altruistes peuvent être problématiques pour les agents car ils peuvent perdre de vue leur(s) objectif(s) individuel(s). Pour pallier ce problème, le facteur social ne peut être plus grand qu'un seuil, noté $sf_{\max,i}$ afin d'empêcher

des comportements altruistes inattendus lorsque les performances individuelles sont mauvaises. Par conséquent, le facteur social devient, selon [Rey et al., 2013] :

$$sf_i = \min\left(\frac{1+\cos\left[\pi\cdot\left(1-\frac{|IP_i-OP_i|}{\max(|IP_i|,|OP_i|)}\right)\right]}{2},\ sf_{\max,i}\right) \tag{3.3.4}$$

avec le seuil $sf_{\max,i} = sf_i|_{OP_i=1}$. La Fig. 3.2 illustre l'évolution du facteur social en fonction des performances comparatives pour quatre performances individuelles fixées. Pour les performances fixées sur cette figure, plusieurs points sont à noter :

- Lorsque les performances individuelles sont bonnes (i.e. $IP = 0.17$), l'agent sera égoïste (i.e. a plus de chance d'avoir une priorité haute) si la majorité des agents a de bonnes performances (i.e. $OP < 0.3$). Sinon, lorsque la majorité des agents ont des performances moins bonnes voire mauvaises, il sera altruiste (si $OP > 0.5$) ou coopératif. Par conséquent, un agent avec de bonnes performances a plus de chance d'être altruiste, permettant ainsi aux autres agents d'avoir un niveau de priorité plus haut que lui.

- Si les performances sont moyennes (par exemple $IP = 0.54$ ou $IP = 0.38$), l'agent sera soit coopératif soit égoïste : coopératif lorsque la plupart des autres agents ont des bonnes ou mauvaises performances, égoïste si la plupart des agents ont des performances moyennes. Un agent avec des performances moyennes peut être égoïste s'il y a autant d'agents avec de mauvaises performances que d'agents avec de bonnes performances. Ceci est dû au fait que les performances comparatives sont définies comme une moyenne des performances individuelles des autres agents. Un agent avec des performances moyennes sera principalement coopératif. De plus, il ne peut être altruiste grâce au seuil $sf_{\max,i}$.

- Un agent avec de mauvaises performances (i.e. $IP = 0.82$) ne peut être qu'égoïste par rapport aux autres. Ceci vient principalement du seuil qu'on lui a imposé. Ainsi, si la plupart des agents ont de plus bonnes performances, son facteur social sera égal au seuil. L'agent en question sera encore plus égoïste si la plupart des autres agents sont aussi égoïstes.

En utilisant la même figure, un exemple est donné pour quatre agents 1, 2, 3 et 4 ayant respectivement les performances individuelles fixées à $IP_1 = 0.17$, $IP_2 = 0.38$, $IP_3 = 0.54$ et $IP_4 = 0.82$. Les performances comparatives, obtenus par l'équation (3.3.3), sont pour chaque agent $OP_1 = 0.58$, $OP_2 = 0.51$, $OP_3 = 0.46$ et $OP_4 = 0.36$. Par conséquent, les facteurs sociaux respectifs, donnés par l'équation (3.3.4), sont $sf_1 = 0.803$, $sf_2 = 0.152$, $sf_3 = 0.058$ et $sf_4 = 0.078$ et permettent de définir l'ordre de priorité. On peut voir que l'agent 4, ayant les plus mauvaises performances, n'a pas le niveau de priorité le plus haut car $sf_4 > sf_3$. Ceci est normal puisque les performances globales (moyenne des performances individuelles) sont moyennes, impliquant un avantage aux agents ayant déjà des

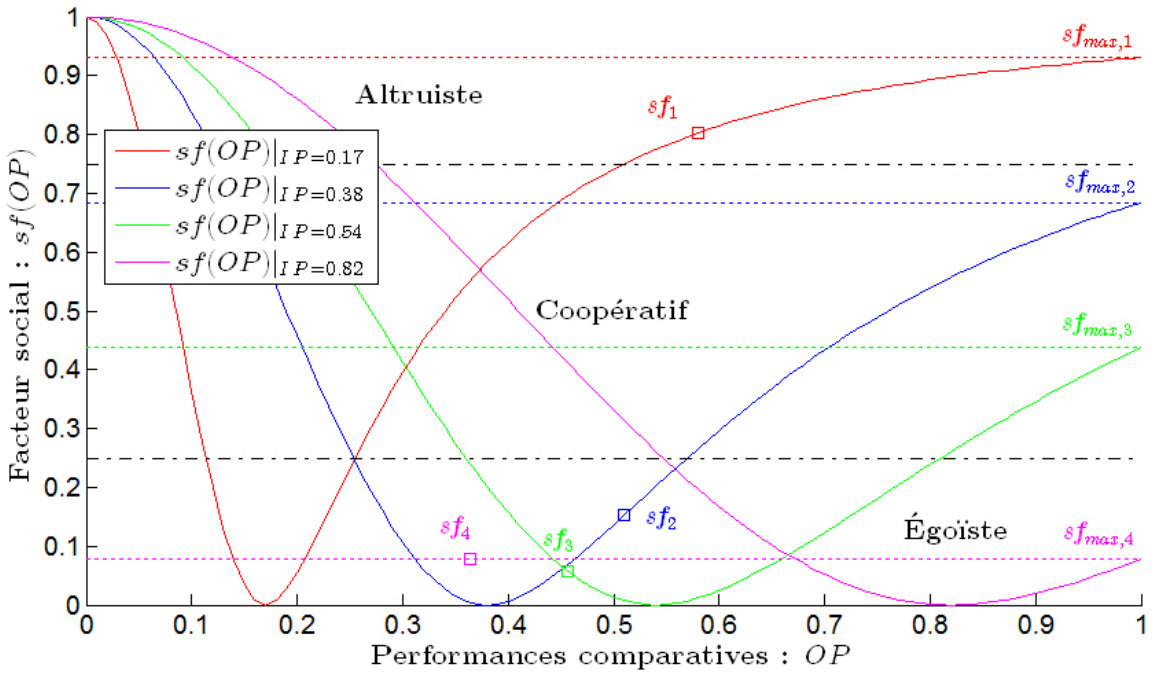


FIGURE 3.2 – Évolution du facteur social en fonction des performances comparatives

performances individuelles moyennes tels que les agents 2 et 3 qui sont égoïstes dans cet exemple. L'agent 1 est altruiste car les performances des autres agents sont plus mauvaises.

On en vient à se demander si le fait que l'agent 4 n'ait pas le niveau de priorité le plus haut pose problème. Pour la problématique sur laquelle ce facteur social a été appliquée initialement [Rey et al., 2013], ceci ne pose pas problème car l'objectif est une production juste à temps où l'on peut se permettre de retarder le traitement d'un produit jusqu'à un certain point. Pour notre problème, il serait préférable qu'un agent avec de mauvaises performances ait plus de chance d'être le plus prioritaire. Par conséquent, il est nécessaire d'adapter le facteur social pour favoriser l'objectif individuel des agents.

3.3.2 Négociation par approche sociale : adaptation à notre problème

Afin d'adapter la négociation par approche sociale à notre problème, différentes modifications sont à prévoir. D'abord, la formulation des performances individuelles doit être changée afin de considérer une seule opération et son échéance et la position de l'agent dans l'environnement de production. Ensuite, les performances comparatives doivent intégrer la portée limitée de communication tout en considérant un maximum d'agents. Enfin, le facteur social doit faire en sorte que les agents avec de mauvaises performances aient plus de chances d'avoir un niveau de priorité plus haut.

Nouvelle formulation des performances individuelles

Les performances individuelles (3.3.2) présentées précédemment ne sont pas adaptée à notre problème car le temps de traitement ct_i et l'échéance dd_i dépendent du produit et non de l'opération en elle-même. De ce fait, il faut formuler un nouvel indice de performances individuelles IP_i tout en gardant les spécificités de base de cet indicateur. C'est–à–dire qu'il faut que les appellations de bonnes et mauvaises performances restent cohérentes et que $IP_i \in [0, 1]$. Afin de faire correspondre les performances individuelles avec les appellations données, il est utile de savoir quand les performances sont bonnes ou mauvaises pour notre problème. Ici, l'objectif de chaque agent est d'achever l'opération le plus tôt possible, qui est équivalent à faire en sorte que l'écart entre le temps pour achever l'opération ct_{nlb} et l'échéance odd_{nl} soit le plus grand possible. Ainsi, lorsque l'écart est grand, les performances seront considérées comme bonnes. Si l'écart est petit, les performances seront mauvaises puisque l'opération sera achevée à une date proche de l'échéance.

Notons que le temps pour compléter une opération $ct_{nlb} = T_{i,upt_i} + TT_{ib} + w_b + opt_{nl}$ dépend de T_{i,upt_i}, qui représente l'instant où la trajectoire (et donc l'indice de performance IP_i) est mise à jour. De plus, notons que la ressource b utilisée pour le calcul de l'indicateur de performance est celle choisie lors du dernier ordonnancement. L'idée est ici de formuler les performances individuelles à l'instant T_{i,upt_i} afin de mettre en oeuvre le mécanisme de priorité. Ceci permettra de résoudre les différents conflits pour chaque agent, qui pourra ensuite calculer sa trajectoire et mettre à jour sa ressource pour cet instant T_{i,upt_i}. Par conséquent, il faut aussi que le nouvel indice de performance prenne en compte la position de l'agent afin de faire le lien entre la navigation et les spécificités du produit transporté. Les nouvelles performances individuelles sont définies par l'équation (3.3.5) pour un agent i transportant un produit l pour l'opération n vers la ressource b :

$$IP_i = \frac{\frac{\|q_i - q_b\|}{v_{max}}}{odd_{nl} - opt_{nl} - w_k - T_{i,upt_i}},\ b \in \mathcal{R}_{nl} \tag{3.3.5}$$

où $\frac{\|q_i - q_b\|}{v_{max}}$ représente une sous estimation du temps de transport TT_{ib}. Cette estimation a été choisie pour deux raisons. D'une part, elle permet de considérer tous les agents de la même manière, c'est–à–dire que tous les agents sont considérés comme s'ils pouvaient atteindre la ressource d'une manière directe (i.e. sans collision à éviter). D'autre part, elle permet d'éviter les répercussions. En effet, si le temps de transport planifié à la dernière mise à jour était choisi, les performances individuelles deviendraient dépendantes de la contrainte (2.3.2). Ainsi, si plusieurs agents se dirigent vers la même ressource, les performances des agents deviendraient dépendantes les unes des autres à chaque mise à jour.

Prenons par exemple deux agents i et j se dirigeant vers une même ressource b et ayant des performances similaires à un instant T_{i,upt_1}. Lorsque le mécanisme de priorité est appliqué, l'agent i doit arriver après l'agent j à la ressource dû à la contrainte (2.3.2). De ce fait, à la mise à jour suivante T_{i,upt_2}, les performances de l'agent i pourraient devenir plus mauvaises que celles de l'agent j. Il y

aurait une répercussion puisque l'agent i doit arriver après l'agent j à la ressource à l'instant T_{i,upt_1} puis l'agent j devra arriver plus tard que l'agent i à l'instant T_{i,upt_2}. Il y aurait donc un changement successif de priorité qui impliquerait que ces deux agents se retardent mutuellement, jusqu'à ce qu'ils ne le puissent plus à cause de la contrainte sur l'échéance. De ce fait, cela empêcherait de trouver une trajectoire faisable (car une des deux contraintes d'échéance pour la ressource, ne serait pas respectée). De ces propos, on remarque l'intérêt de l'estimation proposée du temps de transport.

Afin de mieux comprendre le choix de cette formalisation des performances individuelles (3.3.5), il faut dans un premier temps mentionner sa provenance. Cette formalisation est basée sur la contrainte d'échéance (2.3.1) en utilisant la définition du temps pour achever l'opération $ct_{nlb} = T_{i,upt_i} + TT_{ib} + w_b + opt_{nl}$. En effet, en considérant $IP_i \leq 1$ où $IP_i = 1$ représente la plus mauvaise performance, on obtient l'inégalité suivante :

$$\frac{\frac{\|q_i - q_b\|}{v_{max}}}{odd_{nl} - opt_{nl} - w_k - T_{i,upt_i}} \leq 1$$

ce qui implique, en considérant $\|q_i - q_b\|/v_{max} = TT_{ib}$:

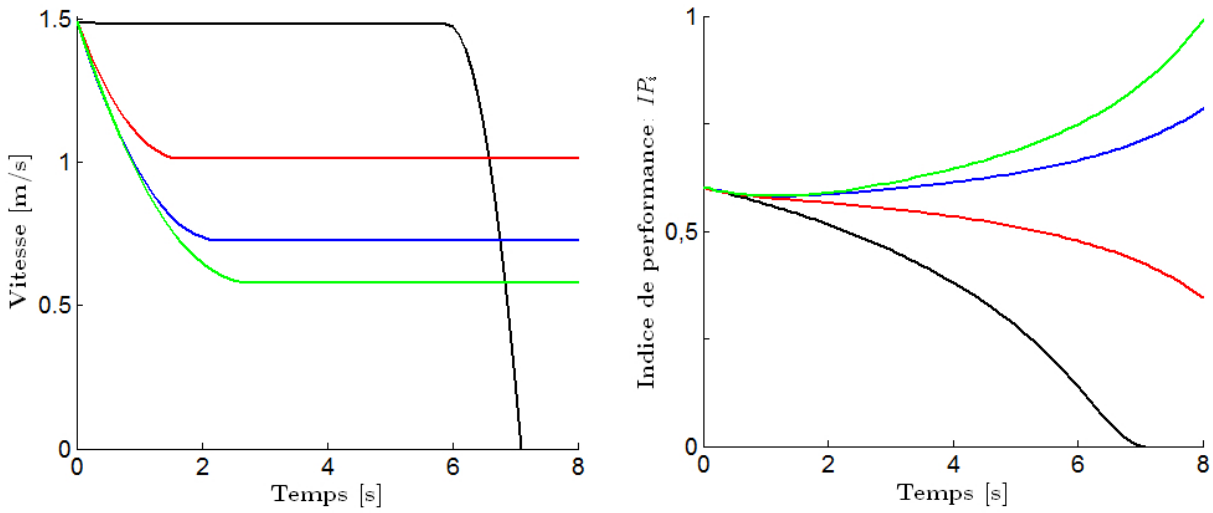
$$TT_{ib} \leq odd_{nl} - opt_{nl} - w_k - T_{i,upt_i}$$

On remarque donc bien la base de la formulation des performances individuelles à partir de la contrainte d'échéance où les appellations de bonnes et mauvaises performances restent cohérentes. En effet, si l'opération est achevée exactement à l'échéance, l'indice se rapprochera de $IP_i = 1$ et les performances seront considérées comme mauvaises. Comme mentionné préalablement, les performances sont bonnes lorsque l'on se rapproche de l'objectif, c'est–à–dire lorsque IP_i se rapproche de zéro. Ceci est aussi valable pour la nouvelle formulation car la notion de distance définie par l'estimation du temps de transport, permet d'avoir de meilleures performances lorsque la distance par rapport à la ressource se réduit, ce qui correspond bien à notre objectif.

Remarque 3.1 *Notons que pour la formulation des performances individuelles, l'instant de mise à jour T_{i,upt_i}, le temps d'attente à la ressource w_b et le temps de traitement de l'opération opt_{nl} sont retranchés à l'échéance odd_{nl}. Le temps de transport estimé $\|q_i - q_b\|/v_{max}$ est au numérateur. Pour le dénominateur, $odd_{nl} - w_b - opt_{nl}$ représente l'échéance d'arrivée à la ressource, c'est–à–dire le temps pour lequel l'agent doit arriver à la ressource b qui assure que l'opération n soit achevée avant l'échéance odd_{nl}. De plus, le temps T_{i,upt_i}, représentant le moment où les performances sont calculées, est aussi retranché afin d'être en accord avec le numérateur.*

Il est maintenant nécessaire de mentionner les deux raisons pour lesquelles les performances peuvent devenir mauvaises lors de la navigation. La première raison est l'arrivée d'un nouvel agent à la ressource b, qui se traduit par une augmentation du temps d'attente w_b. De ce fait, le dénominateur de IP_i devient plus petit et les performances deviennent plus mauvaises puisque le traitement de l'opération n à la ressource b est retardé.

La seconde raison est lorsqu'un agent doit éviter une ou plusieurs collisions. En effet, lorsqu'un agent entreprend une manoeuvre d'évitement, sa vitesse sera réduite et il mettra plus de temps pour atteindre la ressource. Pour illustrer cette perte de performances, provenant d'une réduction de la vitesse de l'agent, un exemple est proposé sur la Fig. 3.3.



(a) (b)

FIGURE 3.3 – Illustration de la perte de performances due à la réduction de la vitesse : (a) profils de vitesse d'un agent, (b) évolution respective des performances individuelles

Pour cet exemple, la vitesse maximale de l'agent est $v_{max} = 1.5m/s$. Lorsque l'agent n'a pas besoin d'éviter de collision, sa vitesse reste proche de cette vitesse maximale (profil de vitesse de couleur noire, Fig. 3.3–(a)) et ses performances individuelles IP_i convergent rapidement vers 0 (cf. Fig. 3.3–(b)). Si cet agent doit un peu réduire sa vitesse (profil de vitesse rouge, Fig. 3.3–(a)), ses performances convergeront plus lentement vers 0 (cf. Fig. 3.3–(b)). Cependant, lorsque cet agent a besoin de réduire sa vitesse d'une manière plus importante comme le montre les profils de vitesse en bleu et en vert sur la Fig. 3.3–(a), les performances individuelles vont devenir de plus en plus mauvaises (cf. Fig. 3.3–(b)). Ceci est dû au fait que le numérateur $\|q_i - q_b\|/v_{max}$ va décroître moins rapidement que le dénominateur $odd_{nl} - w_b - opt_{nl} - T_{i,upt_i}$. Par conséquent, un agent peut subir une perte de performances s'il doit ralentir pour éviter un ou plusieurs autres agents.

Remarque 3.2 *On remarque que lorsque l'agent est proche de la ressource, les performances individuelles se rapprochent de* 0 *si l'agent n'a pas trop réduit sa vitesse lors de la navigation (cf. Fig. 3.3–(b), courbe noire). Ceci a un gros avantage pour l'évitement de collisions aux alentours de la ressource car les agents démarrant de cette ressource ont plus de chance d'avoir de moins bonnes performances que l'agent qui arrive à celle–ci. Par conséquent, les agents sortant d'une ressource auront plus de chance d'avoir un niveau de priorité plus haut que les agents y arrivant.*

Modification des performances comparatives

Comparé au problème traité dans [Rey et al., 2013], les agents ne communiquent pas tous ensemble dans cette approche. Il est donc nécessaire de considérer cette limitation dans le calcul des performances comparatives (3.3.3), qui sont transformées comme suit :

$$OP_i(t) = \frac{1}{N_g - 1} \sum_{\substack{j \in \mathcal{G}_i \\ j \neq i}} IP_j \tag{3.3.6}$$

où $N_g = card(\mathcal{G}_i)$ et $\mathcal{G}_i$ est l'ensemble des agents connectés à l'agent i défini par :

$$\mathcal{G}_i = \{j \in \mathcal{A}, \exists j_1, \dots j_k, (i, j_1) \in \mathcal{E}, \dots (j_k, j) \in \mathcal{E}\} \tag{3.3.7}$$

Cette notion de groupe est illustrée sur la Fig. 3.4 où l'on remarque que tous les agents connectés à i d'une manière directe (i_1, i_2 et j) et indirecte (j_1, j_2 et j_3 via j) appartiennent au groupe $\mathcal{G}_i$. L'agent k, n'étant connecté à aucun de ces agents, n'est pas dans ce même groupe.

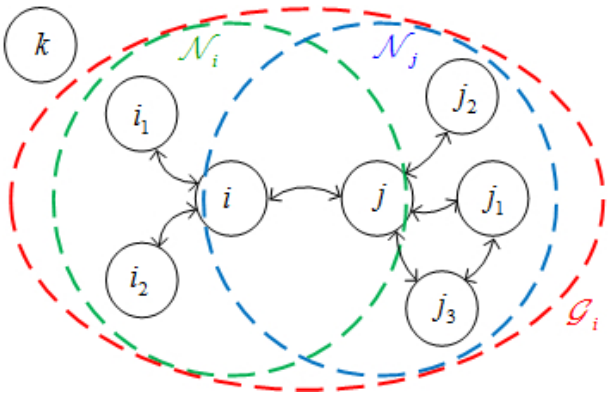


FIGURE 3.4 – Illustration d'un groupe connecté d'agents

Le choix d'appliquer le facteur sur un groupe connecté d'agents (plutôt que sur l'ensemble des voisins) est dû à deux raisons. D'une part, le facteur social doit considérer un maximum d'agents pour améliorer la performance globale d'un groupe en réduisant ainsi la myopie sociale. Ne pouvant pas considérer l'ensemble des agents à cause de la portée de communication limitée, ajouter les communications indirectes (par le biais des voisins) permet d'augmenter le nombre d'agents pour le calcul du facteur social. D'autre part, les agents doivent être sur le même niveau de comparaison, c'est–à–dire qu'ils doivent se comparer avec les mêmes agents afin que l'application du facteur sociale soit cohérente (voir Fig. 3.4 où les voisins des agents i et j sont différents, ce qui est généralement le cas). Cette notion de groupe est applicable car uniquement les performances individuelles sont requises pour le calcul du facteur social et pourront donc être transmises rapidement.

Amplification du facteur social

La dernière modification de l'approche sociale concerne le calcul du facteur social (3.3.4), dont le problème a été évoqué préalablement. En effet, il est possible qu'un agent avec les plus mauvaises performances n'ait pas la plus haute priorité. Pour cela, il serait préférable de renforcer l'aspect d'objectif individuel de l'agent. Le facteur social étant de valeur inférieure à 1, une fonction puissance pourrait permettre d'amplifier le facteur social. Cependant, une puissance fixe est inutile puisqu'elle agirait de la même façon quelques soient les performances individuelles. Par conséquent, il est préférable d'amplifier le facteur social à l'aide d'une fonction puissance qui dépend des performances individuelles IP_i pour le facteur social associé à chaque agent i. Pour cela, on définit, à partir de l'équation (3.3.4), le facteur social amplifié psf_i de la manière suivante :

$$psf_i = (sf_i)^{g(IP_i)} \tag{3.3.8}$$

avec

$$g(IP_i) = e^{a\,(\frac{(IP_i-1)}{2}+b)} + c \tag{3.3.9}$$

où a, b et c sont des constantes telles que la fonction g soit suffisamment croissante, continue et toujours supérieure à 1. La variable c permet de faire une translation sur l'axe des ordonnées afin que $g(0) = 1$. Les constantes a et b sont réglées de telle sorte qu'il y ait une différence suffisante entre bonnes et mauvaises performances dans l'amplification du facteur social. La croissance de cette fonction donne plus d'importance aux comportements égoïstes lorsque les performances sont mauvaises. C'est principalement pour cette raison que la fonction exponentielle a été choisie car elle correspond parfaitement aux caractéristiques nécessaires pour la fonction g.

Afin d'illustrer l'amplification appliquée au facteur social, la Fig. 3.5 est proposée où les constantes de la fonction g sont $a = 0.25$, $b = 2.72$ et $c = -0.493$. Les performances individuelles fixées pour cette figure sont les mêmes que sur la Fig. 3.2.

On remarque sur cette figure que l'amplification permet bien de favoriser les comportements égoïstes, surtout lorsque les performances sont mauvaises puisque le seuil devient beaucoup plus petit (voir $sf_{\max,4}$). En utilisant le même exemple que précédemment (avec les 4 agents), on obtient les facteurs sociaux amplifiés suivants : $psf_1 = 0.72$, $psf_2 = 0.04$, $psf_3 = 0.004$, $psf_4 = 0.002$. Ceci montre bien l'effet de la dépendance de l'amplification sur les performances individuelles et surtout que l'agent 4 dont les performances sont les plus mauvaises a la plus haute priorité grâce à l'amplification. De plus, notons qu'en amplifiant le facteur social défini par (3.3.4), les seuils $sf_{max,i}$ sont aussi amplifiés et deviennent $psf_{max,i}$ comme le montre la Fig. 3.5. L'amplification des seuils a une grande importance car elle permet d'avoir des facteurs sociaux très bas lorsque les performances deviennent mauvaises, ce qui est très bien illustré par cette même figure à l'aide de l'évolution de psf_4.

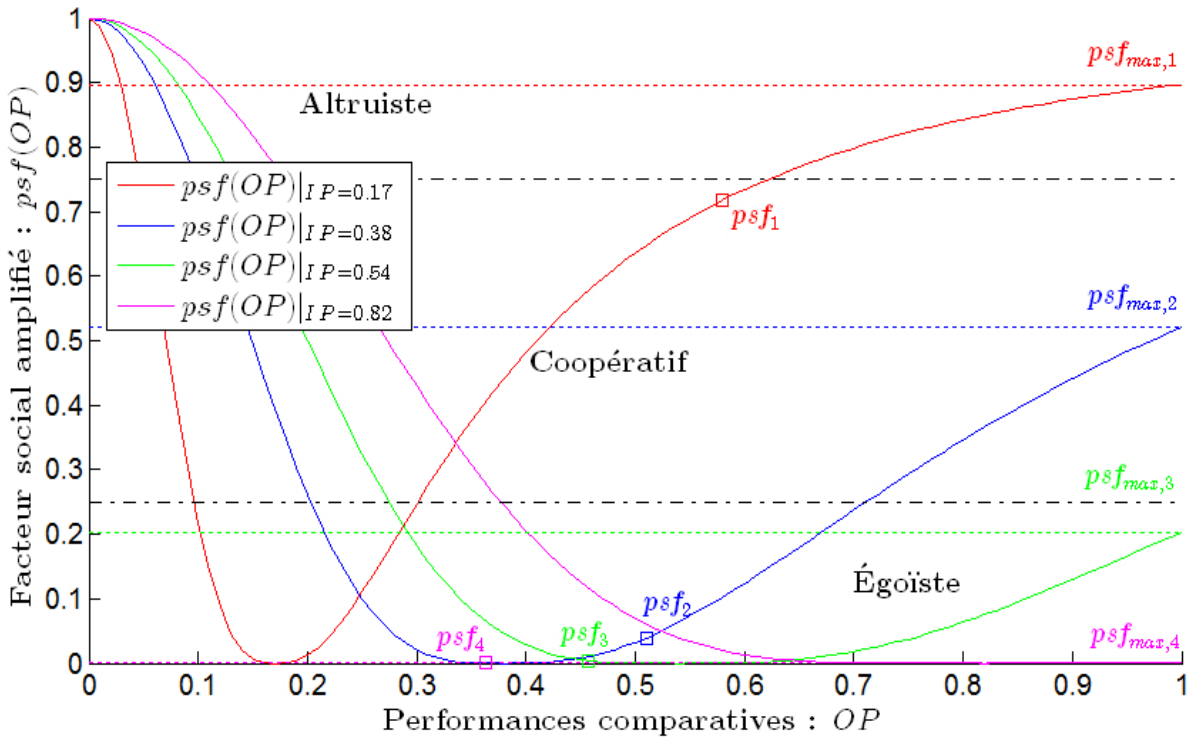


FIGURE 3.5 – Évolution du facteur social amplifié en fonction des performances comparatives

Pour finir, le facteur social ne peut être appliqué pour des groupes de deux agents (car leurs facteurs sociaux seraient identiques). Dans ce cas, la négociation se fera à l'aide des performances individuelles où l'agent dont les performances sont les plus mauvaises aura la plus haute priorité.

3.4 Planificateur de trajectoire avec ordonnancement

Considérons un ensemble d'agents $i \in \mathcal{A}$, le problème le planification de trajectoire consiste à calculer une trajectoire optimale sans collision vers une ressource à sélectionner qui peut être mise à jour lors de la navigation.

Lorsqu'il doit mettre à jour sa trajectoire à l'instant T_{i,upt_i}, chaque agent i doit, d'une manière séquentielle, calculer une trajectoire sans collision $q(t, T_{i,upt_i})$ d'une configuration actuelle q_i vers une ressource appropriée q_b, $b \in \mathcal{R}_{nl}$ pour le produit l qu'il transporte afin d'effectuer l'opération n. Notons que le premier argument t de la trajectoire $q(t, T_{i,upt_i})$ représente le temps et le second argument T_{i,upt_i} représente l'instant où la trajectoire est mise à jour. De plus, la première génération de trajectoire se fait à l'instant $T_{i,upt_i} = T_{i,init}$ lorsque l'agent i démarre de la ressource ayant effectuée l'opération précédente $n-1$. Chaque agent i n'a que les informations relatives à ses voisins $j \in \mathcal{N}_i$, ce qui est un élément clé de cette approche puisque la complexité des calculs est réduite.

Considérons un agent i devant prendre des décisions afin d'achever l'opération $n \in \mathcal{O}_l$ du produit transporté $l \in \mathcal{J}$ en respectant le cahier des charges fourni. L'ensemble actuel $\mathcal{HP}_i \subset \mathcal{N}_i$ caractérise tous les agents voisins ayant un niveau de priorité plus haut que i :

$$\mathcal{HP}_i = \{j \in \mathcal{N}_i, sf_j \leq sf_i\} \tag{3.4.1}$$

Définissons aussi le sous–ensemble actuel $\mathcal{HPD}_i \subset \mathcal{HP}_i$ des agents ayant une priorité plus haute que i et se dirigeant vers la même ressource que celui–ci. Afin de déterminer la trajectoire sans collision qui permet d'achever l'opération en cours n du produit n, les trois étapes suivantes sont à suivre :

1. Une trajectoire sans collision, notée $q_{ib}(t, T_{i,upt_i})$ est planifiée vers chaque ressource $b \in \mathcal{R}_{nl}$. Pour chaque ressource b, on considère le problème d'optimisation suivant qui consiste à déterminer la trajectoire sans collisions $q_{ib}(t, T_{i,upt_i})$ ainsi que le temps de transport TT_{ib} pour un agent i sur l'intervalle $[T_{i,upt_i}, T_{i,upt_i} + TT_{ib}]$. La trajectoire doit permettre d'arriver le plut tôt possible, c'est–à–dire en minimisant le critère suivant :

$$\min_{q_{ib}(t,T_{i,upt_i}),\, T_{i,fin}} \quad TT_{ib} = T_{i,fin} - T_{i,upt_i} \tag{3.4.2}$$

   assujetti aux contraintes de production suivantes :

   - **Contrainte liée à la production :** Cette contrainte temporelle est définie par le cahier des charges fourni par le niveau G.O.R.P. Pour un agent i prenant des décisions afin d'achever l'opération $n \in \mathcal{O}_l$ du produit transporté $l \in \mathcal{J}$, l'inégalité suivante doit être respectée :

$$ct_{nlb} = T_{i,upt_i} + TT_{ib} + opt_{nl} \leq odd_{nl}, \qquad \forall n \in \mathcal{O}_l, \forall l \in \mathcal{J}, \forall b \in \mathcal{R}_{nl} \tag{3.4.3}$$

   - **Contrainte d'arrivée à la ressource :** La contrainte suivante, mise en place suite au processus de négociation, assure que l'agent i arrivera plus tard que les autre agents avec une plus haute priorité à la ressource spécifiée :

$$T_{i,upt_i} + TT_{ib} \geq \max_{j \in \mathcal{HPD}_i} T_{j,fin} \tag{3.4.4}$$

     Cette contrainte assure un choix approprié de ressource. Une inégalité stricte n'est pas nécessaire car la contrainte (3.4.7) assure l'évitement de collision autour de la ressource. Ainsi l'agent i sera retardé en se dirigeant vers la ressource b afin d'éviter une collision avec d'autres agents arrivant approximativement au même moment.

   ainsi qu'aux contraintes liées à l'agent suivantes :

   - **Conditions aux limites :** L'agent commence la mise à jour à l'instant actuel $T = T_{i,upt_i}$ de la position actuelle q_i à la vitesse actuelle v_i pour arriver à la position finale $q_b, b \in \mathcal{R}_{nl}$ à

l'instant final $T_{i,upt_i} + TT_{ib}$ à vitesse nulle, se traduisant par :

$$\begin{aligned} q_i(T_{i,upt_i}, T_{i,upt_i}) &= q_i \\ \dot{q}_i(T_{i,upt_i}, T_{i,upt_i}) &= v_i \\ q_i(T_{i,upt_i} + TT_{ib}, T_{i,upt_i}) &= q_b, \qquad b \in \mathcal{R}_{nl} \\ \dot{q}_i(T_{i,upt_i} + TT_{ib}, T_{i,upt_i}) &= 0 \end{aligned} \tag{3.4.5}$$

Notons que la vitesse actuelle v_i n'est pas nécessairement nulle (surtout lorsque la trajectoire est mise à jour lors de la navigation). Cependant, la vitesse actuelle est nulle lorsque l'agent démarre d'une ressource, c'est–à–dire à la première génération de trajectoire à l'instant actuel $T = T_{i,init}$.

- **Limitation du comportement physique de l'agent :** La vitesse de l'agent est bornée comme suit :

$$\|\dot{q}_i(t, T_{i,upt_i})\| \leq v_{max}, \qquad \forall t \in [T_{i,upt_i}, T_{i,upt_i} + TT_{ib}] \tag{3.4.6}$$

où v_{max} est la vitesse maximale pour chaque agent (puisque tous les agents sont identiques).

- **Contrainte d'évitement de collisions :** Puisque l'environnement manufacturier est considéré sans obstacle stationnaire, l'agent doit uniquement éviter les collisions avec ses voisins ayant un niveau de priorité plus haut.

$$\|q_i(t, T_{i,upt_i}) - q_j(t, T_{j,upt_j})\| \geq d_{safe}, \qquad \forall j \in \mathcal{HP}_i,\ \forall t \in [T_{i,upt_i}, T_{i,upt_i} + TT_{ib}] \tag{3.4.7}$$

Notons que chaque agent $j \in \mathcal{HP}_i$ à éviter n'a pas forcément mis à jour sa trajectoire au même instant, i.e. $T_{j,upt_j} \leq T_{i,upt_i}$. Ces instants sont égaux s'il existe au moins un agent plus prioritaire que l'agent j qui met à jour sa trajectoire au même instant, i.e. $T_{j,upt_j} = T_{i,upt_i}$ si $\mathcal{HP}_j \neq \varnothing$ et $j \in \mathcal{HP}_i$.

2. Lorsque toutes les trajectoires (i.e. une vers chaque ressource) sont planifiées, la ressource $cr_i(upt_i)$ est choisie pour l'instant de mise à jour T_{i,upt_i} en fonction du temps pour achever l'opération de chaque ressource pouvant effectuer celle–ci, i.e. $cr_i(upt_i) = \arg\min_b ct_{nlb}$. Le temps pour achever cette opération ct_{nlb} est ici défini par $ct_{nlb} = T_{i,upt_i} + TT_{ib} + opt_{nl}$ où T_{i,upt_i} est l'instant où l'agent i met à jour sa trajectoire, TT_{ib} est le temps de transport pour chaque ressource b et opt_{nl} le temps de traitement de l'opération n.

3. La ressource étant choisie, la trajectoire finale devient $q_i(t, T_{i,upt_i}) = q_{i(cr_i(upt_i))}(t, T_{i,upt_i})$ et le temps final $T_{i,fin} = T_{i,init} + TT_{i(cr_i(upt_i))}$.

Notons que les étapes 1 et 2 sont étroitement liées. La première permet de trouver la meilleure trajectoire sans collision vers chaque ressource, c'est–à–dire la trajectoire ayant un temps de transport le plus court. La seconde étape permet d'utiliser les temps de transport afin de déterminer le choix

de la ressource en considérant que chaque trajectoire planifiée est la meilleure en termes de temps de transport. Par conséquent, la meilleure trajectoire parmi celles générées vers chaque ressource est celle qui permet d'achever l'opération le plus tôt.

Pour résoudre les problèmes d'optimisation vers chaque ressource, les trajectoires sont mises sous forme paramétrique en utilisant les courbes splines définies dans le chapitre 2. Ensuite, une programmation dynamique séquentielle est adoptée. La programmation séquentielle quadratique est une technique itérative permettant de résoudre des problèmes d'optimisation non linéaires. La programmation linéaire quadratique résout une séquence de sous–problèmes d'optimisation, où chacun de ceux–ci optimise un modèle quadratique de l'objectif assujetti à une linéarisation des contraintes [Boggs et Tolle, 1995].

3.5 Conclusion

Dans ce chapitre, nous avons présenté les éléments essentiels permettant aux agents d'effectuer une opération du produit qu'ils transportent. L'approche présentée est basée sur un algorithme de navigation hétérarchique où chaque agent doit résoudre les conflits par lui–même, planifier une trajectoire sans collision vers une ressource spécifique, et choisir la ressource permettant d'achever l'opération de son produit le plus tôt possible.

Afin de résoudre les conflits, une négociation par approche sociale a été présentée et adaptée à notre problème. Elle permet de comparer les performances entre les différents agents appartenant au même groupe connecté. Son adaptation a permis de considérer trois points importants : la prise en compte de la position des agents dans la formulation des performances individuelles, la portée limitée de communication à intégrer dans les performances comparatives où un maximum d'agents doit être pris en compte et le renforcement des comportements égoïstes en présence de performances individuelles mauvaises dans le calcul du facteur social.

La combinaison entre la planification de trajectoire et l'ordonnancement a été présentée comme une planification vers chaque ressource permettant d'estimer les temps d'arrivée afin de pouvoir, par la suite, sélectionner la ressource qui minimise le temps pour achever l'opération. La trajectoire finale planifiée est bien entendu celle permettant d'atteindre la meilleure ressource.

Les principaux résultats seront donnés dans le chapitre 6 avec l'analyse correspondante. Les avantages de cette approche ainsi que ses inconvénients y seront illustrés. Cependant, le nombre d'agents que l'on peut déployer dans le système de production est limité pour deux raisons. D'une part, si l'on considère un grand nombre d'agents, la myopie sera plus élevée, impliquant de grand conflits à résoudre qui ne peuvent pas être prévus. D'autre part, un grand nombre d'agents impliquerait plus d'évènements et donc, plus de conflits à résoudre. De ces faits, beaucoup de répercussions pourraient se produire, réduisant les performances globales du système de production. L'approche considérée dans le prochain chapitre sera conçue afin de réduire l'impact de ces inconvénients.

Chapitre 4

Planification de trajectoire : architecture supervisée

4.1 Introduction

Dans le chapitre 1, nous avons vu que les architectures hétérarchiques permettent une haute réactivité mais ont de moins bonnes performances à cause du phénomène de myopie. Notons que les performances dépendent avant tout de l'objectif. Ainsi, des performances moins bonnes signifient que l'on s'éloigne de l'objectif désiré. L'approche présentée dans le chapitre 3 peut être améliorée afin d'une part, obtenir de meilleures performances en termes de temps pour achever l'opération et d'autre part, réduire la myopie sociale provenant du manque d'informations des agents qui ne communiquent qu'avec leurs voisins. Pour cela, l'utilisation d'un superviseur permettrait de se rapprocher d'une architecture hybride où les deux points précédents seront améliorés. Il sera donc supposé que des communications entre les agents et le niveaux G.O.R.P ainsi que des communications entre les agents et le superviseur sont possibles lors de la navigation. De plus, les agents pourront aussi être informés sur les files d'attente des ressources qu'ils convoitent.

Dans ce chapitre, nous introduirons ce superviseur permettant d'éviter les inconvénients d'une architecture hétérarchique. Notons dans un premier temps que les fonctions de planification de trajectoire et d'ordonnancement restent combinées. Au vu des remarques sur les impacts de la mutualisation (chapitre 1), ces deux fonctions doivent rester incluses dans l'algorithme de navigation au vu de la difficulté pour les calculer par un superviseur. Par conséquent, le rôle du superviseur sera d'assister les agents dans leurs tâches. Pour cela, le superviseur devra mettre en oeuvre la stratégie de priorité. Cette stratégie sera plus optimale puisque le superviseur considère la totalité des agents. Il aura donc les rôles suivants :

- Il devra permettre aux agents de savoir l'ordre d'arrivée à une ressource lorsqu'ils vont vers la même destination.

- Il autorisera les agent à ordonnancer leur produit à certaines mise à jour, c'est–à–dire leur permettre de choisir la ressource.

- Il permettra de détecter les conflits de collisions auxquels les agents risquent de se confronter.

En incluant un superviseur, les agents peuvent prévoir les conflits sans avoir à communiquer avec l'ensemble de la flottille. Cependant, cela nécessite des changements dans l'algorithme de navigation où le problème sera résolu différemment. Notons que l'objectif des agents reste le même, c'est–à–dire achever l'opération le plus tôt possible. De plus, contrairement à l'approche du chapitre 3, la mise à jour des trajectoires ne se fait plus en fonction des évènements qui se produisent. Ici, tous les agents mettent à jour leur trajectoire au même instant et d'une manière itérative. Pour cela, une planification de trajectoire en deux étapes est mise en oeuvre. A chaque itération, les deux étapes suivantes sont appliquées :

- La première étape du planificateur de trajectoire, considérée comme globale, permet d'éviter les zones de conflits définies par le superviseur. De plus, l'ordonnancement est appliqué durant cette première étape, permettant de choisir la ressource par rapport aux zones de conflits rencontrés vers chaque ressource. Cependant, comme la combinaison entre la planification de trajectoire et l'ordonnancement n'est pas possible d'une manière simultanée pour plusieurs agents (voir chapitre 2), l'ordonnancement ne sera pas appliqué à chaque itération et le superviseur devra autoriser ou non cet ordonnancement pour chaque agent.

- La seconde étape du planificateur, qui est locale, servira à assurer l'évitement de collisions inter–robots autour des zones de conflits. En effet, les informations fournies par le superviseur sont des trajectoires partielles qui dépendent de la zone de conflit à éviter. Cependant, comme ces trajectoires sont partielles, l'évitement de collision n'est pas garanti. Par conséquent, cette seconde étape assure l'évitement de collisions inter–agents.

Comme la planification de trajectoire se fait en deux étapes et d'une manière itérative, il devient nécessaire de réduire les temps de calculs car chaque trajectoire planifiée est complète (i.e. entre une configuration actuelle et une configuration finale d'une ressource). Pour cela, un algorithme d'optimisation par essaims particulaires sera utilisé. Il permet de calculer une trajectoire de manière rapide vers une solution proche de l'optimale. Cette méta heuristique nécessite un réglage des paramètres, qui sera aussi introduit dans ce chapitre.

Dans ce chapitre, des généralités sur l'architecture ainsi que sur les principales fonctions seront données. La manière dont le superviseur est conçu, permettant de résoudre les conflits et d'assister les agents, sera présentée. Ensuite, la planification de trajectoire et son algorithme en deux étapes seront décrits. Le réglage de l'optimisation par essaims particulaires et sa présentation seront présentés. Ce chapitre se terminera par une conclusion.

4.2 Architecture proposée et algorithme de navigation : généralités

Un algorithme de planification de trajectoire est proposé où un superviseur permet d'assister les agents afin qu'ils puissent effectuer l'opération de leur produit. L'objectif des agents est le même que précédemment : planifier une trajectoire sans collision vers une ressource à sélectionner permettant d'achever l'opération en cours dès que possible. Le rôle du superviseur est d'assister les agents dans leurs prises de décision afin d'atteindre leur objectif en respectant le cahier des charges.

Lors de la navigation, les décisions à prendre par les agents peuvent changer selon les informations relatives aux conflits fournies par le superviseur. Ces conflits sont représentés par des zones que l'agent doit éviter. Ceci implique une mise à jour des trajectoire car ces zones peuvent évoluer et dépendent des décisions prises par les agents à chaque instant. De plus, ces zones n'assurent pas nécessairement un évitement de collisions. Elles donnent plutôt un aspect d'anticipation des collisions que les agents doivent éviter.

Le niveau G.O.R.P a pour rôle de fournir des informations, relatives aux produits et aux ressources, au superviseur et aux agents. Les informations relatives aux ressources permettent d'avoir une connaissance de l'attente à chaque ressource à chaque instant de mise à jour. Les informations relatives aux opérations sont les mêmes que précédemment, c'est–à–dire un cahier des charges incluant une échéance et l'ensemble des ressources pouvant effectuer l'opération des produits transportés par les agents. Une vue d'ensemble, incluant l'algorithme de navigation, le niveau G.O.R.P et le superviseur, est proposée dans la Fig. 4.1.

4.2.1 Les conflits et leur gestion par superviseur : généralités

Dans cette approche, le superviseur remplace la négociation permettant de résoudre les conflits pouvant se produire lors de la navigation, réduisant ainsi les décisions que les agents doivent prendre. Les trois conflits considérés sont les suivants :

Conflit de ressource : Comme précédemment, il se produit lorsque deux agents voisins ou plus convoitent la même ressource. Ceci signifie que ces agents en conflit risquent d'arriver à la ressource au même moment. En effet, comme ils sont voisins, ils sont nécessairement à proximité les uns des autres. Par conséquent, leur distance par rapport à la ressource en conflit peut être sensiblement la même et ces agents, identiques en soi, risquent d'arriver au même moment. Inversement, s'ils ne sont pas voisins, il y a moins de risques qu'ils arrivent au même moment. En effet, en se dirigeant vers la même ressource, il est possible qu'ils deviennent voisins à un moment donné. De plus, les conflits de ressource se font uniquement avec les voisins car lorsqu'ils ne le sont pas, il peuvent changer de destination, impliquant pour l'agent de niveau de priorité bas de retarder son arrivée alors qu'il n'y a pas besoin de le faire.

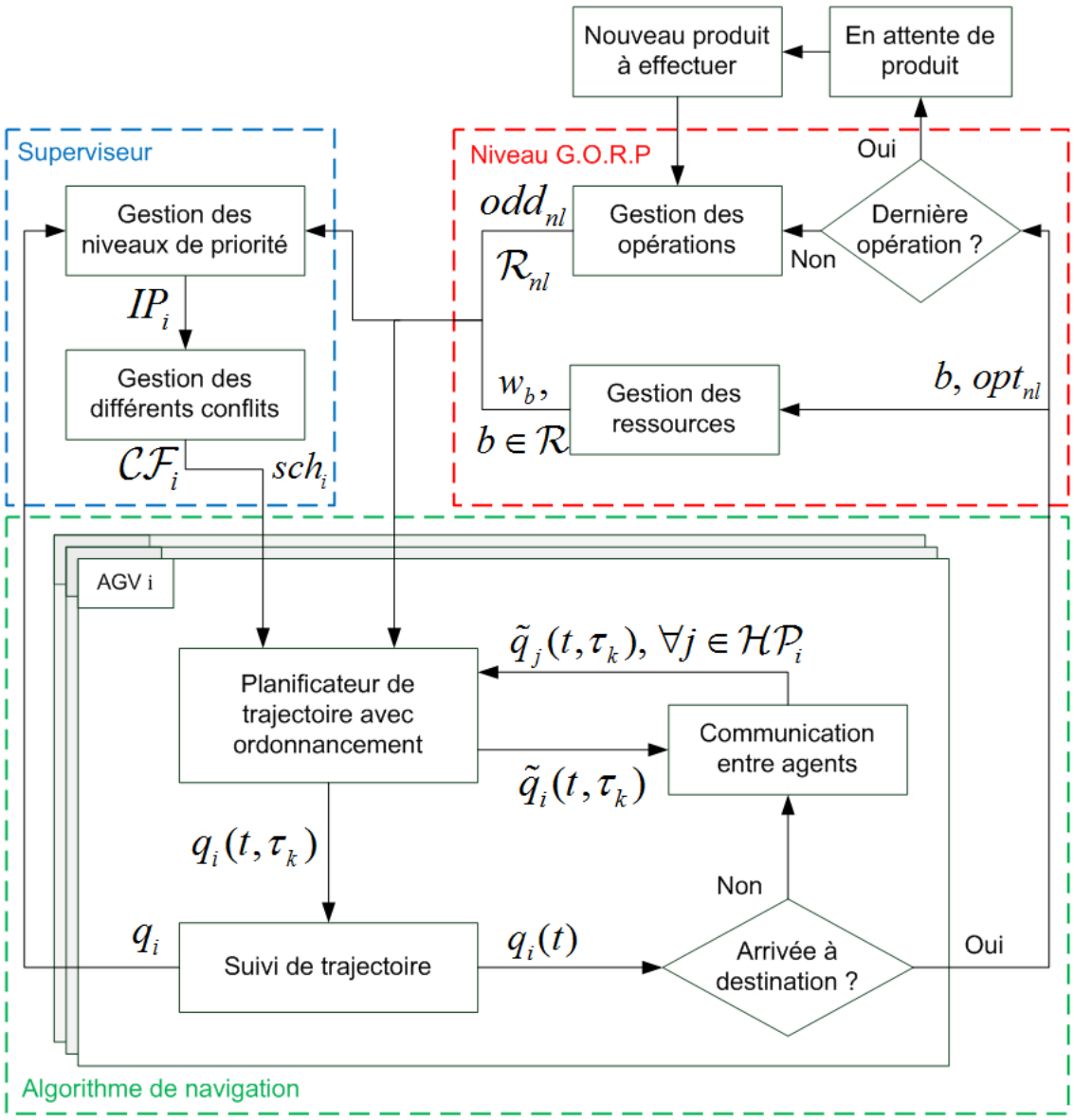


FIGURE 4.1 – Vue d'ensemble de l'architecture supervisée et de l'algorithme de navigation

Conflit d'ordonnancement : Ce conflit se présente lorsque deux agents voisins ou plus veulent ordonnancer leur produit au même instant de mise à jour comme nous l'avons présenté au chapitre 2. En effet, si plusieurs agents voisins ordonnançaient leur produit au même instant, les conflits de collision présentés ci–dessous ne pourrait pas être résolus aisément en parallèle à cause des choix de ressource non connus par ces agents. Pareillement au conflit précédent, ce conflit ne concerne que les agents qui sont voisins. Ceci permet notamment de réduire l'horizon d'ordonnancement.

Conflit de collision : Un conflit de collision apparaît lorsque les trajectoires directes de plusieurs agents sont suffisamment proches pour impliquer une collision. Notons que ces trajectoires directes sont des lignes droites liant la position actuelle des agents avec la position de la ressource choisie. Comme ces trajectoires ne peuvent être confondues (car cela signifierait qu'ils sont à la même configuration au même instant, impossible physiquement), les trajectoires directes ne peuvent se croiser qu'une seule fois. Par conséquent, un agent se dirigeant vers une ressource n'est en conflit de collision avec un autre une fois uniquement. Notons que ce conflit ne concerne pas uniquement les voisins. Tous les agents sont pris en compte dans la détection des conflits de collision, permettant d'une part de fournir des informations globales aux agents et d'autre part, de réduire les myopies sociale et temporelle de chaque agent.

Avant de traiter ces conflits, le superviseur doit d'abord établir une stratégie de priorité en se basant sur les performances individuelles des agents. Ensuite, il doit résoudre les conflits liés aux ressources afin de déterminer l'ordre d'arrivée des agents à celles–ci et résoudre les conflits d'ordonnancement permettant aux agents de savoir s'ils sont autorisés à ordonnancer leur produit ou pas pour l'instant actuel. Enfin, le superviseur permet de détecter les conflits de collisions pour chaque agent à l'instant actuel. Notons que les conflits de collision sont uniquement détectés, les agents seront en charge d'éviter ces conflits d'une manière décentralisée.

4.2.2 Algorithme de navigation : généralités

L'objectif de l'algorithme de navigation est semblable au précédent. Chaque agent doit calculer la meilleure ressource $b \in \mathcal{R}_{nl}$ et générer une trajectoire sans collision vers celle–ci. La meilleure ressource b est celle minimisant le temps pour achever l'opération ct_{nlb} composé par le temps de transport TT_{ik} vers cette ressource, du temps d'attente actuel de la ressource w_b et du temps de traitement opt_{nl}. L'algorithme de navigation de chaque agent, présenté sur la Fig. 4.1 combine les trois fonctions suivantes :

- Le *planificateur de trajectoire avec ordonnancement* utilise les contraintes physiques et temporelles pour sélectionner la ressource optimale en planifiant une trajectoire sans collision vers celle-ci. Il est divisé en deux étapes. La première utilise les informations globales des conflits fournies par le superviseur pour calculer une trajectoire $\widetilde{q}_i(t, \tau_k)$ représentant les intuitions des agents. Ces intuitions se caractérisent aussi en termes de sélection de ressource si l'ordonnancement est autorisé pour un agent. La seconde étape utilise ces intentions pour planifier une trajectoire sans collision en prenant en compte les données locales relatives aux voisins.
- Le *suivi de trajectoire* est le même que celui présenté dans le chapitre 3.
- Le bloc *communication entre agents* permet d'émettre et de recevoir les trajectoires intuitives entre les agents. Chaque agent ne reçoit que les intentions $\widetilde{q}_j(t, \tau_k)$ de ses voisins $j \in \mathcal{HP}_i$.

Ici, l'algorithme de navigation est appliqué graduellement au cours du temps, amenant les agents à mettre à jour leur trajectoire d'une manière itérative et en parallèle. Lorsqu'un agent arrive à une ressource b, le temps d'attente w_b de cette ressource est mis à jour par rapport au temps de traitement opt_{nl} de l'opération n du produit l transporté par l'agent en question. Ceci permet à d'autres agents de reconsidérer cette ressource comme étant la meilleure pour achever leur opération respective. Notons que les agents n'ordonnancent pas leur produit à chaque instant de mise à jour τ_k et doivent attendre un certain nombre d'instants sk_i avant le prochain ordonnancement (on pourrait qualifier ceci d'ordonnancement à horizon glissant).

4.3 Résolution et détection de conflits par le superviseur

Le superviseur que l'on propose utilise différents algorithmes pour résoudre les conflits de ressource et d'ordonnancement. De plus, il transmet les informations requises permettant aux agents de résoudre leurs propres conflits de collision lorsqu'ils sont concernés. De la même manière que la négociation hétérarchique entre agents, présentée dans le Chapitre 3, il est difficile, voire impossible, de donner une justification formelle de l'absence d'interblocage dû au couplage entre résolution de conflits et planification de trajectoire.

La détection a deux rôles étroitement liés. D'une part, la myopie temporelle est réduite puisque le superviseur donne des informations supplémentaires aux agents permettant d'atteindre leur objectif mais aussi d'éviter des conflits difficiles à résoudre (des conflits avec un grand nombre d'agents à prendre en compte par exemple). D'autre part, comme les agents ont plus d'informations, ils peuvent choisir la ressource par rapport aux conflits de collisions, c'est à dire que la ressource qu'ils choisissent a plus de chance d'être la meilleure en termes de temps pour achever l'opération. L'ensemble des étapes de l'algorithme du superviseur est donné sur la Fig. 4.2.

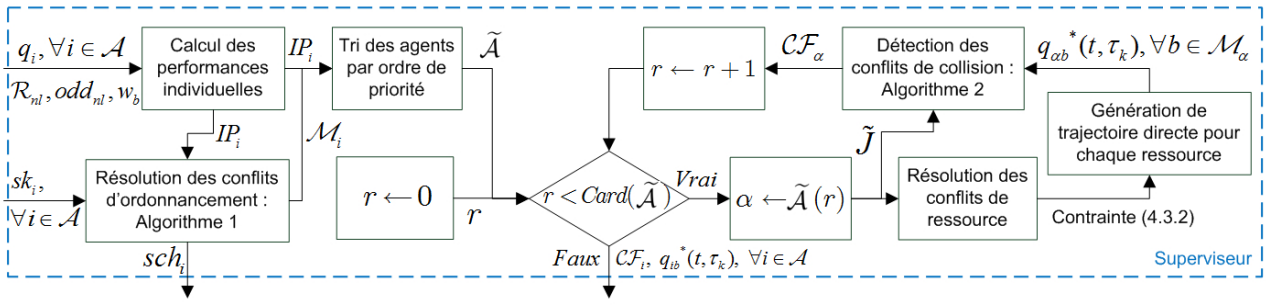


FIGURE 4.2 – Algorithme du superviseur proposé pour résoudre les conflits de ressource et d'ordonnancement et pour détecter les conflits de collision

Avant de résoudre les différents conflits et de détecter les possible collisions, le superviseur doit calculer les performances individuelles. Ensuite, il résout les différents conflits d'ordonnancement,

permettant d'autoriser ou non l'ordonnancement aux agents. Lorsque ces conflits sont résolus, le superviseur tri les agents par ordre de priorité en utilisant les performances individuelles afin, d'une manière séquentielle, de résoudre les conflits de ressource et de détecter les conflits de collision. Pour chaque étape de la résolution séquentielle, le superviseur commence par résoudre les conflits de ressource pour permettre de définir l'ordre d'arrivée des agents à celles-ci. Ensuite, les trajectoires directes sont générées. Pour finir, les conflits de collision sont détectées à l'aide de ces trajectoires directes. Ces différentes étapes sont détaillées dans les sous-sections 4.3.1–4.3.6.

4.3.1 Calcul des performances individuelles

Pour traiter l'ensemble des conflits, il est nécessaire que le superviseur commence par calculer les performances individuelles de l'ensemble des agents en utilisant l'équation (3.3.5). A l'aide de cet indice de performance, le superviseur peut définir les niveaux de priorité actuels de chaque agent à l'aide des ensembles $\mathcal{HP}_i$, pour tout $i \in \mathcal{A}$ suivants :

$$\mathcal{HP}_i = \{j \in \mathcal{N}_i, IP_i \leq IP_j\} \tag{4.3.1}$$

Notons ici que contrairement au chapitre précédent, nous n'utilisons pas l'approche sociale. D'abord, l'idée est d'utiliser le même indice de priorité pour tous les conflits afin d'éviter que les algorithmes permettant de les résoudre n'entrent en conflit entre eux. Le choix de ne pas utiliser l'approche sociale est basé sur deux propos. D'une part, il y a des conflits d'ordonnancement à résoudre dans cette approche en plus des deux autres (ressource et collision). D'autre part, cette approche considère aussi les temps d'attente aux ressources qui peuvent varier lors de la navigation. Une approche sociale permettrait au superviseur de résoudre les conflits en considérant tous les agents pour favoriser leur comportement global. Cependant, lorsque plusieurs agents atteignent successivement une même ressource (à la suite les uns des autres), les autres agents se dirigeant vers cette même ressource seront désavantagés (car soumis à une perte de performances individuelles due à l'augmentation du temps d'attente). Il est donc préférable que le niveau de priorité de ces agents devienne plus haut afin qu'ils puissent réordonnancer leur produit au plus vite. De ces propos, on voit l'intérêt de favoriser les comportement individuels plutôt que le comportement global pour l'établissement du mécanisme de priorité. Cependant, comme un aspect individuel est appliqué, il serait intéressant de faire une remarque sur les impacts en termes de myopie et de performance globale.

Remarque 4.1 *En termes de myopie, le mécanisme de priorité rend les agents ayant des performances mauvaises myopes par rapport à ceux qui ont de meilleures performances. Ceci est voulu car de cette façon, les agents ayant de mauvaises performances n'auront pas à éviter (ou pourront "ignorer") les autres agents avec de meilleures performances. Par conséquent, les performances de ces agents deviendront moins mauvaises. En termes de performances globales, l'impact de ce mécanisme de priorité n'agit que sur les agents ayant de bonnes performances. En effet, le mécanisme de priorité donne un*

niveau de priorité haut aux agents ayant de mauvaises performances, permettant ainsi d'éviter que ces mauvaises performances le deviennent encore plus. On peut remarquer que ce mécanisme permet d'obtenir un compromis dans les performances globales où d'un coté les mauvaises performances de certains agents doivent être améliorées ou maintenues en réduisant les performances des autres agents. Cependant, il n'y a pas de répercussion car comme les indices de performances IP_i sont mises à jour graduellement au cours du temps, le niveau de priorité des agents dont les performances deviennent moins bonnes devient plus haut. Ceci permet d'assurer que l'opération de l'agent respecte le cahier des charges. En effet, comme nous l'avons vu, un agent ayant des mauvaises performances est celui qui achèvera l'opération de son produit à une date proche de l'échéance et le mécanisme de priorité lui permettra donc "d'ignorer" les autres agents pour y parvenir.

4.3.2 Résolution des conflits d'ordonnancement

Pour pouvoir déterminer quels agents sont autorisés à ordonnancer leur produit, le superviseur applique l'algorithme 4.1 ci–dessous :

Algorithme 4.1 Résolution des conflits d'ordonnancement pour tous les agents i à l'instant τ_k

pour tout agents $i \in \mathcal{A}$ **faire**
  si $sk_i \geq m$ **alors**
    $sv \leftarrow VRAI$ {Variable locale}
    pour tout agents $j \in \mathcal{N}_i$ **faire**
      si $sk_j \geq m \wedge sv = VRAI$ **alors**
        si $(sk_i < sk_j) \vee (sk_i = sk_j \wedge IP_i < IP_j)$ **alors**
          $sv \leftarrow FAUX$
        fin si
      fin si
    fin pour
    si $sv = VRAI$ **alors**
      $sch_i \leftarrow 1$
      $sk_i \leftarrow 0$
    sinon
      $sch_i \leftarrow 0$
      $sk_i \leftarrow sk_i + 1$
    fin si
  sinon
    $sk_i \leftarrow sk_i + 1$
  fin si
  si $sch_i = 1$ **alors**
    $\mathcal{M}_i \leftarrow \mathcal{R}_{nl}$
  sinon
    $\mathcal{M}_i \leftarrow cr_{i(k-1)}$
  fin si
  renvoyer sch_i **et** $\mathcal{M}_i$
fin pour

Après avoir ordonnancé leur produit, chaque agent doit attendre un certain nombre de mises à jour, noté sk_i, avant le prochain ordonnancement. Après avoir atteint cette valeur (i.e. $sk_i \geq m$), un agent i peut réordonnancer son produit s'il a la plus haute priorité parmi ses voisins. Une variable booléenne locale sv est introduite et mise à l'état $VRAI$. Cette variable est mise à l'état $FAUX$ lorsqu'au moins un de ses voisins $j \in \mathcal{N}_i$ doit ordonnancer son produit avant lui. Ceci se produit lorsque le nombre de mises à jour de cet agent j depuis son dernier ordonnancement est supérieur, i.e. $sk_j > sk_j$.

Si les nombres de mises à jour depuis le dernier ordonnancement de ces agents sont égaux (i.e. $sk_i = sk_j$), l'agent j ordonnancera son produit avant l'agent i uniquement si les performances individuelles de celui–ci sont plus pauvres (i.e. $IP_i < IP_j$). Si aucun agent j n'est autorisé à ordonnancer son produit avant l'agent i, la variable sch_i est mise à 1, autorisant ainsi un ordonnancement à l'agent i, et le nombre de mises à jour sk_i est remis à 0. Sinon, la variable sch_i reste égale à 0 et le nombre de mises à jour sk_i de l'agent i est incrémenté de 1.

Il est intéressant de noter que $\mathcal{M}_i$ est l'ensemble des ressources à considérer pour les prochains conflits (collision et ressource). Cet ensemble est différent selon que l'agent i peut ordonnancer son produit ou non. Si $sch_i = 0$, $\mathcal{M}_i$ contient uniquement la ressource choisie à la dernière mise à jour (i.e. $cr_{i(k-1)}$) et une seule trajectoire directe sera utilisée pour traiter les deux autres conflits. Sinon, $\mathcal{M}_i$ devient l'ensemble des ressources possibles $\mathcal{R}_{nl}$ et les conflits de ressource et de collision seront traités pour chaque ressource $b \in \mathcal{M}_i$.

Lorsque l'algorithme 1 est terminé, le superviseur fournit aux agents la variable binaire sch_i leur permettant de savoir s'ils sont autorisés à ordonnancer leur produit à l'instant de mise à jour τ_k. Ensuite, l'ensemble des ressources $\mathcal{M}_i$ de chaque agent i, dépendant de la valeur de sch_i, est utilisé pour résoudre les autres conflits. Enfin, la nouvelle valeur de la variable sk_i pour chaque agent i est gardée par le superviseur pour la prochaine mise à jour (voir l'entrée sk_i sur la Fig. 4.2).

4.3.3 Tri des agents pour résolution séquentielle

Afin de pouvoir résoudre les conflits de ressource et détecter les conflits de collision, le superviseur trie les agents par rapport à leur niveau de priorité. L'ensemble trié d'agents $\widetilde{\mathcal{A}} = \{\alpha_1, \alpha_2, ..., \alpha_{N_i}\}$ respecte les inégalités $IP_{\alpha_1} > IP_{\alpha_2} > \ldots > IP_{\alpha_{N_i}}$. Le premier élément de $\widetilde{\mathcal{A}}$ correspond à l'agent ayant le niveau de priorité le plus haut tandis que le dernier élément correspond à celui qui a le niveau de priorité le plus bas. Ce tri d'agents permet de traiter d'une manière séquentielle les conflits de chaque agent i en utilisant uniquement les autres agents ayant un niveau de priorité plus haut que celui–ci.

Pour cette résolution séquentielle, une boucle sur l'ensemble $\widetilde{\mathcal{A}}$ est utilisée. Chaque itération commence par assigner un agent α correspondant à l'élément r de l'ensemble $\widetilde{\mathcal{A}}$. Ensuite, l'ensemble $\widetilde{J} = \{j \in \widetilde{\mathcal{A}} : IP_j > IP_\alpha\}$ d'agents est calculé pour résoudre les conflits de ressource et détecter les conflits de collision. Notons que quelque soit l'agent α, tous les conflits des agents de l'ensemble $\widetilde{J}$ ont déjà été résolu préalablement.

Les dernières étapes de l'algorithme du superviseur, à savoir la résolution des conflits de ressource, la génération de trajectoire directe et la détection de conflits de collision, sont décrites dans les trois sous-sections suivantes. Lorsque la résolution séquentielle est terminée (boucle sur l'ensemble $\widetilde{\mathcal{A}}$, c.f. Fig. 4.2), le superviseur peut fournir les ensembles de conflits détectés $\mathcal{CF}_i$, ainsi que les trajectoires directes $q^*_{ib}(t, \tau_k)$, pour chaque agent $i \in \mathcal{A}$.

4.3.4 Résolution des conflits de ressource

Afin de déterminer l'ordre d'arrivée à chaque ressource, le superviseur doit résoudre les conflits de ressource pour chaque agent $\alpha \in \widetilde{\mathcal{A}}$. Les conflits de ressource d'un agent α sont résolus en retardant son arrivée à chaque ressource $b \in \mathcal{M}_\alpha$. Ceci est représenté en ajoutant la contrainte suivante pour chaque ressource $b \in \mathcal{M}_\alpha$:

$$TT^*_{\alpha b} > TT^*_{jb}, \quad \forall j \in \{\widetilde{J} \cap \mathcal{N}_\alpha : \ \widetilde{cr}_{jk} = b\} \tag{4.3.2}$$

où $TT^*_{\alpha b}$ est le temps de transport associé à la trajectoire directe $q^*_{\alpha b}(t, \tau_k)$ et $\widetilde{cr}_{jk}$ a été calculé pour chaque agent $j \in \widetilde{J}$ lors des itérations précédentes par l'algorithme 4.2 détaillé plus bas. Suite à cette résolution de conflit de ressource, la contrainte (4.3.2) sera utilisée lors de la génération de trajectoire directe pour chaque ressource $b \in \mathcal{M}_\alpha$.

4.3.5 Génération de trajectoire directe

Afin de pouvoir détecter les conflits de collision, il est nécessaire de calculer la distance entre les agents à l'aide de leur trajectoire respective. Cependant, il faut avant tout générer ces trajectoires. Pour cela, l'idée est de générer des trajectoires en ligne droite car elle permettent d'atteindre la destination voulue le plus rapidement possible. Ceci est un élément clé puisque ces trajectoires seront utilisées par les agents lors de la planification de trajectoire. De plus, si les agents n'ont pas de collisions à éviter, ils n'auront pas besoin de générer une trajectoire et pourront utiliser leur trajectoire en ligne droite.

Pour l'instant de mise à jour τ_k, la trajectoire directe, liant la configuration actuelle q_α de l'agent α et une de ses ressource $b \in M_\alpha$, est vue comme la "meilleure" trajectoire en ligne droite (i.e. celle minimisant le temps de transport $TT^*_{\alpha bk}$) qui respecte la contrainte de vitesse $\|\dot{q}^*_{\alpha b}(t, \tau_k)\| \leq v_{max}$, $\forall t \geq \tau_k$, les contraintes liées aux conflits de ressource (4.3.2) ainsi que les conditions aux limites suivantes :

$$q^*_{\alpha b}(\tau_k, \tau_k) = q_\alpha \tag{4.3.3}$$

$$\dot{q}^*_{\alpha b}(\tau_k, \tau_k) = v_\alpha \tag{4.3.4}$$

$$q^*_{\alpha b}(\tau_k + TT^*_{\alpha bk}, \tau_k) = q_b \tag{4.3.5}$$

$$\dot{q}^*_{\alpha b}(\tau_k + TT^*_{\alpha bk}, \tau_k) = 0 \tag{4.3.6}$$

Ces trajectoires directes sont ensuite utilisées pour la détection des conflits de collision de l'agent α à l'aide de l'algorithme 4.2. Notons que comme la trajectoire est une ligne droite, il est aisé de déterminer les points de contrôle de la trajectoire spline. La trajectoire directe ne nécessite que ces points de contrôle ainsi que le temps de transport. Le temps de transport est facilement calculable lorsqu'aucune contrainte de conflit de ressource n'est à respecter. Sinon, il faut trouver ce temps de transport d'une manière simple tout en respectant cette contrainte, en choisissant par exemple $TT^*_{\alpha b} = \max_{\widetilde{J}\cap\mathcal{N}_\alpha : \widetilde{cr}_{jk}=b} TT^*_{jb} + T_c$ où T_c est le temps entre deux mises à jour et $\max_{\widetilde{J}\cap\mathcal{N}_\alpha} TT^*_{jb}$ est le temps de transport le plus grand parmi tous les agents $j \in \widetilde{J}$ voisins de l'agent α se dirigeant vers la ressource b.

Remarque 4.2 *Comme les variables permettant de générer la trajectoire directe sont facilement calculables, aucun algorithme de planification de trajectoire n'est nécessaire. Ceci est un élément très important au vu des temps de calcul de ces algorithmes, qui seraient dans ce cas utilisés d'une manière centralisée pour l'ensemble des agents.*

4.3.6 Détection des conflits de collision

Après avoir générer une trajectoire directe vers chaque ressource $b \in \mathcal{M}_\alpha$, le superviseur entreprend la détection des différents **conflits de collision** que l'agent α peut avoir avec d'autres agents $j \in \widetilde{J}$. L'algorithme 4.2 ci–dessous est proposé pour la détection de ces conflits pour l'agent α.

Un conflit de collision pour un agent α, illustré à l'aide de la Fig. 4.3–(a), est défini comme un quadruplé $\mathcal{C}_{\alpha jb} = (j, b, I_{\alpha j}, cd_{\alpha j})$ où j est l'agent à éviter sur l'intervalle de temps $I_{\alpha j}$ lorsque la ressource b est choisie. La zone où l'agent α devra éviter l'agent j est en jaune et dépend de la trajectoire directe $q^*_{jc}(t, \tau_k), c = \widetilde{cr}_{jk}$ pour $t \in I_{\alpha j}$. $cd_{\alpha j}$ est le degré du conflit dépendant des conflits détectés par l'agent j, expliqué durant l'algorithme.

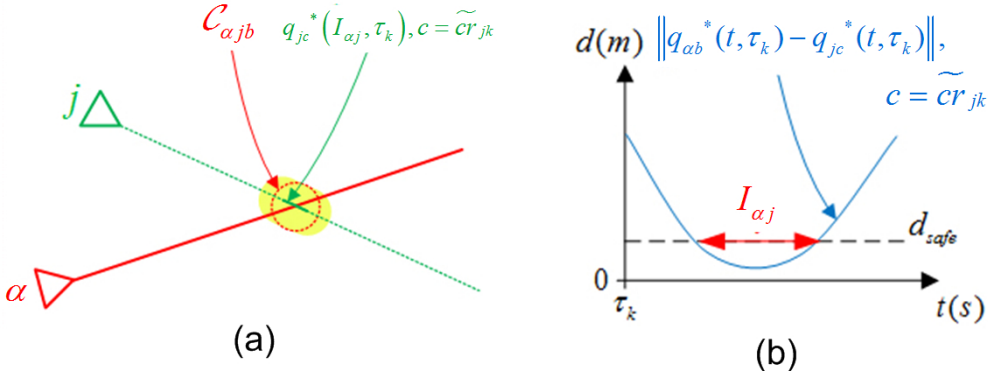


FIGURE 4.3 – (a) Conflit entre deux agents α et j (b) Intervalle du conflit $I_{\alpha j}$

Pour détecter un conflit de collision vers la ressource $b \in \mathcal{M}_\alpha$ avec un agent $j \in \widetilde{J}$, le superviseur vérifie la distance entre les trajectoires directes des deux agents$^{(a)}$ comme le montre la Fig. 4.3–(b). Si

Algorithme 4.2 Détection des conflits de collision de l'agent α à l'instant τ_k

pour tout $b \in \mathcal{M}_\alpha$ **faire**
  pour tout $j \in \widetilde{J}$ **faire**
    $I_{\alpha j} \leftarrow \{t : \|q^*_{\alpha b}(t, \tau_k) - q^*_{j\widetilde{cr}_{jk}}(t, \tau_k)\| < d_{safe}\}$ $^{(a)}$
    si $I_{\alpha j} \neq \varnothing$ **alors**
      création d'un nouveau conflict $C_{\alpha jb}$ $^{(b)}$
      $cd_{\alpha j} \leftarrow 1$
      si $\widetilde{\mathcal{C}}_j \neq \varnothing$ **alors**
        pour tout $(p, \widetilde{cr}_{jk}, I_{jp}, cd_{jp}) \in \widetilde{\mathcal{C}}_j$ **faire**
          si $I_{\alpha j} \cap I_{jp} \neq \varnothing$ **alors**
            $I_{\alpha j} \leftarrow I_{\alpha j} \cup I_{jp}$ $^{(c)}$
            $cd_{\alpha j} \leftarrow cd_{\alpha j} + cd_{jp}$ $^{(d)}$
          fin si
        fin pour
      fin si
      $\mathcal{C}_{\alpha jb} \leftarrow (j, b, I_{\alpha j}, cd_{\alpha j})$
    fin si
  fin pour
  $\widetilde{ct}_{nlb} \leftarrow \tau_k + TT^*_{ibk} + w_{bk} + opt_{nl} + \sum_{\mathcal{C}_{\alpha jb}} cd_{\alpha j} \cdot \triangle t$ $^{(e)}$
fin pour
$\widetilde{cr}_{\alpha k} \leftarrow \arg\min_b(\widetilde{ct}_{nlb})$ $^{(f)}$
$\widetilde{\mathcal{C}}_j \leftarrow \{\mathcal{C}_{\alpha jb} : b = \widetilde{cr}_{\alpha k}, \forall j \in \widetilde{J}\}$ $^{(g)}$
$\mathcal{CF}_\alpha \leftarrow \{\mathcal{C}_{\alpha jb}, \forall b \in \mathcal{M}_\alpha, \forall j \in \widetilde{J}\}$ $^{(h)}$
renvoyer $\mathcal{CF}_{\alpha k}$ **et** $\widetilde{\mathcal{C}}_j$

une collision se produit (i.e. $I_{\alpha j} \neq \varnothing$), un nouveau conflit est créé$^{(b)}$ avec un degré égal à 1. Ensuite, l'algorithme vérifie si durant l'intervalle de temps $I_{\alpha j}$, d'autres conflits de collision ont été détectés pour l'agent j lorsqu'il se dirige vers la ressource intuitive $\widetilde{cr}_{jk}$. Le cas échéant, l'intervalle de temps $I_{\alpha j}{}^{(c)}$ ainsi que le degré du conflit $cd_{\alpha j}{}^{(d)}$ de l'agent α sont mis à jour. Tous les conflits vers toutes les ressources $b \in \mathcal{M}_\alpha$ sont rassemblés dans l'ensemble de conflits $\mathcal{CF}_\alpha{}^{(h)}$ qui sera fourni à l'agent α.

Le degré d'un conflit permet aux agents de contourner la zone de conflit d'une manière différente. En effet, si le degré n'était pas ajouté pour les conflits, les agents éviteraient tous leur conflit en restant autour d'une petite zone (voir Fig. 4.4–(A)), ce qui décalerait juste le conflit mais en engendrerait d'autres autour de celle-ci. Notons que l'évitement d'une zone de conflit par un agent i est réalisé lors de la première étape de planification (de la trajectoire intuitive $\widetilde{q}_{ib}(t, \tau_k)$ vers la ressource b). En utilisant un degré de conflit, les agents doivent éviter la zone mais avec un rayon d'évitement différent (représenté par la contrainte (4.4.11) dans la première étape de planification de trajectoire), ce qui est plus représentatif de ce qu'un agent devrait faire pour éviter plusieurs autres agents comme le montre la Fig. 4.4–(B). D'une manière abstraite, on pourrait assimiler le degré d'un conflit comme un effet de couche où un agent ayant un conflit de degré n doit éviter la couche n, c'est–à–dire n fois plus grande qu'un conflit de degré 1.

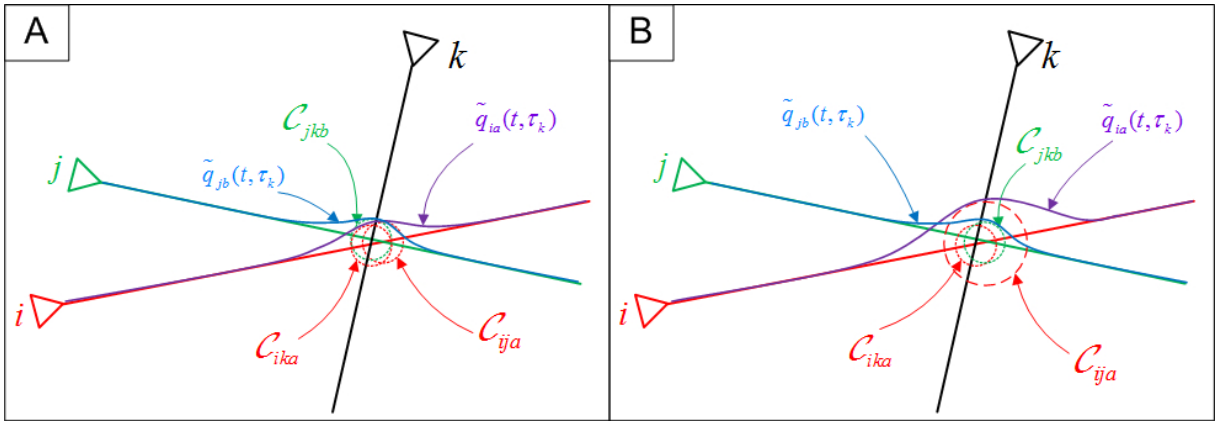


FIGURE 4.4 – Détection des conflits : (A) sans degré de conflit (B) avec degré

Il est utile de souligner que l'ensemble de conflits $\mathcal{CF}_{\alpha k}$ n'est pas directement utilisé dans la détection des conflits des agents suivants $\tilde{\mathcal{A}}$. En effet, puisqu'une seule ressource $cr_{\alpha k}$ sera choisie au final par l'agent α à l'instant τ_k, il n'y a pas de sens à considérer les conflits se produisant pour les ressources non choisies $b \neq cr_{\alpha k}$. En effet, il est nécessaire de définir une intuition sur la ressource que l'agent α choisira si l'ordonnancement lui est autorisé. Cette ressource intuitive $\widetilde{cr}_{\alpha k}$ est calculée en minimisant une estimation du temps pour achever l'opération $\widetilde{ct}_{nlb}{}^{(f)}$. Cette estimation heuristique est calculée par ajout de pénalités pour chaque conflit rencontré vers la ressource spécifiée (voir le terme $\sum_{\mathcal{C}_{\alpha jb}} cd_{\alpha j} \cdot \triangle t$ où $\triangle t$ est un paramètre de conception)$^{(e)}$.

L'ensemble $\widetilde{\mathcal{C}}_j$ rassemble tous les conflits se produisant sur la trajectoire directe menant à la ressource intuitive $\widetilde{cr}_{ik}{}^{(g)}$. Il est important de noter que la ressource intuitive $\widetilde{cr}_{ik}$ sera utilisée pour résoudre les conflits de ressource des agents suivants (voir l'équation (4.3.2)) et pour la détection de leurs conflits de collision$^{(a)}$. Pour finir, un exemple de détection de conflits pour quatres agents est illustré à l'aide de la Fig. 4.5.

Pour cet exemple, considérons les agents i, j, k et l se déplaçant vers les ressources respectives a, b c et d. L'ordre de priorité est $\widetilde{\mathcal{A}} = \{l, k, j, i\}$. L'agent l, ayant le niveau de priorité le plus haut, n'a pas de conflit à résoudre i.e. $\widetilde{J} = \varnothing$ (voir Fig. 4.5–(A)). Ensuite, la trajectoire de l'agent k est générée et un conflit $\mathcal{C}_{klc} = (l, c, I_{kl}, 1)$ est créé car $\widetilde{J} = \{l\}$ (voir Fig. 4.5–(B)). La trajectoire directe de l'agent j a deux conflits engendrés avec les agents k et l ($\widetilde{J} = \{l, k\}$). Le premier $\mathcal{C}_{jlb} = (l, b, I_{jl}, 1)$ avec l'agent l est de degré 1 car l'agent l n'a pas de conflit (voir Fig. 4.5–(C)). Le second conflit $\mathcal{C}_{jkb} = (k, b, I_{jk} \cup I_{kl}, 2)$, avec l'agent k, est de degré 2 car il dépend du conflit $\mathcal{C}_{klc} = (l, c, I_{kl}, 1)$ que l'agent k a avec l'agent l. Enfin, la trajectoire directe de l'agent i engendre trois conflits avec les autres agents ($\widetilde{J} = \{l, k, j\}$). Le premier conflit $\mathcal{C}_{ila} = (l, a, I_{il}, 1)$ est avec l'agent l. Le second $\mathcal{C}_{ika} = (k, a, I_{ik} \cup I_{kl}, 2)$ avec l'agent k qui dépend du conflit $\mathcal{C}_{klc} = (l, c, I_{kl}, 1)$. Le dernier conflit $\mathcal{C}_{ija} = (j, a, I_{ij} \cup I_{jk} \cup I_{kl}, 3)$ est de degré 3 dû au conflit $\mathcal{C}_{jkb} = (k, b, I_{jk} \cup I_{kl}, 2)$ qui dépend lui même du conflit $\mathcal{C}_{klc} = (l, c, I_{kl}, 1)$ (voir Fig. 4.5–(D)).

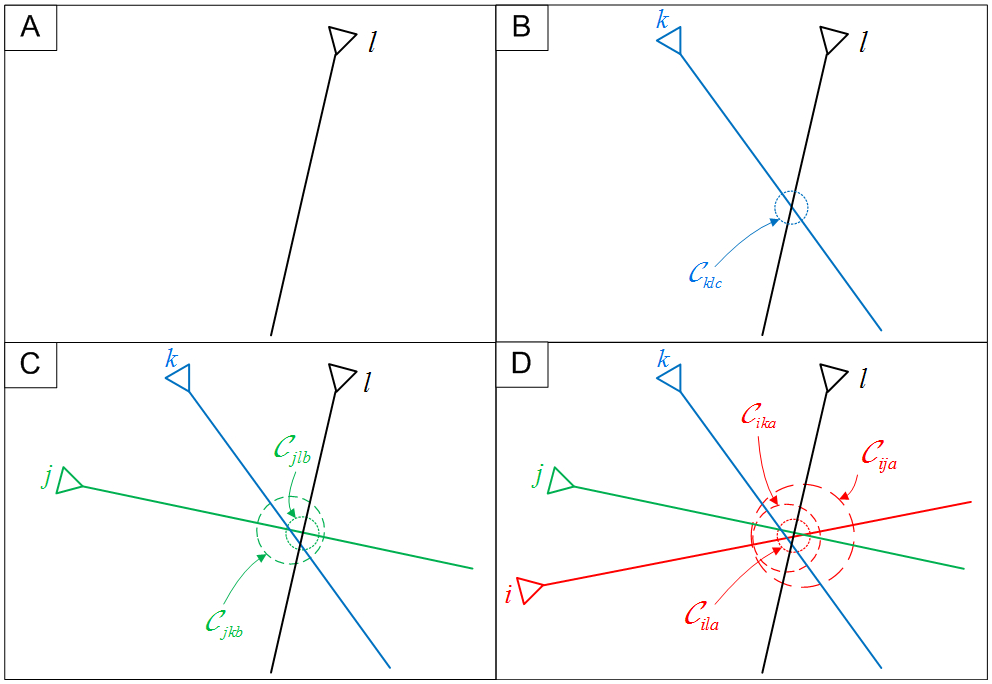


FIGURE 4.5 – Détection des conflits par le superviseur : exemple avec quatre agents

4.4 Planification de trajectoire avec ordonnancement

4.4.1 Vue d'ensemble de l'algorithme de navigation

Le planificateur de trajectoire proposé s'exprime sous forme d'un problème d'optimisation sous contraintes. Une approche décentralisée, où le problème complet est décomposé en sous–problèmes plus simples, est proposée pour une résolution en ligne. Chaque sous–problème est implémenté dans chaque agent. La Fig. 4.6 montre l'algorithme de planification proposé pour chaque agent i.

La finalité du planificateur de trajectoire est que, à chaque mise à jour de trajectoire τ_k, tous les agents aient généré une trajectoire optimale satisfaisant le comportement de l'agent représenté par (2.3.3)–(2.3.4), la contrainte manufacturière (2.3.1) décrite par le cahier des charges de production et la contrainte permettant l'évitement de collisions (2.3.10). Rappelons que les trajectoires de chaque agent sont mises à jour graduellement au cours du temps où T_c est l'horizon de calculs tel que $T_c = \tau_{k+1} - \tau_k, \forall k \in \mathbb{N}$.

L'algorithme de planification proposé est divisé en deux étapes où une trajectoire est planifiée pour chacune d'entre elles.

- L'étape 1 consiste à générer une trajectoire $\widetilde{q}_i(t, \tau_k)$ représentant les intentions de l'agent tout en évitant les conflits de collision fournis par le superviseur. Ces intentions incluent aussi le choix de ressource de l'agent si le superviseur a autorisé l'ordonnancement. Cette étape d'évitement de collision peut être considérée comme globale au vu des données fournies par le superviseur.

- L'étape 2 consiste à générer la trajectoire finale $q_i(t, \tau_k)$ permettant l'évitement de collisions avec les voisins de l'agent i. Comme l'évitement de collision concerne uniquement les voisins de l'agent, on peut qualifier cette étape de locale.

Rappelons que pour chaque trajectoire ($\tilde{q}_i(t, \tau_k)$ ou $q_i(t, \tau_k)$), le premier élément désigne le temps alors que le second permet de spécifier à quelle instant de mise à jour τ_k la trajectoire est planifiée.

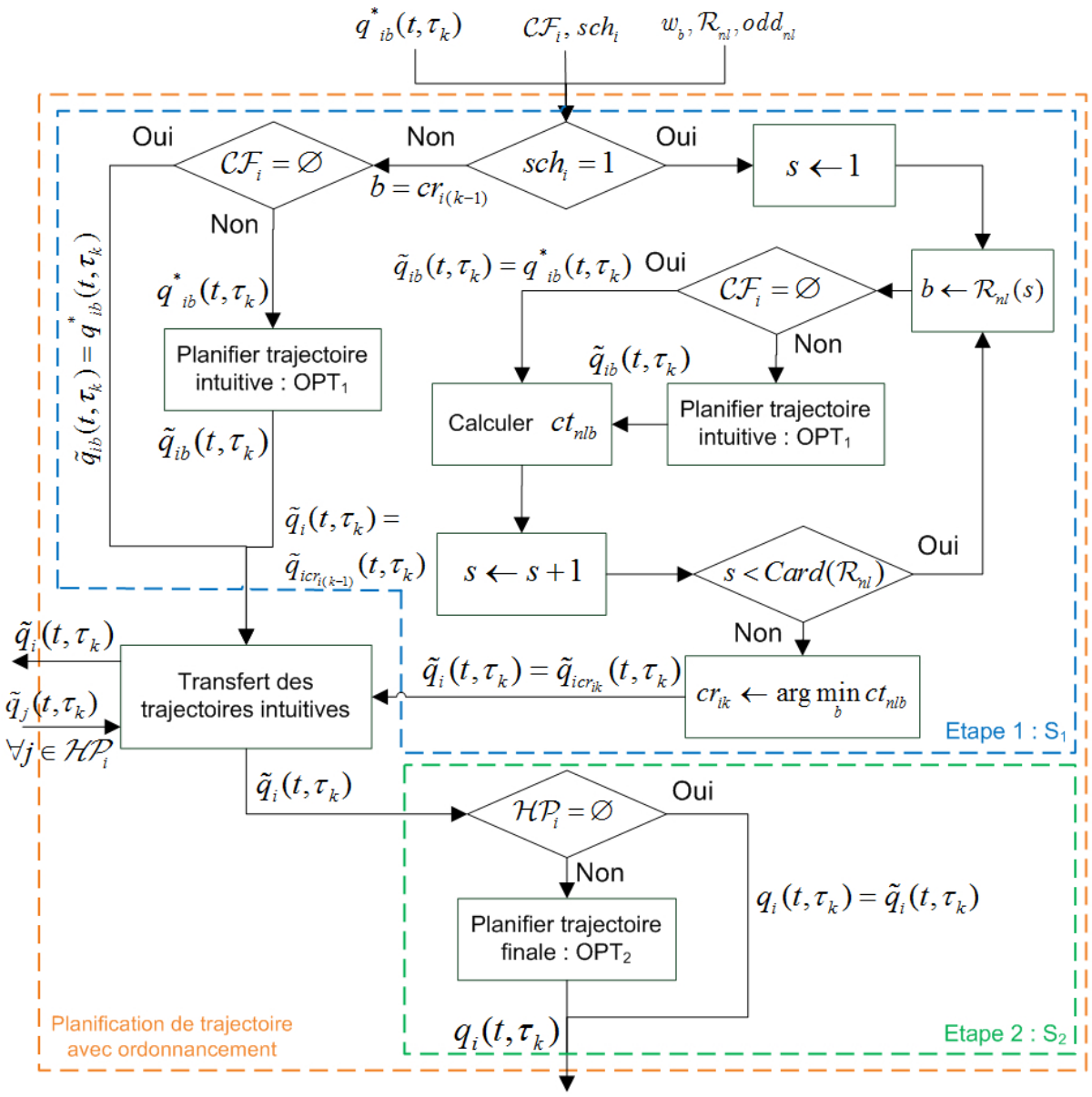


FIGURE 4.6 – Algorithme de planification de trajectoire avec ses deux étapes

4.4.2 Première étape de planification

L'étape 1 permet de générer une trajectoire intuitive $\widetilde{q}_i(t, \tau_k)$ en évitant des conflits sur des zones spécifiques. Ces zones de conflits $\mathcal{C}_{ijb} = (j, b, I_{ij}, cd_{ij})$, préalablement données par le superviseur, sont représentées par des trajectoires directes partielles $q_j^*(I_{ij}, \tau_k)$ d'autres agents j ayant un niveau de priorité plus haut $IP_j > IP_i$. Celles–ci sont dites partielles car uniquement la partie de la trajectoire correspondant à l'intervalle de temps $I_{ij} \neq \varnothing$ est fournies à l'agent pour chaque conflit.

Comme le montre la Fig. 4.6, cette première étape est différente si l'ordonnancement est autorisé ou non par le superviseur.

- Si l'ordonnancement est autorisé, i.e. $sch_i = 1$, l'agent i transportant le produit l pour l'opération n doit choisir une ressource $b \in \mathcal{R}_{nl}$. Une trajectoire intuitive doit alors être calculée pour chaque ressource b tout en assurant l'évitement des conflits détectés vers chacune de ces ressources. Dès que les trajectoires intuitives sont générées, les temps de transport $\widetilde{TT}_{ik}$ vers chaque ressource b, obtenus en résolvant le problème d'optimisation OPT_1, sont utilisés pour calculer les temps pour achever l'opération vers chacune de ces ressources :

$$ct_{nlb} = \tau_k + \widetilde{TT}_i + w_b + wp_{ib} + opt_{nl}, \qquad b \in \mathcal{R}_{nl} \tag{4.4.1}$$

  où w_b est le temps d'attente à la ressource b et opt_{nl} est le temps de traitement de l'opération. Le terme wp_{ib} est un temps d'attente fictif. Il représente un temps additionnel pour lequel l'agent i devra attendre que l'opération pour chaque agent j soit effectuée à cette même ressource. Ceci permet de mieux choisir la ressource car ces autres agents j ont un niveau de priorité plus haut et seront nécessairement arrivés à la ressource avant l'agent i. Ce temps est estimé de la manière suivante :

$$wp_{ib} = \sum_{j \in \mathcal{HP}_i : cr_{jk} = b} opt_{(n_j)(l_j)} \tag{4.4.2}$$

  où $opt_{(n_j)(l_j)}$ est le temps de traitement de l'opération (n_j) du produit (l_j) transporté par l'agent j. Il est important de noter que lorsque l'agent i est autorisé à ordonnancer son produit, ses voisins $j \in \mathcal{HP}_i$ n'ordonnancent pas leur produit au même instant de mise à jour τ_k puisque cela n'a pas été permis par le superviseur, i.e. $sch_i = 1 \Rightarrow sch_j = 0 \Rightarrow cr_{jk} = cr_{j(k-1)}$.

  Ensuite, la ressource est choisie en minimisant ce temps pour achever l'opération,

$$cr_{ik} = \arg\min_b ct_{nlb} \tag{4.4.3}$$

- Lorsque l'agent i n'est pas autorisé à ordonnancer son produit, c'est–à–dire lorsque $sch_i = 0$, la ressource choisie est la même que l'instant de mise à jour précédent, i.e. $cr_{ik} = cr_{i(k-1)}$. L'agent n'a alors plus qu'à générer sa trajectoire intuitive, en résolvant le problème OPT_1, vers cette ressource en évitant les conflits de collision détectés.

Pour chaque agent i, la génération d'une trajectoire intuitive $\widetilde{q}_{ib}(t,\tau_k)$ vers une ressource b, illustrée par la Fig. 4.7, se fait par la résolution de l'algorithme d'optimisation ci-dessous :

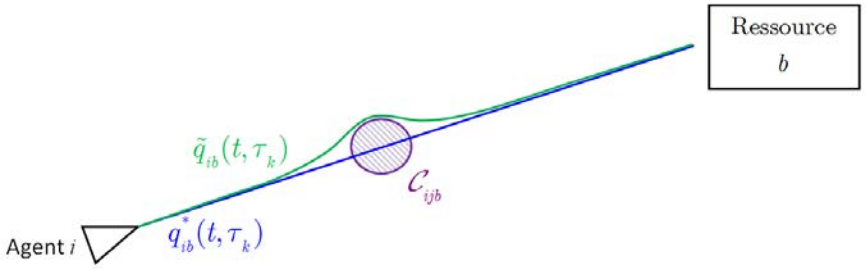


FIGURE 4.7 – Illustration d'une trajectoire intuitive $\widetilde{q}_{ib}(t,\tau_k)$ par rapport à une trajectoire directe $q^*_{ib}(t,\tau_k)$.

OPT$_1$: Considérons le problème suivant qui consiste à déterminer la trajectoire intuitive $\widetilde{q}_{ib}(t,\tau_k)$ de l'agent i à l'instant τ_k vers une ressource b. Pour cela, l'idée est de minimiser la déformation entre les trajectoires intuitive $\widetilde{q}_{ib}(t,\tau_k)$ et directe $q^*_{ib}(t,\tau_k)$ à l'aide du critère

$$\min_{\widetilde{q}_{ib}(t,\tau_k),\widetilde{TT}_i} \int_{\tau_k}^{\tau_k+\widetilde{TT}_i} \|\widetilde{q}_{ib}(t,\tau_k) - q^*_{ib}(t,\tau_k)\| \cdot dt \tag{4.4.4}$$

où $\widetilde{TT}_{ik}$ est le temps de transport associé à la trajectoire intuitive. Notons que ce critère a été choisi (à la place d'une minimisation du temps de transport $\widetilde{TT}_{ik}$) pour améliorer l'évitement de conflits d'une manière décentralisée. En effet, si un critère de minimisation du temps avait été choisi, il ne serait pas nécessairement assuré que l'agent se rapprocherait au maximum de sa trajectoire directe. En effet, il peut y avoir des fluctuations dans l'évitement de conflit comme le montre la Fig. 4.8–(A) à l'aide des doubles flèches.

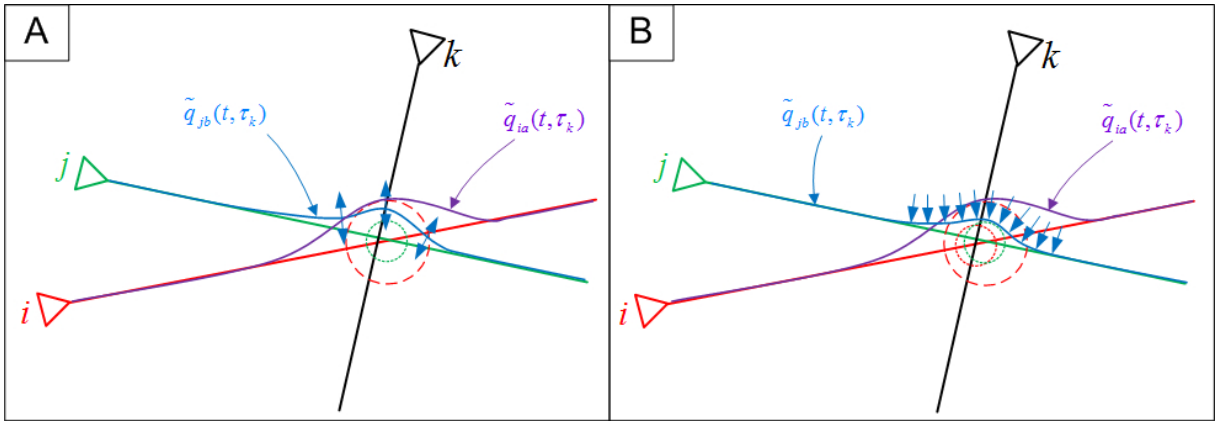


FIGURE 4.8 – Résolution des conflits : (A) minimisation du temps (B) minimisation de la déformation

Ces fluctuations pourront gêner un agent devant éviter le conflit avec un degré supérieur. De ce fait, s'il y a plusieurs autres agents avec des conflits de degrés supérieurs, un effet de répercussion pourrait se produire. Il est utile de mentionner que la génération de la trajectoire intuitive est primordiale puisque la trajectoire finale est contrainte de s'en rapprocher. Par conséquent, une minimisation de la déformation est préférable, permettant de forcer l'agent à se rapprocher de la zone de conflit sans y entrer du fait de la contrainte d'évitement de conflit (voir 4.8–(B)). De plus, ce choix de critère permet tout de même d'avoir un bon temps de transport. En effet, en se rapprochant de la trajectoire directe, le temps de transport est réduit car il se rapproche du temps de transport de la trajectoire directe, qui est une borne inférieure. Pour ce problème d'optimisation, les contraintes suivantes doivent être respectées :

Conditions aux limites : L'agent i planifie à partir de l'instant τ_k de la configuration actuelle q_i avec la vitesse actuelle v_i vers une des configurations cibles q_b à l'instant $\widetilde{T}_{i,fin} = \tau_k + \widetilde{TT}_{ik}$ à vitesse nulle, pour tout $b \in \mathcal{R}_{nl}$:

$$\begin{aligned} \widetilde{q}_{ib}(\tau_k, \tau_k) &= q_i && (4.4.5)\\ \dot{\widetilde{q}}_{ib}(\tau_k, \tau_k) &= v_i && (4.4.6)\\ \widetilde{q}_{ib}(\widetilde{T}_{i,fin}, \tau_k) &= q_b, \quad b \in \mathcal{R}_{nl} && (4.4.7)\\ \dot{\widetilde{q}}_{ib}(\widetilde{T}_{i,fin}, \tau_k) &= 0 && (4.4.8) \end{aligned}$$

Comportement physique de l'agent : La vitesse linéaire est bornée comme suit :

$$\left\| \dot{\widetilde{q}}_{ib}(t, \tau_k) \right\| \leq v_{max}, \forall t \in [\tau_k, \widetilde{T}_{i,fin}] \qquad (4.4.9)$$

Contraintes manufacturière : Cette contrainte assure le respect du cahier des charges fourni par le niveau G.O.R.P. En effet, l'opération n du produit l doit être achevée avant l'échéance odd_{nl}, $\forall b \in \mathcal{R}_{nl}$. Le temps pour achever l'opération ct_{nlb}, défini par l'équation (4.4.1), doit satisfaire l'inégalité suivante :

$$ct_{nlb} \leq odd_{nl}, \quad b \in \mathcal{R}_{nl} \qquad (4.4.10)$$

Évitement des zones de conflit : Cette contrainte assure que les zones de conflits définies par l'ensemble $\mathcal{CF}_{ik}$, fourni par le superviseur, soient évitées. Ceci s'exprime par :

$$\|\widetilde{q}_{ib}(t, \tau_k) - q^*_{jc}(t, \tau_k)\| > cd_{ij} \cdot d_{safe}, \quad \forall t \in I_{ij},\ \forall (j, b, I_{ij}, cd_{ij}) \in \mathcal{CF}_{ik},\ b \in \mathcal{R}_{nl} \qquad (4.4.11)$$

où $c = \widetilde{cr}_{jk}$ est la ressource choisie intuitivement par l'agent j. L'ordre du conflit cd_{ij} permet de régler la taille de la zone à éviter. Ainsi, lorsque plusieurs agents doivent éviter un conflit dans la même zone, ils évitent cette zone différemment car le superviseur fait en sorte que leur degré de

conflit soit différent. Ceci implique que chaque agent dévie sa trajectoire directe autour de la zone de conflit d'une manière différente.

Notons que l'algorithme d'optimisation OPT_1 n'est appliqué que s'il y a des conflits détectés par le superviseur pour l'agent i vers la ressource b. S'il n'y a pas de conflit à éviter, la trajectoire intuitive sera égale à la trajectoire directe (i.e. $\widetilde{q}_{ib}(t,\tau_k) = q^*_{ib}(t,\tau_k)$) comme le montre la Fig. 4.6 avec la condition $\mathcal{CF}_{ik} = \varnothing$.

Remarque 4.3 *Notons que les trajectoires intuitives générées vers chaque ressource représentent approximativement les intentions de l'agent en termes d'évitement de collision global. C'est principalement pour cette raison que l'ordonnancement est appliqué durant cette première étape. En effet, de cette manière, si un agent a beaucoup de conflits de collisions à éviter vers une ressource, il pourra en choisir une autre où il y a peu de conflits voire aucun.*

Remarque 4.4 *Dans la remarque précédente, le terme approximativement est utilisé pour justifier que les trajectoires intuitives ne permettent que d'éviter des trajectoire partielles. En effet, pour chaque conflit, l'intervalle d'évitement I_{ij} assure l'évitement autour du conflit mais l'évitement de collision au voisinage de cet intervalle n'est pas assuré. De plus, chaque agent i n'a aucune connaissance sur la manière dont les autres agents ont évité les conflits, ce qui peut aussi engendrer des répercussions en termes de collisions. Par conséquent une seconde étape de planification est nécessaire pour assurer que la trajectoire planifiée par chaque agent soit sans collision.*

4.4.3 Seconde étape de planification

Suite à la Remarque 4.4, l'étape 2 du planificateur de trajectoire consiste à ajuster la trajectoire intuitive afin d'assurer l'évitement de collisions. Le problème OPT_2 est donc résolu après l'étape 1. Pour cela, tous les voisins doivent se transmettre leur trajectoire intuitive à l'instant τ_k dès que l'étape 1 s'achève. Ensuite, pour chaque agent $i \in \mathcal{A}$, les trajectoires intuitives des voisins $j \in \mathcal{HP}_i$ sont utilisées pour générer la trajectoire finale sans collision, obtenue par la résolution du problème d'optimisation OPT_2.

Pour chaque agent i, la génération de la trajectoire finale sans collision $q_i(t,\tau_k)$ vers la ressource choisie $b = cr_{ik}$, illustrée par la Fig. 4.9, se fait par la résolution de l'algorithme d'optimisation ci-dessous :

$\mathbf{OPT}_2$: Lorsque le problème $\mathbf{OPT}_1$ est résolu, pour chaque agent $i \in \mathcal{A}$, considérons le second problème qui consiste à planifier la trajectoire finale sans collisions $q_i(t,\tau_k)$ à l'instant τ_k. Puisque la ressource a été choisie lors de la première étape, l'objectif est d'arriver le plus tôt possible en minimisant le temps de transport TT_{ik}, qui se traduit par le critère d'optimisation suivant :

$$\min_{q_i(t,\tau_k),T_{i,fin}} TT_i \tag{4.4.12}$$

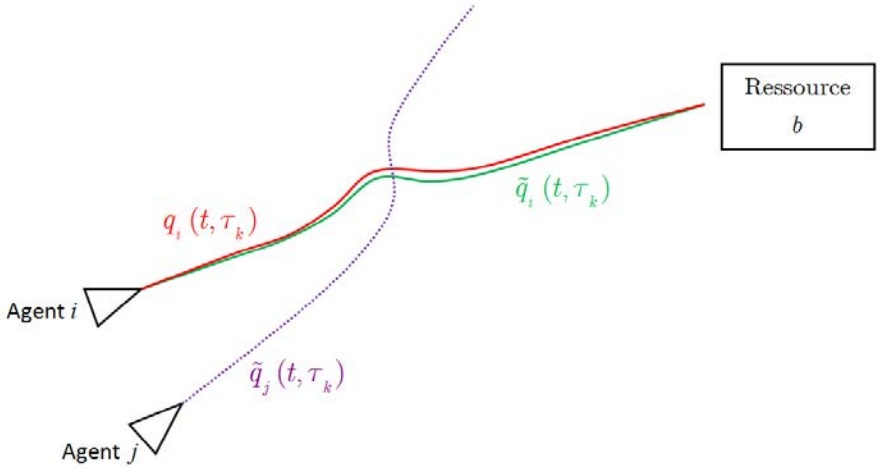


FIGURE 4.9 – Illustration de la trajectoire finale $q_i(t,\tau_k)$ par rapport à la trajectoire intuitive $\widetilde{q}_i(t,\tau_k)$.

Ce problème doit, similairement au précédent, respecter les contraintes physiques (4.4.9), les conditions aux limites (4.4.5) et les contraintes manufacturières (4.4.10). Cependant, au lieu d'éviter les conflits, l'évitement de collisions inter-robots doit être respecté. Ainsi, la contrainte (4.4.11) est remplacée par les deux contraintes couplées suivantes :

- **Déviation maximale permise** : Chaque agent doit rester à un voisinage ξ de sa trajectoire intuitive, se caractérisant par :

$$\|q_i(t,\tau_k) - \widetilde{q}_i(t,\tau_k)\| \quad \leq \quad f(t), \forall t \in [\tau_k, T_{i,fin}] \tag{4.4.13}$$

où la fonction $f(t)$ dépend du voisinage ξ et est conçue telle que :

$$\begin{cases} f(t) = \xi & , \forall t \in [\tau_k, \tau_k + T_p] \\ f(t) = \xi \cdot (1 + Card(\mathcal{HP}_i)/2) & , \forall t \in \left[\tau_k + TT^*_{i(cr_{ik})}, T_{i,fin}\right] \\ f'(t) > 0 & , \forall t \in \left[\tau_k + T_p, \tau_k + TT^*_{i(cr_{ik})}\right] \end{cases} \tag{4.4.14}$$

où $T_p > T_c$ représente l'horizon de faible déviation de la trajectoire finale planifiée $q_i(t,\tau_k)$. Il permet que celle–ci soit contrainte pour être au maximum à une distance ξ de la trajectoire intuitive $\widetilde{q}_i(t,\tau_k)$ sur $t \in [\tau_k, \tau_k + T_p]$. $f'(t)$ représente la dérivée de la fonction f par rapport au temps afin que l'inégalité $f'(t) > 0$ permette que la fonction $f(t)$ soit strictement croissante sur $t \in \left[\tau_k + T_p, \tau_k + TT^*_{i(cr_{ik})}\right]$. $TT^*_{i(cr_{ik})}$ est le temps de transport lié à la trajectoire directe $q^*_{i(cr_{ik})}(t,\tau_k)$ qui est une borne inférieure du temps de transport, i.e. $T_{i,fin} = \tau_k + TT_i > \tau_k + TT^*_{i(cr_{ik})}$ où cr_{ik} est la ressource choisie lors de la première étape de planification. La Fig. 4.10 illustre cette fonction $f(t)$ pour différentes valeur de $Card(\mathcal{HP}_i)$.

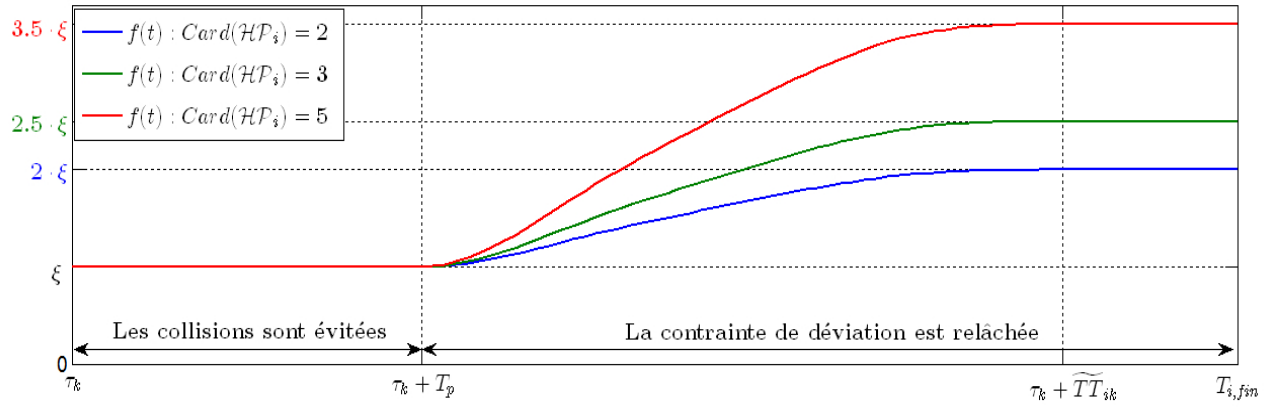


FIGURE 4.10 – Exemple de la fonction $f(t)$ pour un nombre différent d'agents ayant un plus haut niveau de priorité.

- **Évitement de collisions** : Chaque agent doit éviter ses voisins actuels j ayant un niveau de priorité plus haut en considérant à la fois la distance de sécurité d_{safe} et le voisinage ξ, i.e.

$$\|q_i(t,\tau_k) - \widetilde{q}_j(t,\tau_k)\| \quad > \quad d_{safe} + \xi, \qquad \forall j \in \mathcal{HP}_i \tag{4.4.15}$$

  Le voisinage ξ est ajouté pour garantir l'évitement de collisions avec chaque agent $j \in \mathcal{HP}_i$ en considérant l'écart entre les trajectoires intuitives et finales de ceux–ci.

Les deux contraintes couplées (4.4.13) et (4.4.15) assurent que les collisions soient évitées sur l'intervalle $t \in [\tau_k, \tau_k + T_p]$. Notons que ce type de couplage a été utilisé pour de la planification à horizon glissant où la trajectoire planifiée est partielle (i.e. sur l'horizon de planification) [Defoort et al., 2009a]. Cependant, la trajectoire planifiée lors de la seconde étape de notre planificateur n'est pas partielle. C'est pour cela qu'une fonction $f(t)$ a été utilisée. Elle permet de relâcher la contrainte de déviation (4.4.13) pour tout $t > \tau_k + T_p$, permettant de réduire la complexité de l'algorithme. En effet, il peut être difficile de maintenir une faible déviation sur une trajectoire complète. Le choix de la fonction $f(t)$ n'est pas unique et pourrait être modifié dans d'autres applications. Pour notre problème, comme une coordination par leadership est appliquée, il est préférable que la fonction $f(t)$ dépendent du nombre d'agents à éviter, représenté par $Card(\mathcal{HP}_i)$. Cependant, bien que le relâchement de la déviation pourrait engendrer des collisions lorsque $t > \tau_k + T_p$, l'évitement de collision reste assuré car la trajectoire sera remise à jour lors du prochain instant de planification τ_{k+1}. Cette mise à jour sera faite avant que l'horizon de faible déviation ne soit atteint car $T_p > T_c = \tau_{k+1} - \tau_k$. Notons que cette fonction $f(t)$ est définie par (4.4.14) uniquement si $T_p < TT^*_{i(cr_{ik})}$ afin que les différents intervalles de la variable t soient complémentaires. Si $T_p > TT^*_{i(cr_{ik})}$, on prendra $f(t) =$, pour tout $t \in [\tau_k, T_{i,fin}]$ car le couplage pourra facilement être respecté. En effet, $T_p > TT^*_{i(cr_{ik})}$ signifie que l'agent est proche d'arriver à la ressource. De ces propos, le choix de T_p a une grande importance sur l'efficacité de l'évitement de collision.

D'une manière similaire à précédemment, l'algorithme d'optimisation OPT_2 n'est appliqué que s'il y a des collisions à éviter. S'il n'y en a pas, la trajectoire finale sera égale à la trajectoire intuitive comme le montre la Fig. 4.6 avec la condition $\mathcal{HP}_i = \varnothing$. De ce fait, l'agent ayant le plus haut niveau de priorité n'a pas de trajectoire à planifier. Il n'a qu'à utiliser la trajectoire directe donnée par le superviseur. Parallèlement, un agent n'ayant pas de voisin n'aura qu'à éviter les conflits à l'étape 1 en utilisant l'algorithme OPT_1 et sa trajectoire finale sera l'intuitive.

Remarque 4.5 *Il est essentiel de noter que les informations requises pour chaque problème d'optimisation sont partielles, permettant de bien réduire la complexité des calculs. En effet, l'algorithme OPT_1 ne nécessite que des trajectoires **partielles** données par le superviseur et l'algorithme OPT_2 n'a besoin que des trajectoires des agents voisins ayant un niveau de priorité plus haut. Ceci est un élément très important puisque les informations transmises par communication et les informations à traiter par les agents sont réduites.*

Remarque 4.6 *Le choix des horizons T_c et T_p reste ouvert puisqu'ils sont corrélés. Cependant, ce choix mérite quelques réflexions. L'horizon de calcul T_c doit être choisi le plus petit possible pour garantir une bonne réactivité face aux incertitudes mais doit être suffisamment grand pour assurer la résolution des problèmes d'optimisation. L'horizon de faible déviation est nécessairement supérieure à T_c afin d'éviter les blocages entre les contraintes couplées entre les agents. Cependant, il ne doit pas être trop grand pour pouvoir relâcher suffisamment la contrainte de déviation afin de réduire la complexité de l'algorithme.*

4.5 Application de l'optimisation par essaims particulaires (PSO)

4.5.1 Introduction

Les deux algorithmes d'optimisation $\mathbf{OPT}_1$ et $\mathbf{OPT}_2$ proposés, permettant de générer les trajectoires désirées, sont des problèmes non linéaires sous contraintes où les variables de décisions sont continues. Comme les trajectoires sont mises à jour graduellement sur le temps, les algorithmes méta heuristiques sont plus appropriés, permettant une convergence vers des solutions optimales ou proches de l'optimale. L'optimisation par essaims particulaires (PSO), introduite par [Eberhart et al., 1995], est ici utilisée pour résoudre le problème de planification. Cet algorithme est choisi pour sa convergence relativement rapide et son caractère de recherche globale [Saska et al., 2006, Bekrar et al., 2011]. De plus, il a été conçu pour résoudre des problèmes d'optimisation à variables continues, ce qui est notre cas.

Le PSO est basé sur un modèle sociologique où les solutions initiales, appelées particules, évoluent pour trouver la meilleure solution après un certain nombre d'itérations. L'évolution des particules se base sur le déplacement de différentes espèces dans leur environnement respectif comme les groupes d'oiseaux ou de poissons [Eberhart et al., 1995]. Chaque particule à sa propre position, représentant sa solution et sa propre vitesse qui représente son évolution.

L'algorithme du PSO utilisé est présenté sur la Fig. 4.11.

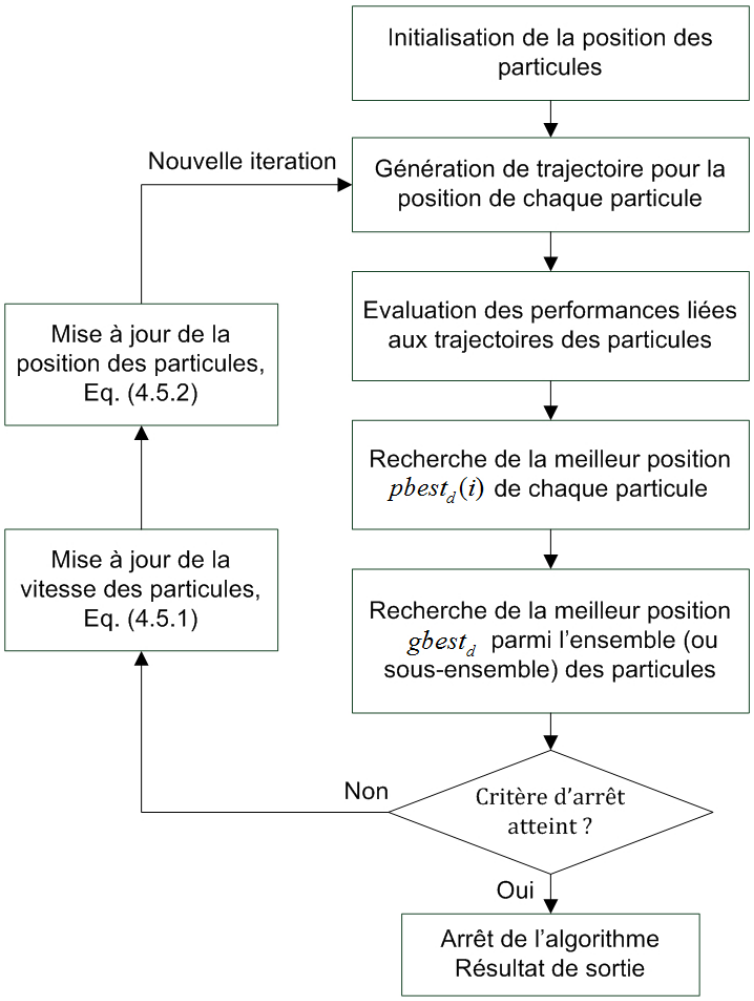


FIGURE 4.11 – Algorithme de l'optimisation par essaims particulaires (PSO) adaptée à notre problème.

Pour le problème considéré, la position d'une particule correspond aux variables nécessaires pour générer une trajectoire vers une ressource. A chaque itération, une trajectoire est générée pour chaque particule et ses performances sont ensuite évaluées selon le critère d'optimisation. Lorsque ces performances sont évaluées, l'algorithme recherche pour chaque particule i la meilleure solution $pbest_d$ obtenue jusqu'à présent puis la meilleure position $gbest_d$ obtenue parmi toutes les particules (ou un sous–ensemble). Notons que le terme "meilleur" se réfère ici aux performances évaluées préalablement.

Si le critère d'arrêt n'est pas atteint, la vitesse de chaque particule est calculée afin de déplacer celles-ci vers une nouvelle position. Si le critère est atteint, l'algorithme s'achève et la position de la particule ayant obtenue les meilleures performances est prise comme solution. Pour notre problème, le critère d'arrêt est le nombre d'itérations, qui dépendra du réglage du PSO présenté dans la section 4.5.3.

L'équation la plus utilisée pour mettre à jour la position de chaque particule pour une nouvelle itération est proposée dans [Hu et Eberhart, 2002] comme suit :

$$v_d(i,k+1) = I_w \cdot v_d(i,k) + b_1 \cdot (pbest_d(i) - x_d(i,t)) + b_2 \cdot (gbest_d - x_d(i,t)) \quad (4.5.1)$$

$$x_d(i,k+1) = x_d(i,k) + v_d(i,k+1) \quad (4.5.2)$$

où $x_d(i,k)$ (resp. $v_d(i,k)$) est la composante d de la position (resp. vitesse) actuelle de la particule i (notons que l'indice i, par abus de notation, représente la particule et non l'agent) ; I_w est un poids représentant l'inertie introduit dans [Hu et Eberhart, 2002]. On remarque que la vitesse est calculée à partir de la meilleure position $pbest_d$ de la particule ainsi que de la meilleure position $gbest_d$ parmi l'ensemble (ou sous-ensemble) des particules. Les coefficients b_1 et b_2 représentent l'accélération de la particule. Ceux–ci sont générés d'une manière aléatoire à chaque itération (i.e. $b_j = c_j \cdot rand(), j = 1, 2$ où c_j sont des constantes positives). Afin d'inclure les contraintes dans le PSO, deux méthodes sont utilisées. La première consiste à utiliser des fonctions de pénalité [Bland, 1993] alors que la seconde permet de préserver la faisabilité des solutions [Parsopoulos et Vrahatis, 2005]. Les fonctions de pénalité sont choisies pour notre approche car elles permettent une application simple du PSO, c'est–à–dire sans avoir à modifier l'algorithme en lui–même. Quelques généralités sur ces fonctions de pénalité sont introduites dans l'annexe D.

Remarque 4.7 *Notons que les fonctions de pénalité ont un avantage important dans l'évitement de conflit. On peut remarquer que lorsqu'un agent a un conflit de degré n, il y aura $n-1$ autres conflits de degré $1, 2, \ldots, n-1$ "à l'intérieur" du conflit de degré n, c'est–à–dire dans la même zone de conflit. Dans un premier temps, ces $n-1$ autres conflits ont été gardés parce que ces conflits ne sont pas nécessairement "à l'intérieur" du conflit de plus haut degré car les conflits ne sont pas des cercles mais un ensemble de cercle dépendant du temps comme nous l'avons montré à l'aide de la Fig. 4.3. De plus, ces $n-1$ conflits ont un rôle de renforcement de contraintes comme le montre la Fig. 4.12 avec trois conflits.*

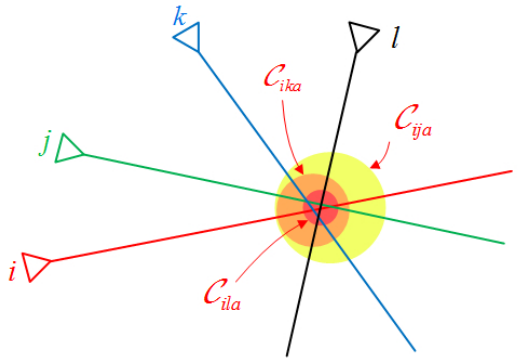


FIGURE 4.12 – Renforcement des contraintes de conflits pour un agent i

En effet, la zone de conflit de plus faible degré sera renforcée d'au plus n fois. Par exemple, la zone dessinée en rouge sur la Fig. 4.12 sera trois fois plus contrainte pour l'agent i que la zone dessinée en jaune. Ceci a une grande utilité lorsque l'on utilise des fonctions de pénalité pour la prise en compte des contraintes dans le problème d'optimisation. En effet, la zone de conflit de degré 1 *sera la plus pénalisée, réduisant ainsi les risques que cette zone ne soit traversée au mauvais moment par la trajectoire intuitive générée. Par conséquent, il est nécessaire de considérer toutes les zones de conflit, même si elles se "chevauchent".*

Rappelons que les trajectoires à planifier sont paramétrisées à l'aide de courbe splines. Ainsi, la position des particules (i.e. variables de décision) est, pour nos problèmes d'optimisation, les points de contrôle P_{ai} liées aux B–splines et le temps de transport TT_{ik}. Dans le chapitre 6, neuf points de contrôles seront utilisés pour définir une trajectoire dont 4 points permettront de respecter les conditions aux limites (voir Eq. (2.4.9)–(2.4.11)) et les cinq points restant seront à déterminer. Par conséquent, les variables de décisions incluent cinq points de contrôle $P_{ai}, a = 2, \ldots, 6$. Notons que les points de contrôle P^*_{ai}, $\tilde{P}_{ai}$ et P_{ai} sont respectivement ceux qui représentent les trajectoires directe $q^*_{ib}(t, \tau_k)$, intuitive $\widetilde{q}_{ib}(t, \tau_k)$ et finale $q_i(t, \tau_k)$. L'algorithme PSO proposé est codé sur Matlab 8.0 et implémenté sur le PC (Intel Core $i7$, $1.80GHz$ avec $8GB$ de RAM). Afin d'adapter le PSO pour nos problèmes, ses paramètres doivent être réglés. Notons que le réglage est présenté pour le problème $\mathbf{OPT}_1$.

4.5.2 Adaptation du PSO pour notre problème

Le réglage des paramètres du PSO, permettant de l'adapter au problème d'optimisation, se fait considérant la génération (i.e. nombre d'itérations), le nombre de particules, la taille de l'espace de recherche et sur les paramètres d'inertie I_w et d'accélération c_1, c_2. D'abord, considérons les trois classes de paramètres utilisés dans notre réglage :

- **Classe 1 :** PSO commun [Eberhart et Shi, 2001]. $I_w \in [0.3, 0.9]$, $c_1 = c_2 = 2.2$
- **Classe 2 :** Réglage de Tréléa [Trelea, 2003]. $I_w = 0.6$, $c_1 = c_2 = 1.7$
- **Classe 3 :** Recommandations de Clerc [Clerc, 1999]. $I_w = 0.729$, $c_1 = c_2 = 1.494$.

Ensuite, l'espace de recherche, c'est–à–dire l'espace sur lequel chaque particule évolue, doit être considéré pour améliorer le taux de convergence du PSO. Pour réduire la taille de cet espace de recherche, il est aussi possible d'ajouter des inégalités valides.

Pour notre réglage, l'espace de recherche est dit "non réduit" (nrSS) lorsque tous les points de contrôle sont à l'intérieur d'un rectangle (parce qu'il est toujours à l'intérieur de l'espace de solutions) qui dépend des points de contrôle initiaux et finaux P^*_{0i}, P^*_{8i} comme le montre la Fig. 4.13 à l'aide du rectangle gris) et lorsque $\widetilde{T}_{i,fin} \in [\|q_i(\tau_k) - q_b\|/v_{max}, odd_{nl}]$. A l'inverse, l'espace de

recherche est "réduit" (rSS) lorsque chaque point de contrôle est dans un rectangle réduit dépendant uniquement des points de contrôle suivant et précédent (voir rectangles de couleur sur la Fig. 4.13) et $\widetilde{T}_{i,fin} \in [T^*_{ik,fin}, odd_{nl} - opt_{nl} - w_{bk} - wp_{ibk}]$. Notons que la réduction de l'espace de recherche des points

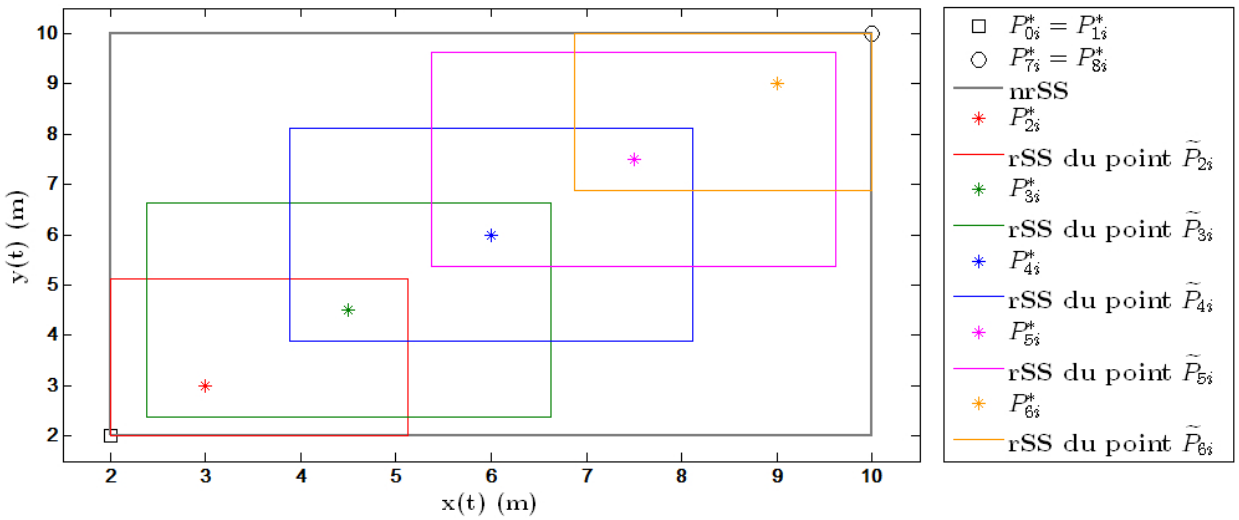


FIGURE 4.13 – Espace de recherche de chaque point de contrôle : (nrSS) non réduit, (rSS) réduit.

de contrôle se base sur le fait que si deux points de contrôle successifs s'éloignent trop, la trajectoire de vitesse autour des ces points va augmenter dû aux propriétés des B–splines, pouvant mener à un dépassement de la vitesse maximales. Pour empêcher ce dépassement, il faut diminuer le temps de transport, ce qui n'est pas optimal en soit. Il est donc préférable d'éviter ce problème en réduisant l'espace de recherche des points de contrôle. De plus, la réduction de l'espace de recherche du temps de transport permet de directement intégrer les deux contraintes liées au temps (4.3.2) et (4.4.10).

Afin d'éviter un conflit de collision (ou une collision avec un voisin pour le problème $\mathbf{OPT}_2$), chaque agent a deux possibilités : réduire sa vitesse ou déformer sa trajectoire dans l'espace par rapport à la trajectoire directe. Chaque possibilité implique une déviation par rapport à la trajectoire directe. Pour améliorer la convergence de l'algorithme, deux inégalités valides sont introduites. Elles permettent d'obtenir un bon compromis entre une réduction de vitesse et une déformation de trajectoire et sont définies par :

$$\|\widetilde{P}_{ai}, P^*_{ai}\| < \epsilon_1 \quad (4.5.3)$$

$$\widetilde{T}_{i,fin} - T^*_{ik,fin} < \epsilon_2 \quad (4.5.4)$$

où ϵ_1 and ϵ_2 sont des constantes petites. L'équation (4.5.3) permet de limiter la distance entre les points de contrôle des trajectoires intuitive et directe alors que l'équation (4.5.4) réduit l'écart entre les temps de transport correspondants à ces trajectoires. Par conséquent, quatre cas sont proposés par rapport à la taille de l'espace de recherche et à la prise en compte des inégalités valides (4.5.3)–(4.5.4).

- **Cas 1 :** Les inégalités (4.5.3)–(4.5.4) ne sont pas considérées et l'espace de recherche est non réduit (nrSS).
- **Cas 2 :** L'espace de recherche n'est pas réduit (nrSS) et les inégalités (4.5.3)–(4.5.4) sont prises en compte.
- **Cas 3 :** L'espace de recherche est réduit (rSS) et les inégalités (4.5.3)–(4.5.4) ne sont pas considérées.
- **Cas 4 :** Les inégalités (4.5.3)–(4.5.4) sont prises en compte et l'espace de recherche est réduit (rSS).

4.5.3 Cas d'étude et réglage des paramètres

Afin de pouvoir régler le PSO, le cas d'étude est le suivant. Considérons trois agents 1, 2 et 3 se dirigeant vers leur ressource respective c, d et b. L'agent 3 a le niveau de priorité le plus bas et doit éviter les conflits avec les deux autres agents $C_{32b} = (2, b, I_{32}, 1)$ et $C_{31b} = (1, b, I_{31} \subset I_{32}, 2)$. Le réglage se fait pour l'agent 3 qui doit générer sa trajectoire intuitive $\widetilde{q}_{3b}(t, \tau_k)$ en évitant les deux zones de conflits données comme le montre la Fig. 4.14-(B). Les résultats comparatifs sont fournis

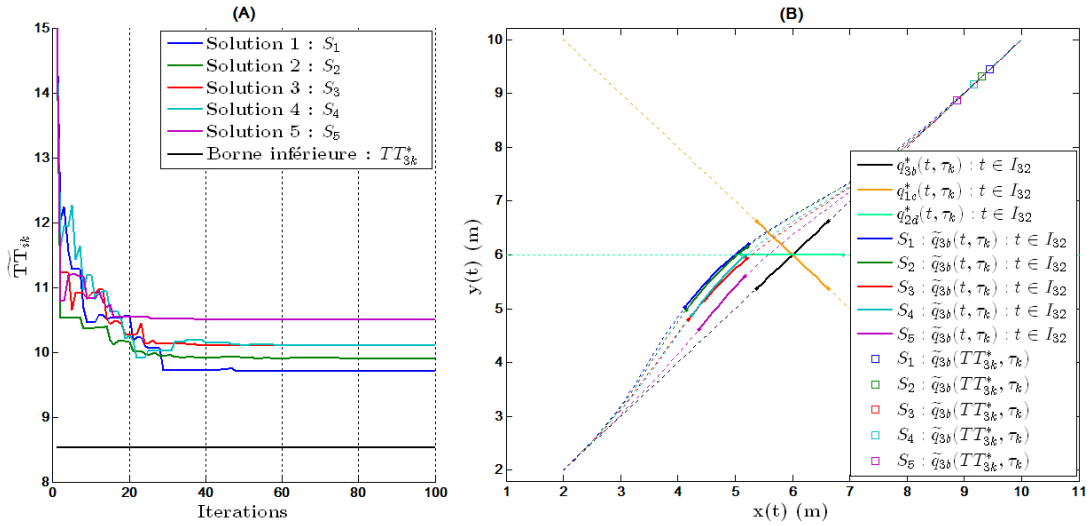


FIGURE 4.14 – (A) Illustration de la rapidité de convergence pour cinq solutions du réglage choisi, (B) Trajectoires liées à ces solutions.

dans le tableau 4.1 où les trois classes de paramètres ainsi que les quatre cas sont testés 50 fois avec des générations et nombres de particules différents. La population initiale est générée aléatoirement. Le temps de calcul pour 40 particules et 50 itérations est de $1.32s$ et celui pour 100 itérations et 80 particules est de $4.85s$, quelques soient le cas et la classe. Dans le tableau 4.1, Gen. représente le nombre d'itérations et Sw. le nombre de particules. Les valeurs données dans ce tableau sont les temps

de transport $\widetilde{TT}_{3k} = \widetilde{T}_{3,fin}$ (où $\tau_k = 0$) correspondant à la trajectoire intuitive $\widetilde{q}_{3b}(t, \tau_k)$. Une borne inférieure de ce temps de transport est le temps correspondant à la trajectoire directe $TT^*_{3k} = 8.52s$ où il n'y a pas de conflit de collision à prendre en compte. En guise de comparaison avec un autre algorithme, une programmation séquentielle quadratique (SQP) a aussi été testée. Les résultats de cette SQP sont moins bons que pour le PSO ($TT_{3k} = 13.45s$ en utilisant SQP) avec un temps de calcul de $4853s$. Dans ce cas, une solution optimale, pouvant être obtenue à l'aide de méthode de résolution exacte (par exemple, la programmation non linéaire mixte), pourrait être difficile à obtenir rapidement, ce qui n'est pas souhaitable pour l'implémentation.

Tableau 4.1 – Comparaison entre les quatres cas et le trois classes de paramètres du PSO

	Gen.	Sw.	Classe 1			Classe 2			Classe 3		
			Min	Moy.	Max	Min	Moy.	Max	Min	Moy.	Max
Cas 1	50	40	9.81	12.54	17.47	9.81	11.82	13.63	9.59	12.03	17.27
	50	80	10.07	12.05	17.93	9.3	11.54	14.18	9.75	11.65	13.57
	100	40	10.02	12.68	20.67	9.63	12.15	16.77	10.09	12.3	24.52
	100	80	9.96	12.15	13.97	9.62	11.58	14.79	9.82	12.01	13.63
Cas 2	50	40	10.03	12.62	18.76	9.70	12.04	14.29	9.5	12.05	17.27
	50	80	9.77	12.13	14.78	9.69	12.02	14.01	9.57	11.39	13.68
	100	40	9.78	11.71	13.73	9.71	12.98	20.76	9.66	11.94	14.08
	100	80	9.75	12.36	14.21	9.61	11.41	13.47	9.74	11.67	14.25
Cas 3	50	40	9.82	11.63	13.72	9.42	11.16	13.41	9.73	11.16	13.17
	50	80	9.98	12.18	13.67	9.46	10.52	12.93	9.72	11.04	13.24
	100	40	9.95	11.58	13.09	9.75	11.76	13.54	9.70	11.36	13.29
	100	80	9.76	11.41	13.11	9.61	11.28	12.97	9.67	11.52	12.98
Cas 4	50	40	9.69	11.57	13.69	9.49	11.48	13.62	9.70	11.22	12.99
	50	80	9.65	11.43	13.64	9.37	11.24	12.94	9.57	10.89	12.94
	100	40	9.87	11.62	13.21	9.47	11.49	13.19	9.48	10.77	13.17
	100	80	9.46	10.91	12.98	**9.35**	**10.58**	**12.7**	9.43	10.49	12.98

Les résultats écrit en couleur noir sont ceux qui respectent les contraintes. Inversement, les résultats sont d'une autre couleur lorsque sur une partie des tests, les contraintes ne sont pas respectées. Dans ces cas de non respect des contraintes, le PSO aurait besoin de plus d'itérations afin de les respecter. Les résultats sont de couleur rouge, vert, bleu quand plus de 25%, 12%, 2% des tests n'ont pas respecté les contraintes, respectivement. Le meilleur réglage du PSO pour notre approche, mis en **gras** sur le tableau 4.1, est pour la classe 2 avec 100 itérations, 80 particules et pour le cas 4. Pour ce meilleur réglage, 5 solutions des 50 tests sont présentés sur la Fig. 4.14 où les taux de convergence sont illustrés dans 4.14–(A) et les trajectoire intuitives liées à ces solutions dans 4.14–(B). Notons que les trajectoires complètes sont tracées en pointillées pour souligner la partie de la trajectoire (ligne pleine) qui correspond à l'intervalle des conflits à éviter I_{32}. De plus, la position de l'agent 3 à l'instant final TT^*_{3k} pour chaque solution est indiqué sur cette même figure. Il permet de montrer que la solution

en bleu donne le meilleur résultat. On remarque ainsi que le meilleur moyen d'éviter le conflit est de déformer plutôt que de réduire la vitesse. Le meilleur réglage sera gardé pour les résultats fournis dans le chapitre 6 (i.e. 80 particules, 100 itérations, $I_w = 0.6$, $c_1 = c_2 = 1.7$ où l'espace de recherche est réduit (rSS) et les inégalités valides (4.5.3)–(4.5.4) sont prises en compte). La même méthode de réglage a été appliquée pour le problème d'optimisation $\mathbf{OPT}_2$. Le meilleur réglage est le même que pour le problème $\mathbf{OPT}_1$ car ces problèmes sont similaires en termes de contraintes et ont les mêmes variables de décision.

Remarque 4.8 *Les réglages proposés ne sont pas les seuls possibles. En effet, d'autres réglages, plus récents, pourraient être testés comme ceux proposés dans [Chatterjee et Siarry, 2006] ou [Nickabadi et al., 2011]. Cependant, l'étude montre la robustesse de l'algorithme puisque les résultats numériques s'avèrent peu dépendants du réglage des paramètres.*

4.6 Conclusion

Dans ce chapitre, une nouvelle architecture, permettant aux agents d'effectuer une opération de leur produit, a été introduite. L'architecture est, contrairement à celle du chapitre 3, hybride dû à l'ajout d'un superviseur qui assiste les agents dans leur tâche. Son intérêt réside principalement dans la gestion des conflits. Ainsi, les agents seront moins myopes puisque le superviseur leur donne des informations régulièrement. Par conséquent, l'objectif des agents consiste à planifier une trajectoire sans collision vers une ressource et à choisir la ressource lorsque le superviseur le permet.

Le superviseur a deux rôles. D'une part, il doit résoudre les conflits d'ordonnancement permettant d'autoriser les agents à choisir une ressource et les conflits de ressource qui donne l'ordre d'arrivée de plusieurs agents voisins à une ressource. D'autre part, il est utilisé pour détecter les conflits de collision que les agents devront résoudre, permettant ainsi de réduire la myopie des agents. Notons que tous les conflits sont gérés par un mécanisme de priorité qui utilise les performances individuelles des agents.

En termes de planification de trajectoire, la mise à jour se fait d'une manière itérative et en parallèle. Les agents doivent attendre un certain nombre de mises à jour avant de pouvoir réordonnancer leur produit. La planification se divise en deux étapes. La première consiste à éviter, d'un point de vue global, les zones de conflit fournies par le superviseur. La seconde étape est locale où les agents doivent éviter leurs voisins ayant un niveau de priorité plus haut.

Puisqu'à chaque mise à jour, plusieurs trajectoires doivent être planifiées, une optimisation par essaims particulaires est proposée afin de réduire les temps de calcul sans s'éloigner de la solution optimale. Un réglage de paramètres est proposé et testé sur un cas d'étude. Les principaux résultats seront donnés dans le chapitre 6 avec l'analyse correspondante. Une comparaison avec l'approche du chapitre 3 sera faite afin de souligner les avantages du superviseur mais aussi l'utilité du PSO dans l'obtention de trajectoires proches de l'optimale.

Chapitre 5

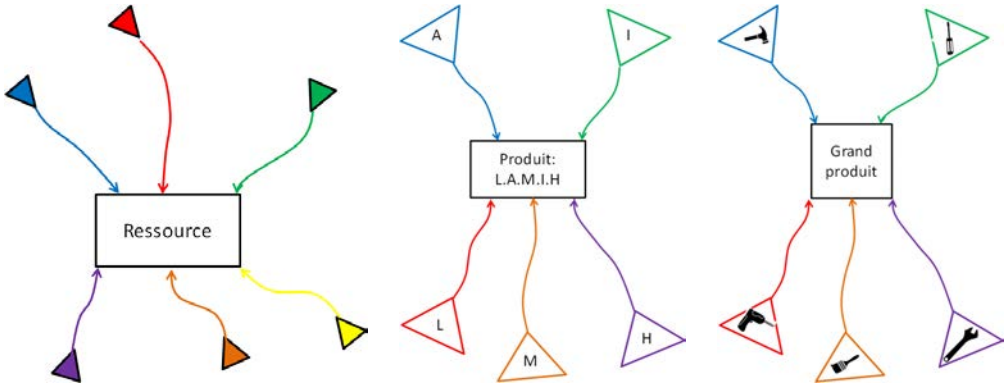
Navigation par commande coopérative et rendez–vous

5.1 Introduction

Dans le chapitre 1, un ensemble de travaux sur la commande coopérative permettant la navigation d'agents a été présenté. Nous avons vu que la commande coopérative inclut le problème de rendez–vous avec ou sans meneur et le contrôle de formation. A l'aide du tableau 1.1, nous avons aussi vu le manque de travaux sur la commande coopérative avec une convergence à temps fixe.

Dans ce chapitre, la notion de convergence à temps fixe va être appliquée au problème du rendez–vous pour une flottille d'agents non holonomes. L'objectif est de faire converger l'ensemble des agents vers la configuration d'un meneur immobile en un temps fixe, c'est–à–dire en un temps fini qui est indépendant des configurations initiales des agents. Cependant, il est primordial de donner l'intérêt de cette stratégie pour des applications manufacturières.

Cet intérêt est divisé en deux points. D'une part, il convient de noter que la convergence à temps fixe permet de prendre en compte un cahier des charges. En effet, ce type de convergence permet aux agents d'atteindre un objectif, peu importe où ils se situent dans l'environnement manufacturier, avant une certaine échéance, donnée aux agents et préalablement calculée par le système de production. D'autre part, le problème du rendez–vous vers un meneur immobile à un grand intérêt en soi. En effet, en considérant que le meneur est virtuel, ce problème de rendez–vous peut être assimilé au rendez–vous d'agents à une position précise dans l'environnement de production. Ainsi, on pourrait par exemple permettre aux agents d'atteindre une ressource au même moment pour diverses raisons comme le montre la Fig. 5.1–(a). Une de ces raisons pourrait être par exemple l'assemblage d'un produit à une ressource comme le montre la Fig. 5.1–(b) où le produit $L.A.M.I.H$ est formé à partir des pièces L, A, M, I et H. D'autres applications manufacturières peuvent aussi nécessiter un rendez–vous comme pour les ateliers à projet (voir chapitre 1) où les AGVs sont des machines avec des outils. Ainsi, pour



(a) (b) (c)

FIGURE 5.1 – Application du problème de rendez-vous dans un système de production.

la fabrication d'un produit de grande taille, certains opérations sur le produit pourraient avoir besoin de plusieurs outils différents comme illustré sur la Fig. 5.1–(c).

La suite de ce chapitre se déroule de la manière suivante. D'abord, des préambules sur les notations utilisées et sur des notions utiles au développement théorique présenté seront donnés. Les notions présentées concernent d'une part la théorie des graphes, permettant de représenter les communications inter–agents et d'autre part la stabilité à temps fixe, nécessaire pour le problème traité dans ce chapitre. Ensuite, nous formulerons le problème de rendez–vous d'une flottille d'agents en présentant le modèle non holonome ainsi que la transformation de celui–ci en forme chaînée. En effet, comme l'algorithme de navigation est ici centré sur la commande, il est préférable que le modèle dynamique des agents soit suffisamment réaliste, c'est–à–dire un modèle intégrant les contraintes de non holonomie. Avant de présenter le protocole de commande, nous proposerons des théorèmes permettant de faire converger, en temps fixe, un agent ayant un modèle mis sous forme chaînée. Enfin, le protocole de commande en temps fixe d'une flottille d'agents non holonomes sera présenté. Ce protocole nécessite plusieurs commutations, qui ne dépendent pas des conditions initiales et qui sont connus, pour pouvoir intégrer la non holonomie. Ce chapitre s'achèvera par une conclusion.

5.2 Préambules

Avant d'introduire la théorie des graphes, nous aimerions rappeler quelques notations utiles au développement proposé dans ce chapitre

Notations : Pour une matrice carrée $P \in \mathbb{R}^{N\times N}$, $P \succ \mathbf{0}$ (resp. $P \prec \mathbf{0}$) signifie que la matrice P est définie positive (resp. négative). $\lambda_{min}(P)$ est la plus petite valeur propre de la matrice P.

Soit $\text{diag}(a_1, a_2 \dots a_{N-1}, a_N)$ la matrice diagonale définie par $\begin{pmatrix} a_1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & a_N \end{pmatrix} \in \mathbb{R}^{N\times N}$. Pour un vecteur $x \in \mathbb{R}^N$ donné, $|x|$ (resp. $||x||$) désigne la norme 1 (resp. 2 ou euclidienne) de x.
Pour $x = (x_1, \dots, x_N)^T \in \mathbb{R}^N$ et $b \geq 0$, définissons $\lfloor x \rceil^b = \left(sign\,(x_1)\,|x_1|^b, \dots, sign\,(x_N)\,|x_N|^b\right)^T$.
Soit $D^*f(t)$ la dérivée supérieure droite de Dini d'une fonction $f(t)$, i.e. $D^*f(t) = \lim_{h\to 0^+} \sup \frac{f(t+h)-f(t)}{h}$. Rappelons que la dérivée de Dini est une généralisation de la dérivée d'une fonction.

Théorie des graphes

Afin de résoudre le problème de rendez–vous et modéliser l'échange d'informations entre les agents, la théorie des graphe est rappelée ci–dessous.

Considérons un groupe de N systèmes, appelés agents. La topologie de communication parmi tous les agents peut être représentée par un graphe noté $\mathcal{G}$ qui consiste à un ensemble de noeuds $\mathcal{V} = \{1, 2, \cdots, N\}$ et un ensemble de liens $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. Notons que chaque noeud de l'ensemble $\mathcal{V}$ correspond à un agent i et chaque lien $(i, j) \in \mathcal{E}$ correspond à un lien de communication de l'agent i vers l'agent j.

La topologie du graphe $\mathcal{G}$ est représentée par la matrice pondérée $A = (a_{ij}) \in \mathbb{R}^{N\times N}$, appelée matrice adjacente qui est donnée par $a_{ij} > 0$ si $(j, i) \in \mathcal{E}$ et $a_{ij} = 0$ sinon. La matrice Laplacienne du graphe $\mathcal{G}$ est définie par $L = (l_{ij}) \in \mathbb{R}^{N\times N}$ où $l_{ii} = \sum_{j=1,\, j\neq i}^{N} a_{ij}$ et $l_{ij} = -a_{ij}$ pour $i \neq j$. En guide d'exemple, on peut voir sur les Fig. 5.2 et 5.3 deux graphes de communication avec leur matrice Laplacienne respective. Un graphe $\mathcal{G}$ est dit connecté si et seulement si sa matrice Laplacienne a une seule valeur propre nulle avec son vecteur propre associé $\mathbf{1_N} = (1, 1, \dots, 1)^T \in \mathbb{R}^N$ [Ren et al., 2005].

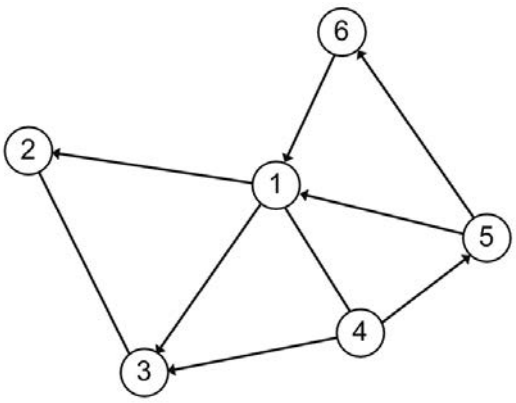
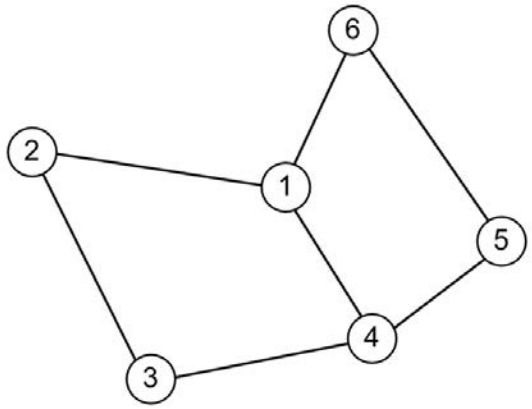


FIGURE 5.2 – Exemple de graphe de communication dirigé.



$$L = \begin{pmatrix} 3 & -1 & 0 & -1 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ -1 & 0 & -1 & 3 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & 0 & -1 & 2 \end{pmatrix}$$

FIGURE 5.3 – Exemple de graphe de communication non dirigé.

Le graphe de communication peut être dirigé ou non et fixe ou commutant. Le graphe est dirigé (cf. Fig. 5.2) si les communications sont unidirectionnelles entre les agents, i.e. $\exists i, j \in \mathcal{V}, (i,j) \in \mathcal{E} \Rightarrow (j,i) \notin \mathcal{E}$. Pour les graphes dirigés, la matrice adjacente n'est pas symétrique. Le graphe de communication est non dirigé (cf. Fig. 5.3) si les communications entre les agents se font dans les deux sens, i.e. $\forall i, j \in \mathcal{V}, (i,j) \in \mathcal{E} \Leftrightarrow (j,i) \in \mathcal{E}$. De plus, pour les graphes non dirigés, la matrice adjacente A est symétrique et la matrice Laplacienne est symétrique définie positive. Lorsque le graphe est fixe, la topologie de communication ne varie pas au cours du temps. A l'inverse, le graphe est commutant si les liens de communication entre les agents changent au cours du temps comme on peut le voir sur la Fig. 5.4.

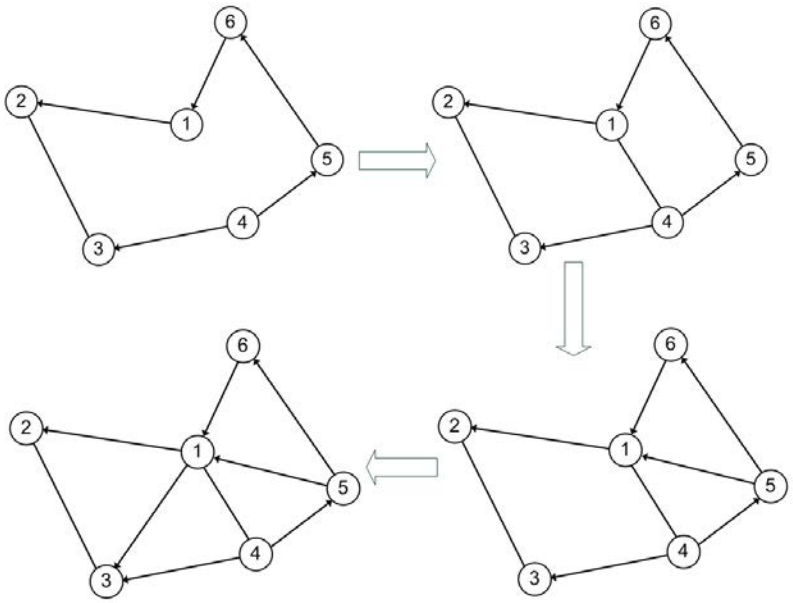


FIGURE 5.4 – Exemple de graphe de communication dirigé et commutant.

Pour ce chapitre, le graphe de communication est fixe et non dirigée, c'est–à–dire que les communications se font dans les deux sens et ne varient pas au cours du temps.

Stabilité à temps fixe

La stratégie de commande proposée dans ce chapitre se base sur la stabilité à temps fixe. Il est donc préférable de la rappeler brièvement. Considérons le système suivant

$$\begin{cases} \dot{x}(t) & \in & F(t, x(t)) \\ x(0) & = & x_0 \end{cases} \tag{5.2.1}$$

où $x \in \mathbb{R}^n$ est l'état du système, $F : \mathbb{R}^+ \times \mathbb{R}^n \to \mathbb{R}^n$ est une fonction semi-continue à valeur convexe, telle que l'ensemble $F(t, x(t))$ soit non vide pour chaque $(t, x) \in \mathbb{R}^+ \times \mathbb{R}^n$ et $F(t, 0) = 0$ pour $t > 0$. Les solutions du système (5.2.1) sont comprises au sens de Filippov [Fillippov, 1988].

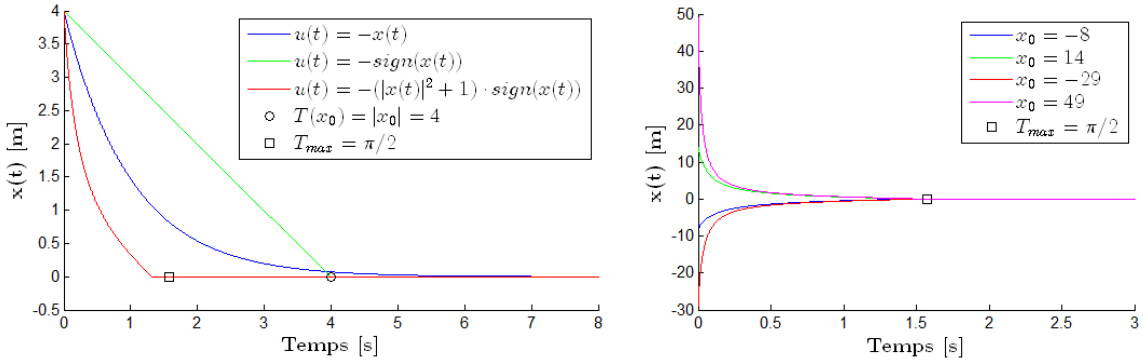
Définition 5.1 *[Bhat et Bernstein, 2005] L'origine du système* (5.2.1) *est globalement un équilibre à temps fini s'il existe une fonction $T : \mathbb{R}^n \mapsto \mathbb{R}^+$ telle que pour tout $x_0 \in \mathbb{R}^n$, la solution $x(t, x_0)$ du système* (5.2.1) *est définie, stable au sens de Lyapunov et que $x(t, x_0) \in \mathbb{R}^n$ pour $t \in [0, T(x_0))$ et $\lim_{t \to T(x_0)} x(t, x_0) = 0$. $T(x_0)$ correspond au temps de convergence.*

Définition 5.2 *[Polyakov, 2012b] L'origine du système* (5.2.1) *est globalement un équilibre à temps fixe s'il est globalement stable à temps fini et le temps de convergence $T(x_0)$ est bornée par un réel positif $T_{max} > 0$, i.e. $T(x_0) \leq T_{max}$ pour tout $x_0 \in \mathbb{R}^n$.*

La stabilisation à temps fixe à l'origine peut être démontrée à l'aide du plus simple système de contrôle scalaire $\dot{x}(t) = u(t)$ (simple intégrateur), où $x \in \mathbb{R}$ est l'état du système et $u \in \mathbb{R}$ est son entrée. Si le retour d'état $u = -\operatorname{sign}(x)$ est appliqué, alors le système en boucle fermée est stable en temps fini avec le temps de stabilisation $T(x_0) = |x_0|$. Il convient de noter que le temps de convergence est dépendant des conditions initiales et est non borné. Le protocole de commande à temps fixe issu de [Polyakov, 2012b] est de la forme $u = -(|x|^2 + 1) \operatorname{sign}(x)$. Ceci garantit une convergence à temps fini indépendamment des conditions initiales, à savoir une convergence à temps fixe où $T(x_0) \leq T_{max} = \frac{\pi}{2}$. Pour illustrer ces propos, deux figures sont proposées. La Fig. 5.5–(a) donne une comparaison entre les convergences asymptotique, à temps fini et à temps fixe pour le système $\dot{x}(t) = u(t)$. La Fig. 5.5–(b) montre l'indépendance par rapport aux conditions initiales du protocole de commande proposé dans [Polyakov, 2012b] pour ce système.

Lemme 5.1 *[Polyakov, 2012b] Supposons qu'il existe une fonction définie continuement dérivable et radialement non bornée $V : \mathbb{R}^n \to \mathbb{R}^+$ telle que*

$$D^*V(x) \leq -\left(aV^p(x) - bV^q(x)\right)^k \quad \textit{pour} \quad x \neq 0$$



(a) (b)

FIGURE 5.5 – Exemples pour le système $\dot{x}(t) = u(t)$: (a) comparaison entre convergence asymptotique, à temps fini et à temps fixe (b) Indépendance par rapport aux conditions initiales du protocole de commande à temps fixe

avec $a, b, p, q, k > 0$, $pk < 1$ *et* $qk > 1$. *Alors, l'origine de l'inclusion différentielle* (5.2.1) *est globalement stable à temps fixe avec un temps de convergence estimé par*

$$T(x_0) \leq T_{\max} = \frac{1}{a^k(1-pk)} + \frac{1}{b^k(qk-1)}$$

En affinant ce lemme, on peut obtenir un résultat moins conservatif à l'aide du lemme ci–dessous.

Lemme 5.2 *[Parsegov et al., 2012] Supposons qu'il existe une fonction définie continuement dérivable et radialement non bornée* $V : \mathbb{R}^n \to \mathbb{R}^+$ *telle que*

$$D^*V(x) \leq -aV^p(x) - bV^q(x) \quad \textit{pour} \quad x \neq 0$$

où $a, b > 0$, $p = 1 - \frac{1}{\mu}$, $q = 1 + \frac{1}{\mu}$ *et* $\mu \geq 1$. *Alors, l'origine de l'inclusion différentielle* (5.2.1) *est globalement stable à temps fixe avec un temps de convergence estimé par*

$$T(x_0) \leq T_{\max} = \frac{\pi\mu}{2\sqrt{ab}}$$

5.3 Formulation du problème

Considérons un groupe de N agents homogènes, ayant les mêmes caractéristiques. Chaque agent n ($n \in \{1, \ldots, N\}$) est de type unicycle, représenté sur la Fig. 5.6.

Avant d'introduire la formulation du problème de rendez–vous à temps fixe d'une flottille d'agents non holonomes, il est préférable de donner quelques détails sur la non holonomie et sur les caractéristiques des robots de type unicycle considérés. Pour la plupart des systèmes mécaniques, des contraintes de position et/ou de vitesse sur les masses du système en mouvement doivent être satisfaites. Un système

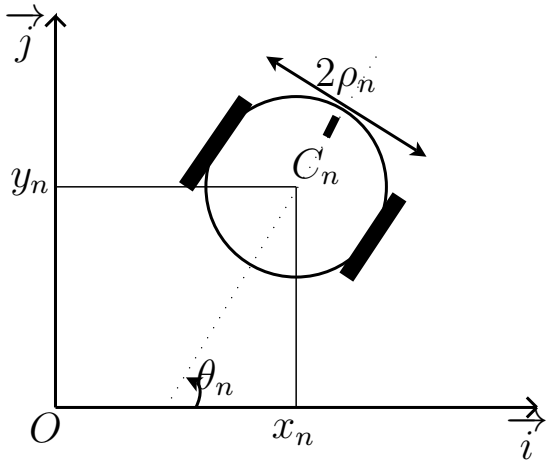


FIGURE 5.6 – Robot mobile de type unicycle.

est dit non holonome en présence de contraintes non intégrables. En effet, pour ces systèmes, il est impossible de les intégrer en les éliminant par changement de variables approprié à l'aide de relations algébriques sur les paramètres de la configuration. Plus de détails sur la non holonomie ainsi que sur la modélisation des robots sont rassemblés dans [Defoort, 2007].

Pour chaque agent (robot) n, le centre des masses (centre de gravité) se situe au centre géométrique C_n du corps du robot. Chaque agent a deux roues directrices fixées sur l'axe passant par C_n et une roue libre orientable et centrée. Les deux roues fixes sont séparées d'une distance de $2\rho_n$ et sont indépendamment contrôlées à l'aide d'un actionneur (moteur à courant continu). La roue libre est passive et empêche le robot de basculer lorsqu'il se déplace sur le plan horizontal.

Dans ce chapitre, il est supposé que le mouvement de la roue libre peut être ignoré dans la dynamique de l'agent. Ainsi, le robot peut être modélisé en considérant les deux roues motrices. Notons que l'on pourrait assimilé ce type de modélisation à des robots ayant des chenilles qui remplacent la roue libre, seulement si les frottements engendrés par les chenilles sont négligeables. Le centre de masse C_n, dont les coordonnées sont (x_n, y_n) pour chaque agent n, se situe à l'intersection d'une ligne passant à travers le milieu du véhicule et l'axe des deux roues motrices.

Dans les chapitres précédents, le modèle dynamique des robots était simple car l'algorithme de navigation se focalisait sur la planification de trajectoire. Ici, l'algorithme de navigation est directement centré sur la commande de bas niveau. Le modèle dynamique des agents doit donc être suffisamment réaliste, c'est–à–dire qu'il doit prendre en compte les contraintes de non holonomie du robot. Par conséquent, la configuration de chaque robot est de type non holonome, décrite par l'état :

$$q_n(t) = (x_n(t), y_n(t), \theta_n(t))^T$$

où $\theta_n(t)$ est l'orientation du robot.

Les contraintes de non holonomie imposent des conditions de roulement sans glissement qui sont ici décrites par

$$\psi^T(q_n)\dot{q_n} = 0 \quad \text{avec} \quad \psi^T(q_n) = \begin{pmatrix} -\sin\theta_n & \cos\theta_n & 0 \end{pmatrix}$$

L'équation cinématique peut donc être définie par

$$\dot{q}_n(t) = f(q_n(t), u_n(t)) \tag{5.3.1}$$

où le champ de vecteur $f : \mathbb{R}^3 \times \mathbb{R}^2 \to \mathbb{R}^3$ et les entrées u_n sont définis par

$$\begin{cases} f(q_n(t), u_n(t)) &= \begin{pmatrix} \cos\theta_n(t) & 0 \\ \sin\theta_n(t) & 0 \\ 0 & 1 \end{pmatrix} u_n(t) \\ u_n(t) &= (v_n(t), w_n(t))^T \end{cases}$$

$v_n(t)$ et $w_n(t)$ sont les vitesses linéaires et angulaires, respectivement. Notons que ce modèle est appelé unicycle car son modèle (5.3.1) est identique au modèle d'une roue verticale seule [Defoort, 2007].

Pour le problème de rendez–vous, un meneur, pouvant être virtuel, est supposé immobile à la configuration $q_0 = (x_0, y_0, \theta_0)^T$. De plus, l'état q_0 du meneur est supposé être disponible uniquement pour une partie de la flottille (et non la totalité).

Par conséquent, nous considérons dans ce chapitre un groupe de $N+1$ agents, incluant les N agents mobiles et un agent meneur fixe, désigné par l'indice 0. La topologie de communication est représentée par le graphe $\bar{\mathcal{G}}$, dépendant du graphe $\mathcal{G}$ entre les N agents mobiles et des communications du meneur vers les suiveurs. La topologie du graphe $\bar{\mathcal{G}}$ est décrite par la matrice pondérée $H = L + D \in \mathbb{R}^{N\times N}$ où $D = diag(a_{10}, \ldots, a_{N0})$. De plus, $a_{i0} > 0$ lorsque l'état désiré (i.e. la configuration du meneur) est disponible pour l'agent i, sinon $a_{i0} = 0$. Rappelons que L est la matrice Laplacienne décrivant la topologie du graphe $\mathcal{G}$ entre les N agents suiveurs. Un exemple de graphe de communication est donné sur la Fig. 5.7.

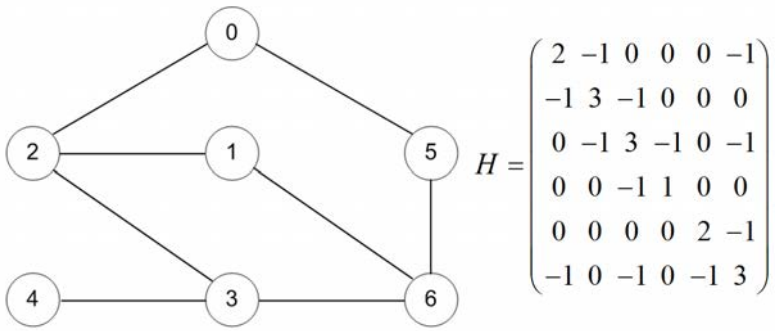


FIGURE 5.7 – Exemple de graphe de communication non dirigé.

L'objectif de ce chapitre est de concevoir un protocole de commande distribuée u_n $(n = 1, \ldots, N)$, basé sur les informations locales disponibles, tel que le problème de rendez–vous vers un meneur soit résolu en un temps fixe donné, noté T_{max}, i.e.

$$q_n(t) = q_0, \qquad \forall t \geq T_{max}, \qquad \forall n \in \{1, \cdots, N\} \tag{5.3.2}$$

Pour résoudre le problème (5.3.2), l'hypothèse suivante est nécessaire.

Hypothèse 5.1 *Il est supposé que le graphe de communication $\mathcal{G}$ est connecté et qu'au moins un agent i a l'information sur l'état du meneur, c'est–à–dire qu'il y ait au moins un $a_{i0} > 0$.*

Avant de concevoir le protocole permettant de résoudre le problème de rendez–vous vers un meneur, la transformation suivante est proposée

$$\left\{ \begin{array}{lcl} \xi_{n1} & = & \theta_n \\ \xi_{n2} & = & x_n \sin\theta_n - y_n \cos\theta_n \\ \xi_{n3} & = & x_n \cos\theta_n + y_n \sin\theta_n \\ \nu_{n1} & = & w_n \\ \nu_{n2} & = & v_n - \xi_{n2} w_n \end{array} \right. , \qquad n \in \{1, \cdots, N\} \tag{5.3.3}$$

A l'aide de la transformation (5.3.3), le système (5.3.1) peut être mis sous forme chaînée de la manière suivante

$$\left\{ \begin{array}{lcl} \dot{\xi}_{n1} & = & \nu_{n1} \\ \dot{\xi}_{n2} & = & \xi_{n3}\nu_{n1} \\ \dot{\xi}_{n3} & = & \nu_{n2} \end{array} \right. , \qquad n \in \{1, \cdots, N\} \tag{5.3.4}$$

L'objectif de commande (5.3.2) est donc équivalent à concevoir un protocole de commande distribuée $\nu_n = (\nu_{n1}, \nu_{n2})^T$ $(n = 1, \ldots, N)$ utilisant les informations des agents voisins telles que l'état $\xi_n = (\xi_{n1}, \xi_{n2}, \xi_{n3})^T \in \mathbb{R}^3$ suit, en un temps désiré T_{max}, l'état du meneur $\xi_0 = (\xi_{01}, \xi_{02}, \xi_{03})^T \in \mathbb{R}^3$ défini par

$$\left\{ \begin{array}{lcl} \xi_{01} & = & \theta_0 \\ \xi_{02} & = & x_0 \sin\theta_0 - y_0 \cos\theta_0 \\ \xi_{03} & = & x_0 \cos\theta_0 + y_0 \sin\theta_0 \end{array} \right.$$

5.4 Stabilisation en temps fixe d'un système sous forme chaînée

Avant de présenter le protocole de commande permettant le rendez–vous vers un meneur à temps fixe, il est nécessaire de présenter la stabilisation en temps fixe d'un système chaîné de la forme

suivante :

$$\begin{array}{lll} (\sum_1) & \dot{z}_1 = & \upsilon_1 \\ & & \\ (\sum_2) & \begin{array}{l} \dot{z}_2 = \\ \dot{z}_3 = \end{array} & \begin{array}{l} z_3\upsilon_1 \\ \upsilon_2 \end{array} \end{array} \tag{5.4.1}$$

où $z = (z_1, z_2, z_3)^T \in \mathbb{R}^3$ désigne l'état du système et $\upsilon = (\upsilon_1, \upsilon_2)^T \in \mathbb{R}^2$ son entrée. La dynamique du système (5.4.1) est divisée en deux sous-systèmes : une dynamique d'intégrateur simple $\sum_1$ ainsi qu'une dynamique du second ordre couplée $\sum_2$. A cause de la contrainte de non holonomie, une des difficulté repose sur le fait que la stabilisation des variables (z_1, z_3) mène à une mise à zéro au coté droit de la seconde équation de (5.4.1). Par conséquent, la variable z_2 ne peut plus converger vers zéro. Le système (5.4.1) ne respecte pas les conditions nécessaires sur l'existence d'un retour d'état continu statique garantissant la stabilité asymptotique à l'équilibre [Brockett et al., 1983]. Une stratégie par commutation est ici recherchée.

Pour achever la stabilisation à temps fixe du système (5.4.1), deux étapes sont définies :

a) En posant $\upsilon_1 = 1$, le sous–système Σ_2 peut être écrit comme

$$\begin{array}{ll} \dot{z}_2 = & z_3 \\ \dot{z}_3 = & \upsilon_2 \end{array} \tag{5.4.2}$$

L'entrée υ_2 sera conçue pour satisfaire

$$\left\{ \begin{array}{ll} z_2(t) = & 0 \\ z_3(t) = & 0 \end{array} \right., \qquad \forall t \geq T_1 \tag{5.4.3}$$

b) Lorsque $t > T_1$, l'Eq. (5.4.3) reste valable. En effet, l'entrée υ_2 est conçue de telle sorte que $z_3(t) = 0$. A partir de (5.4.1), ceci implique que z_2 et z_3 restent constantes quelque soit la valeur de l'entrée υ_1. Une commande à temps fixe sera mise en oeuvre pour satisfaire

$$z(t) = 0, \qquad \forall t \geq T_2 \tag{5.4.4}$$

Le théorème suivant présente la commande assurant la propriété de stabilité à temps fixe d'un système sous forme chaînée.

Théorème 5.1 *Considérons le système* (5.4.1) *dont la commande est définie par*

$$\begin{aligned} \upsilon_1 &= \begin{cases} 1 & si\ t \leq T_1 \\ -\alpha_3 \lfloor z_1 \rceil^2 - \beta_3 sign(z_1) & sinon \end{cases} \\ \upsilon_2 &= \begin{cases} \varphi(z_2, z_3) & si\ t \leq T_1 \\ -\alpha_4 sign(z_3) & sinon \end{cases} \end{aligned} \tag{5.4.5}$$

où le contrôleur à mode glissant d'ordre deux est défini comme

$$\varphi(z_2, z_3) = -\frac{\alpha_1 + 3\beta_1 z_2^2}{2} sign(s(z_2, z_3)) - \left\lfloor \alpha_2 s(z_2, z_3) + \beta_2 \lfloor s(z_2, z_3) \rceil^3 \right\rceil^{\frac{1}{2}} \tag{5.4.6}$$

avec la surface de glissement suivante

$$s(z_2, z_3) = z_3 + \left\lfloor \lfloor z_3 \rceil^2 + \alpha_1 z_2 + \beta_1 \lfloor z_2 \rceil^3 \right\rceil^{\frac{1}{2}}$$

et les constantes positives α_i, β_j ($i = 1, \ldots, 4$, $j = 1, \ldots, 3$). Le temps de commutation, qui ne dépend pas des conditions initiales du système, est explicitement défini par

$$T_1 = \frac{2}{\sqrt{\alpha_2}} + \frac{2}{\sqrt{\beta_2}} + \frac{2\sqrt{2}}{\sqrt{\alpha_1}} + \frac{2\sqrt{2}}{\sqrt{\beta_1}} \tag{5.4.7}$$

Par conséquent, l'origine du système (5.4.1) *est globalement stable en temps fixe où le temps de stabilisation est estimé par*

$$T < T_2 = T_1 + \frac{\pi}{2\sqrt{\alpha_3 \beta_3}} \tag{5.4.8}$$

Démonstration. La preuve est divisée en deux étapes

- Pour $0 \leq t \leq T_1$, comme $\upsilon_1 = 1$, le sous–système $\sum_2$ devient (5.4.2). D'après [Polyakov, 2012b], considérons la fonction candidate de Lyapunov $V_1 = |s(z_2, z_3)|$. Sa dérivée supérieure droite de Dini le long des trajectoires du système est, pour $s(z_2, z_3) \neq 0$,

$$D^* V_1 = \upsilon_2 sign(s(z_2, z_3)) + \frac{|z_3| \upsilon_2 sign(s(z_2, z_3)) + \frac{\alpha_1 + 3\beta_1 z_2^2}{2} z_3 sign(s(z_2, z_3))}{\left\lfloor \lfloor z_3 \rceil^2 + \alpha_1 z_2 + \beta_1 \lfloor z_2 \rceil^3 \right\rceil^{\frac{1}{2}}}$$

  L'équation (5.4.6) implique :

$$\upsilon_2 sign(s(z_2, z_3)) = -\frac{\alpha_1 + 3\beta_1 z_2^2}{2} - (\alpha_2 |s(z_2, z_3)| + \beta_2 |s(z_2, z_3)|^3)^{\frac{1}{2}}$$

On obtient donc :

$$D^*V_1 \leq -\left(\alpha_2 V_1 + \beta_2 V_1^3\right)^{\frac{1}{2}}$$

A l'aide du Lemme 5.1 avec $a = \alpha_2$, $b = \beta_2$, $p = 1$, $q = 3$ et $k = \frac{1}{2}$, on peut conclure que $V = |s(z_2, z_3)| = 0$ pour tout $t \geq \frac{1}{\alpha_2^{\frac{1}{2}}(1-\frac{1}{2})} + \frac{1}{\beta_2^{\frac{1}{2}}(\frac{3}{2}-1)} = \frac{2}{\sqrt{\alpha_2}} + \frac{2}{\sqrt{\beta_2}}$.

En mode glissant, i.e. $s = 0$, la dynamique devient

$$\dot{z}_2 = -\left\lfloor \frac{\alpha_1 z_2 + \beta_1 \lfloor z_2 \rceil^3}{2} \right\rceil^{\frac{1}{2}}$$

Considérons la seconde fonction de Lyapunov $V_2 = |z_2|$. Sa dérivée supérieure droite de Dini le long des trajectoires du système est

$$D^*V_2 \leq -\left(\frac{\alpha_1}{2} V_2 + \frac{\beta_1}{2} V_2^3\right)^{\frac{1}{2}}$$

Du Lemme 5.1, on peut conclure que $z_2 = 0$ pour tout $t \geq T_1$. On peut aussi noter que si $z_2 = 0$ et $s = 0$, alors $z_3 = 0$.

- Lorsque $t > T_1$, l'entrée υ_2 est conçue pour maintenir $z_3(t) = 0$. Considérons une troisième fonction candidate de Lyapunov $V_3 = \frac{1}{2} z_3^2$. Sa dérivée généralisée par rapport au temps est

$$D^*V_3 \leq -\alpha_4 \left(2V_3\right)^{\frac{1}{2}}$$

Comme $z_3(T_1) = 0$, on peut conclure que $z_3(t) = 0$ est maintenue pour $t > T_1$. A partir du système (5.4.1), ceci implique que $\dot{z}_2(t) = 0$ pour tout $t \geq T_1$. Donc, les états z_2 et z_3 restent constants peu importe la valeur de l'entrée υ_1. Considérons ensuite la fonction de Lyapunov $V_4 = \frac{1}{2} z_1^2$. Sa dérivée par rapport au temps le long des trajectoires du système est

$$\begin{aligned} D^*V_4 &= -\alpha_3 |z_1|^3 - \beta_3 |z_1| \\ &\leq -\alpha_3 \left(2V_4\right)^{\frac{3}{2}} - \beta_3 \left(2V_4\right)^{\frac{1}{2}} \end{aligned}$$

A partir du Lemme 5.2, l'origine du système (5.4.1) est globalement stable à temps fixe avec l'estimation du temps de stabilisation $T < T_2$.

■

Remarque 5.1 *Il convient de noter que comme le temps de commutation T_1 est indépendant des conditions initiales du système et qu'il peut être estimé à priori, la stabilité globale du système en boucle fermée est garantie.*

5.5 Rendez-vous d'agents non holonomes en un temps fixe

Pour résoudre le problème de rendez–vous à temps fixe vers un meneur, une stratégie, similaire à celle proposée dans la section précédente, est proposée pour prendre en compte la contrainte de non holonomie. Dans ce cas, le protocole de commande distribuée est divisé en deux étapes :

Étape 1 En utilisant les informations locales, l'état ξ_n du système (5.3.4) suit, en un temps donné T_2, l'état intermédiaire désiré $\xi_0^d = (\xi_{01}, \xi_{02}, 0)^T \in \mathbb{R}^3$. Cet état correspond à la configuration désirée $q_0^d = (x_0^d, y_0^d, \theta_0)^T \in \mathbb{R}^3$ avec

$$\begin{cases} x_0^d &= x_0 \sin^2\theta_0 - y_0 \cos\theta_0 \sin\theta_0 \\ y_0^d &= y_0 \cos^2\theta_0 - x_0 \cos\theta_0 \sin\theta_0 \end{cases} \tag{5.5.1}$$

On pourrait noter que (5.5.1) signifie que le point (x_0^d, y_0^d), défini par $x_0^d \cos\theta_0 + y_0^d \sin\theta_0 = 0$, appartient à la droite passant par le point (x_0, y_0) avec une pente $\tan\theta_0$. De plus, il convient de rappeler que l'état du meneur est seulement disponible pour une partie des N agents de la flottille (défini par la matrice D)

Pour achever l'étape 1, la procédure décrite par le Théorème 5.1 sera utilisée.

Étape 2 Lorsque $t > T_2$ et en utilisant les données locales, l'état ξ_n du système (5.3.4) suit, en un temps donné T_{max}, l'état désiré ξ_0. A partir du choix de q_0^d, il est préférable de souligner que pour $t > T_2$, le système (5.3.4) peut être réduit à un modèle simple intégrateur en utilisant la transformation

$$\begin{cases} \overline{x}_n &= \cos(\theta_n) x_n + \sin(\theta_n) y_n \\ \overline{x}_0 &= \cos(\theta_0) x_0 + \sin(\theta_0) y_0 \end{cases} \tag{5.5.2}$$

Cette transformation correspond à une rotation d'un angle θ_0. En effet, l'étape 1 a permis aux agents de converger vers le point (x_0^d, y_0^d) et de les orienter avec la pente voulue. Ainsi, l'étape 2 consiste à "aller tout droit" pour atteindre le point (x_0, y_0). Dans ce cas, considérons le vecteur $\widetilde{\overline{x}} = \left(\widetilde{\overline{x}}_1, \widetilde{\overline{x}}_2, \ldots, \widetilde{\overline{x}}_N\right)^T$, appelé vecteur de désaccord, exprimé par,

$$\widetilde{\overline{x}}_n = -\left(\sum_{j=1}^{N} a_{nj} \left(\overline{x}_j - \overline{x}_n\right) + a_{n0} \left(\overline{x}_0 - \overline{x}_n\right)\right), \quad \forall n = 1, \ldots, N \tag{5.5.3}$$

où les $a_{nj}, n, j = 1, \ldots, N$ sont les pondérations de la matrice adjacente A, correspondant aux liens de communication entre les agents n et j. Similairement, a_{n0} correspond à un lien de communication du meneur 0 vers l'agent n.

Remarque 5.2 *Le vecteur de désaccord $\widetilde{\overline{x}}_n$ peut être vu comme une erreur entre l'agent n et ses voisins j (incluant l'information du meneur, le cas échéant) dont il a l'information (i.e. $a_{nj} > 0$).*

D'une manière matricielle, le vecteur de désaccord est obtenu à l'aide de la matrice pondérée H (qui inclut la connaissance des liens entre le meneur et ses suiveurs) comme suit :

$$\widetilde{\overline{x}} = -(H\overline{x} - D1_N\overline{x}_0)$$

On peut noter que lorsque le temps de stabilisation est atteint, les vecteurs de désaccord convergent vers 0*, ce qui est équivalent à dire que tous les agents ont atteint la configuration du meneur.*

Le principe de l'algorithme de commande proposé est représenté sur la Fig. 5.8.

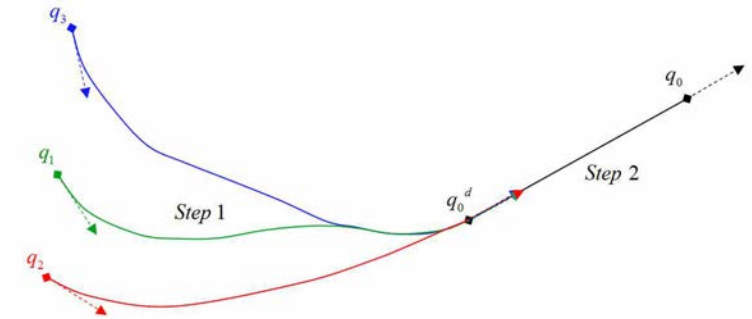


FIGURE 5.8 – Principe de la stratégie de commande distribuée proposée.

Pour de multiples systèmes sous forme chaînée, définissons les vecteurs de désaccord

$$\widetilde{\xi}_1 = \left(\widetilde{\xi}_{11}, \widetilde{\xi}_{21}, \ldots, \widetilde{\xi}_{N1}\right)^T, \widetilde{\xi}_2 = \left(\widetilde{\xi}_{12}, \widetilde{\xi}_{22}, \ldots, \widetilde{\xi}_{N2}\right)^T \text{ et } \widetilde{\xi}_3 = \left(\widetilde{\xi}_{13}, \widetilde{\xi}_{23}, \ldots, \widetilde{\xi}_{N3}\right)^T$$

tels que pour tout $n = 1, \ldots, N$,

$$\begin{array}{rcl} \widetilde{\xi}_{n1} & = & -\left(\sum_{j=1}^{N} a_{nj}\left(\xi_{j1} - \xi_{n1}\right) + a_{n0}\left(\xi_{01} - \xi_{n1}\right)\right) \\ \widetilde{\xi}_{n2} & = & -\left(\sum_{j=1}^{N} a_{nj}\left(\xi_{j2} - \xi_{n2}\right) + a_{n0}\left(\xi_{02} - \xi_{n2}\right)\right) \\ \widetilde{\xi}_{n3} & = & -\left(\sum_{j=1}^{N} a_{nj}\left(\xi_{j3} - \xi_{n3}\right) - a_{n0}\xi_{n3}\right) \end{array} \tag{5.5.4}$$

Théorème 5.2 *Supposons que l'hypothèse 5.1 est satisfaite. Le problème de rendez–vous à temps fixe vers un meneur est résolu par le protocole de commande distribué ν_n $(n = 1 \ldots, N)$ avec*

$$\begin{array}{rcl} \nu_{n1} & = & \begin{cases} 1 & si\ t \le T_1 \\ -\alpha_3 \left\lfloor \widetilde{\xi}_{n1} \right\rceil^2 - \beta_3 sign\left(\widetilde{\xi}_{n1}\right) & sinon \end{cases} \\ \nu_{n2} & = & \begin{cases} \left(\sum_{j=0}^{N} a_{nj}\right)^{-1}\left(\varphi(\widetilde{\xi}_{n2}, \widetilde{\xi}_{n3}) + \sum_{j=1}^{N} a_{nj}\nu_{j2}\right) & si\ t \le T_1 \\ -\alpha_4 sign\left(\widetilde{\xi}_{n3}\right) & si\ T_1 < t \le T_2 \\ -\xi_{n2}\nu_{n1} - \alpha_5 \left\lfloor \widetilde{\overline{x}}_n \right\rceil^2 - \beta_4 sign\left(\widetilde{\overline{x}}_n\right), & si\ t > T_2 \end{cases} \end{array} \tag{5.5.5}$$

où $\varphi(\widetilde{\xi}_{n2}, \widetilde{\xi}_{n3})$ est le contrôleur à mode glissant d'ordre deux (5.4.6) *et α_i, β_j ($i = 1, \ldots, 5$, $j = 1, \ldots, 4$) sont des constantes positives. Les temps de commutation, qui ne dépendent pas des conditions initiales du système, sont explicitement données par*

$$\begin{array}{rcl} T_1 & = & \frac{2}{\sqrt{\alpha_2}} + \frac{2}{\sqrt{\beta_2}} + \frac{2\sqrt{2}}{\sqrt{\alpha_1}} + \frac{2\sqrt{2}}{\sqrt{\beta_1}} \\ T_2 & = & T_1 + \frac{\pi N^{\frac{1}{4}}}{2\lambda_{min}(H)\sqrt{\alpha_3\beta_3}} \end{array} \tag{5.5.6}$$

Le temps de convergence est borné par

$$T < T_{max} = T_2 + \frac{\pi N^{\frac{1}{4}}}{2\lambda_{min}(H)\sqrt{\alpha_5\beta_4}} \tag{5.5.7}$$

Les gains de commande $\alpha_i (i = 1, 2, 3, 5)$ et $\beta_i (i = 1, \ldots, 4)$ peuvent être réglés pour obtenir un temps de convergence désiré par rapport au nombre d'agents ainsi qu'à la topologie de communication

Remarque 5.3 *On peut souligner que le temps de commutation T_2 et le temps de stabilisation T_{max} dépendent du nombre d'agents N. Par conséquent, si l'on veut garder un temps de stabilisation T_{max} avec un plus grand nombre d'agents N, les gains de commande α_3, β_3, α_5 et β_4 doivent être réglés, ce qui affecte directement les commandes* (5.5.5).

Remarque 5.4 *En raison des Eq.* (5.5.6)-(5.5.7)*, il est clair que les temps de commutation T_1 et T_2 dépendent du temps de stabilisation T_{max}. En effet, si un temps de stabilisation T_{max} est prescrit pour le problème de rendez–vous à temps fixe vers un meneur, alors les temps de commutation T_1 et T_2 devront être réglés à l'aide des gains α_i (i=1,2,4,5) et $\beta_i (i = 1, \ldots, 4)$.*

Démonstration. La preuve est ici divisée en trois étapes :

- Pour $0 \leq t \leq T_1$, comme $\nu_{n1} = 1$, les dérivées par rapport au temps des désaccords $\widetilde{\xi}_{n2}$ et $\widetilde{\xi}_{n3}$, pour tout $n = 1, \ldots, N$, sont données par

$$\begin{array}{rcl} \dot{\widetilde{\xi}}_{n2} & = & \widetilde{\xi}_{n3} \\ \dot{\widetilde{\xi}}_{n3} & = & \varphi(\widetilde{\xi}_{n2}, \widetilde{\xi}_{n3}) \end{array} \tag{5.5.8}$$

  De la même manière que la première étape du Théorème 5.1, l'origine du système (5.5.8) est globalement stable à temps fixe avec un temps de stabilisation estimé par $T < T_1$. Par conséquent, $\widetilde{\xi}_2$ converge vers zéro en un temps fixe borné par T_1. Il en vient du Théorème 4 dans [Khoo et al., 2009], que

$$\left\{ \begin{array}{rcl} \xi_{n2}(T_1) & = & \xi_{02} \\ \xi_{n3}(T_1) & = & 0 \end{array} \right. , \qquad \forall n \in \{1, \cdots, N\}$$

- Pour $T_1 < t \leq T_2$, l'entrée ν_{n2} est conçue pour maintenir $\xi_{n3}(t) = 0$. Définissons $\widehat{\xi}_3 = (\xi_{13}, \xi_{23}, \ldots, \xi_{N3})^T$. Considérons la fonction candidate de Lyapunov $V_1 = \frac{1}{2}(\widehat{\xi}_3)^T H \widehat{\xi}_3$. $H \succ \mathbf{0}$ car

le graphe $\mathcal{G}$ est connecté et il a au moins un a_{i0} positif. Sa dérivée généralisée par rapport au temps est

$$\begin{aligned} D^*V_1 &= -\alpha_4 \|H\widehat{\xi}_3\|_1 \\ &\leq -\alpha_4 \sqrt{2\lambda_{min}(H)V_1} \end{aligned}$$

Similairement à la seconde étape de la preuve du Théorème 5.1, on peut conclure que $\xi_{n3}(t) = 0$ pour $t \in [T_1, T_2]$. A partir de l'équation (5.3.4), ceci implique que $\dot{\xi}_{n2}(t) = 0$ pour tout $t \in [T_1, T_2]$. Donc, les états ξ_{n2} et ξ_{n3} restent constant quelque soit la valeur de ν_{n1}.

Définissons ensuite $\widehat{\xi}_1 = (\xi_{11} - \xi_{01}, \xi_{21} - \xi_{01}, \ldots, \xi_{N1} - \xi_{01})^T$. Considérons la fonction candidate de Lyapunov $V_2 = \frac{1}{2}(\widehat{\xi}_1)^T H \widehat{\xi}_1$. Sa dérivée par rapport au temps le long des trajectoires du système est définie par

$$\begin{aligned} D^*V_2 &= -\alpha_3 (\widehat{\xi}_1)^T H \left\lfloor H\widehat{\xi}_1 \right\rceil - \beta_3 \|H\widehat{\xi}_1\|_1 \\ &\leq -\alpha_3 N^{\frac{-1}{2}} \left(2\lambda_{min}(H)\right)^{\frac{3}{2}} V_2^{\frac{3}{2}} - \beta_3 \sqrt{2\lambda_{min}(H)V_2} \end{aligned} \tag{5.5.9}$$

En appliquant le Lemme 5.2, les équations suivantes sont valables.

$$\begin{aligned} \xi_{n1}(T_2) &= \xi_{01} \\ \xi_{n2}(T_2) &= \xi_{02} \ , \qquad \forall n \in \{1, \cdots, N\} \\ \xi_{n3}(T_2) &= 0 \end{aligned}$$

Il convient de souligner qu'en raison de la transformation (5.3.3), ce protocole de commande par rendez–vous achève la convergence de $\theta_n - \theta_0$ $(n = 1, \ldots, N)$ vers zéro en un temps fini borné par T_2. Ainsi, la dernière étape consiste à maintenir θ_n à θ_0 tout en atteignant le rendez–vous en un temps fixe, i.e.,

$$\begin{aligned} x_n(t) &= x_0 \\ y_n(t) &= y_0 \end{aligned} \ , \qquad \forall t \geq T_{max}, \qquad \forall n \in \{1, \cdots, N\} \tag{5.5.10}$$

- Pour $t > T_2$, l'entrée ν_{n1} est conçue pour maintenir $\theta_n - \theta_0$ à 0. Comme pour tout $n \in \{1, \cdots, N\}$, $\theta_n(T_2) = \theta_0$, on peut conclure à l'aide de (5.5.9) que, quelque soit l'entrée ν_{n2},

$$\theta_n(t) = \theta_0, \qquad \forall t \geq T_2, \qquad \forall n \in \{1, \cdots, N\} \tag{5.5.11}$$

Par conséquent, pour tout $n = 1, \ldots, N$, l'équation (5.5.2) devient

$$\left\{ \begin{array}{rcl} \overline{x}_n &=& \cos(\theta_0) x_n + \sin(\theta_0) y_n \\ \overline{y}_n &=& \cos(\theta_0) y_n - \sin(\theta_0) x_n \end{array} \right. \tag{5.5.12}$$

Pour $t > T_2$, la dynamique de l'état transformé peut être réduite à

$$\begin{cases} \dot{\widetilde{\overline{x}}}_n &= -\alpha_5 \left\lfloor \widetilde{\overline{x}}_n \right\rceil^2 - \beta_4 sign\left(\widetilde{\overline{x}}_n\right) \\ \dot{\overline{y}}_n &= 0 \end{cases} \tag{5.5.13}$$

De plus, en raison des Eq. (5.3.3), (5.5.1) et (5.5.13) il en vient que

$$\begin{array}{rcl} \overline{y}_n(t) &=& \xi_{n2}(t) \\ &=& x_0^d \sin\theta_0 - y_0^d \cos\theta_0 \end{array} , \qquad \forall t \geq T_2, \qquad \forall n \in \{1, \cdots, N\} \tag{5.5.14}$$

Définissons $\widehat{x} = (\overline{x}_1 - \overline{x}_0, \overline{x}_2 - \overline{x}_0, \ldots, \overline{x}_N - \overline{x}_0)^T$. Considérons la fonction candidate de Lyapunov $V_3 = \frac{1}{2}(\widehat{x})^T H \widehat{x}$. Sa dérivée le long des trajectoires est

$$D^* V_3 \leq -\alpha_5 N^{\frac{-1}{2}} \left(2\lambda_{min}(H)\right)^{\frac{3}{2}} V_3^{\frac{3}{2}} - \beta_4 \sqrt{2\lambda_{min}(H) V_3}$$

A l'aide du Lemme 5.2, on obtient

$$\overline{x}_n(t) = \cos(\theta_0) x_0 + \sin(\theta_0) y_0 \ , \qquad \forall t \geq T_{max}, \qquad \forall n \in \{1, \cdots, N\} \tag{5.5.15}$$

A partir des Eq. (5.5.11)-(5.5.15), on peut finir par conclure que le problème de rendez–vous vers un leader est résolu en un temps de stabilisation estimé par $T < T_{max}$.

■

5.6 Conclusion

Dans ce chapitre, un protocole de commande permettant un rendez–vous d'agents vers un meneur immobile a été développé. La position du meneur étant fixe, le protocole permet aux agents d'atteindre une position spécifique comme par exemple une ressource disponible. Après avoir rappelé la théorie des graphes, représentant les communications entre agents, la stabilité à temps fixe a été présentée. Elle permet à un ou plusieurs systèmes de converger vers un état désiré en un temps qui ne dépend pas des conditions initiales du ou des systèmes, ce qui est un atout en production où on est contraint par le temps.

Ensuite, le problème a été formulé et les transformations du modèle des agents ont été introduites. Le protocole consiste dans un premier temps à passer sous forme chaînée. Par conséquent, la stabilisation d'un système chaîné, utile au contrôleur proposé, a été présentée. Au vu de la difficulté de trouver une commande stabilisant le système, une stratégie par commutation a été adoptée. Elle permet, en divisant le système en sous–systèmes, de faire converger l'état en plusieurs étapes. Nous avons vu que chaque temps de commutation est indépendant des conditions initiales, permettant de garder l'aspect

de temps fixe à chaque étape.

Après avoir présenté la stabilisation du système sous forme chaînée, le protocole de commande de rendez–vous vers un meneur a été introduit. Il se déroule en deux étapes où la première permet d'atteindre en temps fixe une position intermédiaire particulière (qui dépend de la position désirée) en mettant le système sous forme chaînée. La seconde étape consiste à rejoindre, en ligne droite, la position désirée initialement en partant de la position intermédiaire. Les différents résultats numériques du protocole de commande de rendez–vous vers un meneur en temps fixe seront présentés et discutés dans le chapitre suivant.

Chapitre 6

Les résultats numériques et expérimentaux

6.1 Introduction

Dans ce chapitre, les résultats numériques de la planification de trajectoire en milieu manufacturier pour les architectures hétérarchique (Chapitre 3) et supervisée (Chapitre 4) sont présentés. Les résultats du problème de rendez–vous, introduit dans le Chapitre 5, le seront aussi. De plus, des résultats expérimentaux pour la planification de trajectoire seront fournis pour tester la faisabilité des méthodes proposées.

Pour la planification de trajectoire, des scénarios, basés sur des ateliers de production flexibles (i.e. avec un nombre suffisant d'agents et de ressources), seront mis en place pour tester les algorithmes de planification développés. Ceci permettra de montrer la faisabilité de ces algorithmes, les avantages de la combinaison planification de trajectoire/ordonnancement et de voir les impacts de la myopie. Ensuite, un scénario plus simple sera étudié pour pouvoir comparer les deux algorithmes de planification de trajectoire. En terme de commande coopérative, un scénario de rendez–vous vers un meneur en temps fixe sera proposé. Il permettra de montrer les différentes étapes conçues pour permettre un rendez–vous de plusieurs agents. Nous verrons donc l'effet des commandes pour chaque étape du protocole avec des temps de commutation et un temps de stabilisation donnés préalablement.

Ensuite, nous proposerons des résultats expérimentaux pour l'approche supervisée de planification de trajectoire qui permettront de montrer la faisabilité de l'approche supervisée. Pour ces résultats, des robots Lego Mindstorms seront utilisés et leurs principales caractéristiques seront données. Enfin, nous proposerons une discussion sur les différentes hypothèses données dans le Chapitre 2 et sur les capacités des AGVs, importantes pour le déploiement de ceux–ci en milieu industriel.

6.2 Résultats numériques

Dans cette section, nous allons présenté les résultats numériques de la planification de trajectoire et de la commande coopérative. Pour les résultats sur la planification de trajectoire, les scénarios que nous proposons sont différents pour permettre de montrer certains aspects, que l'on peut difficilement montré avec un scénario commun. Pour cela, nous avons aussi proposé un scénario plus simple pour pouvoir comparer les deux approches.

Notons que pour l'ensemble des résultats de la planification, la vitesse maximale tolérée par les agents est $v_{max} = 1.5m/s$. Les trajectoires sont paramétrisées en utilisant 9 points de contrôle (voir B-spline, Chapitre 2). Pour l'algorithme de navigation correspondant à l'architecture supervisée, la période de mise à jour de trajectoire est $T_c = 0.5s$, le nombre de mise à jour de trajectoire entre deux ordonnancements est $m = 6$ et l'horizon de faible déviation, décrit dans le Chapitre 4 via l'Eq. (4.4.14), est $T_p = 3s$.

6.2.1 Résultats de la planification : architecture hétérarchique

Le scénario utilisé pour tester l'approche hétérarchique est inspiré d'un atelier flexible réel (voir Fig. 6.1). Ce système manufacturier, se situant à l'AIP–PRIMECA dans l'université de Valenciennes, va nous permettre de tester l'algorithme de navigation introduit dans le Chapitre 3 pour un scénario orienté vers l'industrie. L'atelier AIP est composé d'éléments industriels (robots, système de convoi, ...) et fournit

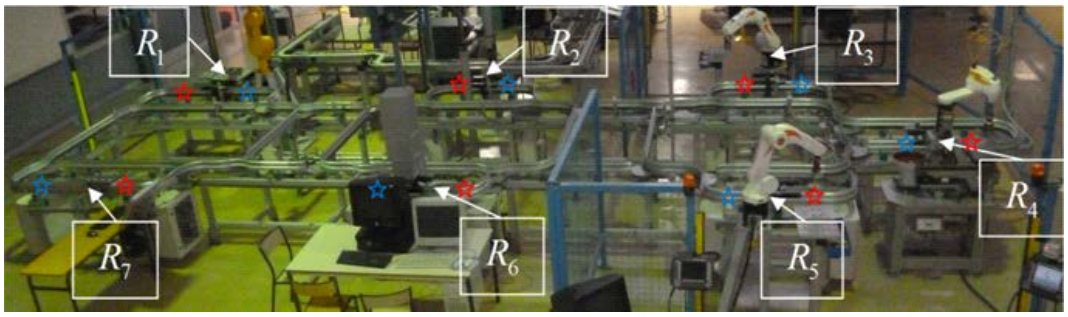


FIGURE 6.1 – L'atelier flexible AIP-PRIMECA.

plusieurs flexibilités (ressource, manutention, ...). Cependant, le transport des produits se fait d'une manière guidée et on ne peut donc pas utiliser cet atelier pour obtenir des résultats expérimentaux. Par conséquent, l'atelier est modélisé en un scénario de simulation pour pouvoir considérer la navigation des AGVs. L'atelier AIP modifié a sept ressources statiques où, pour chaque ressource, les agents démarrent à une position de départ (représentée par une étoile bleue) et arrivent à une position finale (étoile rouge) représentant la file d'attente de la ressource. La distance de sécurité est $d_{safe} = 0.8m$ et la portée de communication est $R_{com} = 3m$. Les données concernant chaque ressource sont fournies dans le tableau 6.1. Rappelons que les temps d'attente ne sont pas connus par les agents et sont supposés

fixes pour ceux–ci. Pour ce scénario, on considérera que les temps de traitement donnés sur le tableau 6.1 incluent le temps d'attente à la ressource.

Tableau 6.1 – Données des différentes ressources

Ressource R_b	Position de départ	Position finale	Temps de traitement
R_1	$\begin{bmatrix} 1.2 & 15 \end{bmatrix}^T$	$\begin{bmatrix} 3.2 & 15 \end{bmatrix}^T$	$28.1s$
R_2	$\begin{bmatrix} 11.2 & 17.5 \end{bmatrix}^T$	$\begin{bmatrix} 13.2 & 17.5 \end{bmatrix}^T$	$21.3s$
R_3	$\begin{bmatrix} 21.2 & 15 \end{bmatrix}^T$	$\begin{bmatrix} 23.2 & 15 \end{bmatrix}^T$	$26.4s$
R_4	$\begin{bmatrix} 27.5 & 9.3 \end{bmatrix}^T$	$\begin{bmatrix} 27.5 & 11.3 \end{bmatrix}^T$	$33.8s$
R_5	$\begin{bmatrix} 23.8 & 5 \end{bmatrix}^T$	$\begin{bmatrix} 21.8 & 5 \end{bmatrix}^T$	$21s$
R_6	$\begin{bmatrix} 13.8 & 2.5 \end{bmatrix}^T$	$\begin{bmatrix} 11.8 & 2.5 \end{bmatrix}^T$	$19.1s$
R_7	$\begin{bmatrix} 3.8 & 5 \end{bmatrix}^T$	$\begin{bmatrix} 1.8 & 5 \end{bmatrix}^T$	$24.2s$

Pour ce scénario, on considère vingt agents naviguant dans l'atelier AIP modifié. Ici, le nombre d'agents et de ressource est suffisamment élevé pour montrer l'effet de la myopie. De plus, il permettra aussi de montrer plus facilement la combinaison entre planification de trajectoires et ordonnancement. Le tableau 6.2 donne les paramètres initiaux des agents ainsi que les résultats finaux. Notons que les termes "initial" et "final" correspondent respectivement aux résultats en début de simulation (à l'aide d'une trajectoire directe générée avant que les agents ne démarrent de leur ressource) et aux résultats en fin de simulation. Rappelons qu'il est supposé que chaque agent démarre de la ressource où son opération précédente a été achevée. De plus, lorsqu'un agent démarre d'une ressource, l'agent suivant attend (le temps que son opération soit achevée) avant de démarrer de cette même ressource.

Comme le montre le tableau 6.2, tous les agents sont capables d'atteindre une ressource permettant d'achever leur opération avant l'échéance donnée (i.e. $ct_{nlb} < odd_{nl}$). Les trajectoires sans collision des agents sont illustrées sur la Fig. 6.2. La Fig. 6.3 permet de vérifier que les contraintes de vitesse et d'évitement de collision soient respectées.

En utilisant le tableau 6.2, on remarque que les agents A_3, A_8, A_{12}, A_{16} et A_{17} se dirigent initialement vers la même ressource (la ressource choisie initialement est R_2). Ceci peut engendrer un problème d'interblocage autour de la ressource dû à l'évitement de collision. Cependant, la combinaison de la planification de trajectoire avec l'ordonnancement permet aux agents de changer de destination lors de la navigation. Par exemple, les agents A_8, A_{12} et A_{17} ont pu éviter l'interblocage en choisissant une autre ressource (la ressource finale choisie est différente de R_2).

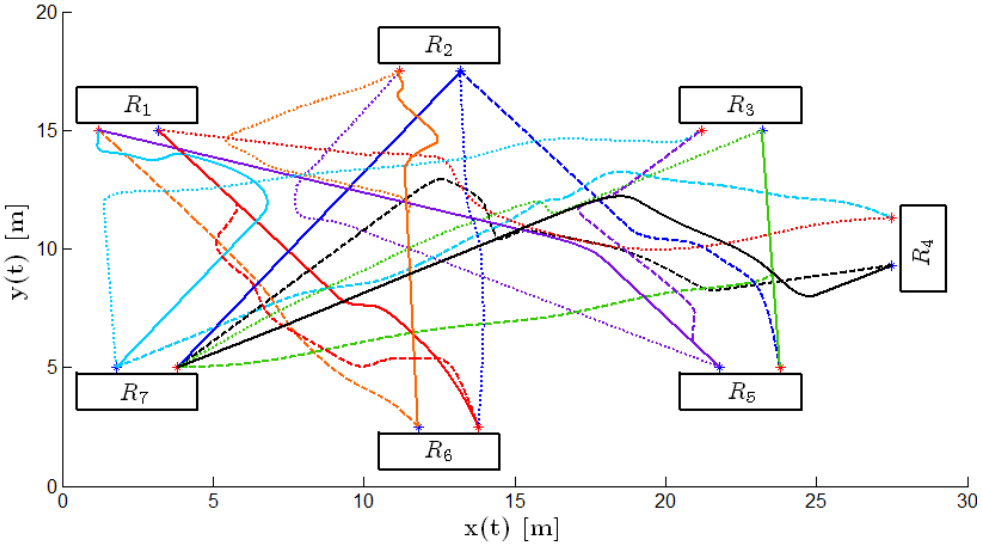
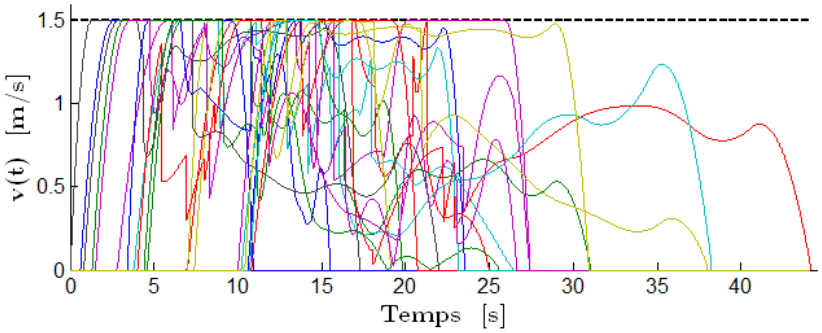
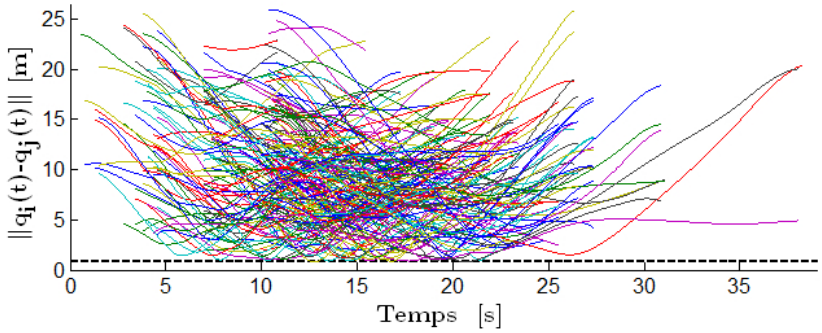


FIGURE 6.2 – Trajectoires des agents (approche hétérarchique) dans l'atelier AIP modifié.



(a)



(b)

FIGURE 6.3 – Vérification des contraintes de l'approche hétérarchique pour l'atelier AIP modifié : (a) vitesse, (b) évitement de collision

1. SR : Ressource de départ, [2] RP : Possibilités de ressource, [3] ICR : Ressource choisie initialement, [4] FCR : Ressource finale choisie, [5] ST : Temps de départ, [6] FT : Temps final, [7] DD : Échéance de l'opération, [8] ICT : Temps d'achèvement initial, [9] FCT : Temps d'achèvement final.

Tableau 6.2 – Paramètres et résultats numériques pour chaque agent : approche hétérarchique

A_i	SR[1] b	RP[2] $\mathcal{R}_{nl}$	ICR[3] b	FCR[4] b	ST[5] $T_{i,init}$	FT[6] $T_{i,fin}$	DD[7] odd_{nl}	ICT[8] ct_{nlb}	FCT[9] ct_{nlb}
A_1	R_6	$\{1,4\}$	R_1	R_1	$10.6s$	$23.5s$	$64.6s$	$51s$	$51.6s$
A_2	R_1	$\{4,6\}$	R_6	R_6	$4.6s$	$19.8s$	$69.6s$	$36s$	$38.9s$
A_3	R_5	$\{2\}$	R_2	R_2	$7s$	$25s$	$86s$	$40.6s$	$46.3s$
A_4	R_7	$\{3,4\}$	R_3	R_4	$3.8s$	$26.4s$	$64.8s$	$46.6s$	$60.2s$
A_5	R_4	$\{1,7\}$	R_7	R_7	$2.8s$	$27.4s$	$70.8s$	$45s$	$51.6s$
A_6	R_2	$\{7\}$	R_7	R_7	$7.4s$	$19.1s$	$64.4s$	$43.3s$	$43.3s$
A_7	R_3	$\{5,7\}$	R_5	R_7	$0s$	$17.3s$	$73s$	$28.5s$	$41.5s$
A_8	R_7	$\{1\}$	R_2	R_1	$0.6s$	$15.5s$	$58.6s$	$33.6s$	$43.6s$
A_9	R_6	$\{1,2,4\}$	R_2	R_2	$4.4s$	$31.1s$	$70.4s$	$37s$	$52.4s$
A_{10}	R_1	$\{4\}$	R_4	R_4	$10.8s$	$44.2s$	$81.8s$	$63s$	$78s$
A_{11}	R_3	$\{7\}$	R_7	R_7	$10.2s$	$38.3s$	$87.2s$	$50.7s$	$62.5s$
A_{12}	R_5	$\{1,2,3\}$	R_2	R_1	$10s$	$27.4s$	$89s$	$43.6s$	$55.5s$
A_{13}	R_7	$\{1,3\}$	R_1	R_3	$7s$	$38.1s$	$74s$	$42.6s$	$64.5s$
A_{14}	R_2	$\{5,7\}$	R_5	R_5	$0.8s$	$22s$	$55.8s$	$34.1s$	$43s$
A_{15}	R_3	$\{5\}$	R_5	R_5	$3.4s$	$10.9s$	$58.4s$	$31.9s$	$31.9s$
A_{16}	R_6	$\{2,4\}$	R_2	R_2	$1.3s$	$25.6s$	$59.3s$	$33.9s$	$46.9s$
A_{17}	R_5	$\{2,3\}$	R_2	R_3	$4s$	$20.7s$	$80s$	$37.6s$	$47s$
A_{18}	R_2	$\{4,6\}$	R_6	R_6	$10.7s$	$23.2s$	$68.7s$	$41.1s$	$42.3s$
A_{19}	R_1	$\{4,6\}$	R_6	R_6	$1.5s$	$26.6s$	$76.5s$	$32.9s$	$45.7s$
A_{20}	R_4	$\{2,6,7\}$	R_6	R_7	$10.4s$	$30.9s$	$88.4s$	$41s$	$55.1s$

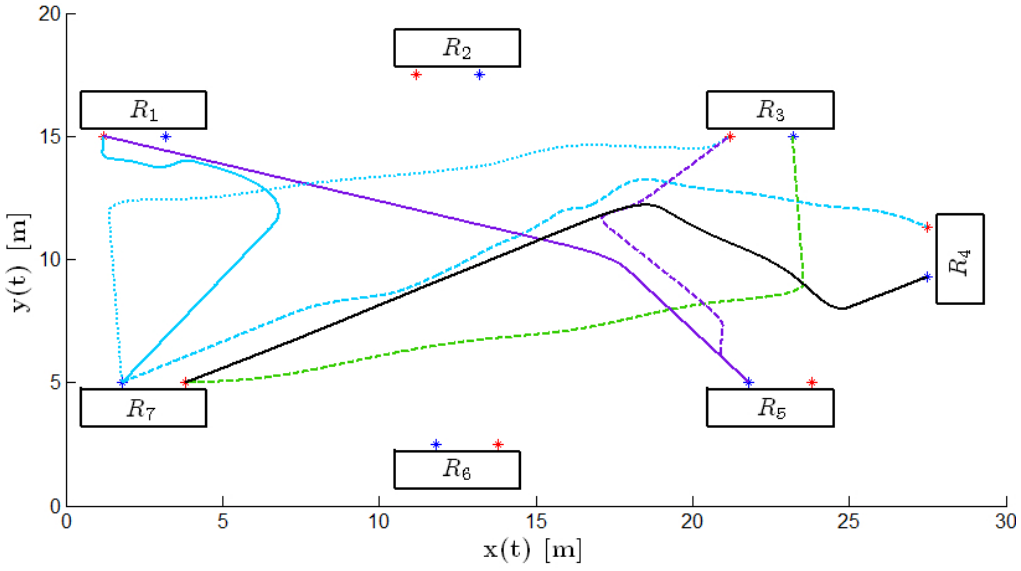


FIGURE 6.4 – Trajectoire des agents (approche hétérarchique) qui changent de ressource.

D'une manière générale, on remarque à l'aide de la Fig. 6.4 que sept agents ont mis à jour leur destination lors de la navigation grâce à la fonction d'ordonnancement combinée à la planification de trajectoire. La grande partie des résultats existants dans la littérature nécessite un ordonnancement avant que l'agent ne démarre d'une ressource. Ceci implique que les agents doivent se diriger vers la ressource allouée sans possibilité de changer. On en vient donc à s'interroger sur ce qui se passerait si ces sept agents ne pouvaient pas ordonnancer leur produits. Pour cela, un scénario avec les même données a été testé. Le seul changement est pour les sept agents qui ont pour seule possibilité de ressource, celle choisie initialement (i.e. celle correspondant à une trajectoire directe entre les ressources). Par conséquent on pourra vérifier la pertinence de la combinaison entre la planification de trajectoire (plan.) et l'ordonnancement (ord.) en comparant les résultats avec un découplage de ces fonction, c'est–à–dire sans combinaison où l'ordonnancement est fait à l'instant initial. Le tableau 6.3 est donnée en guise de comparaison entre approches découplée et combinée.

Tableau 6.3 – Comparaison de temps d'achèvement pour les agents ayant changé leur destination

A_i	ICR[3] b	Temps d'achèvement avec découplage ord./plan.	FCR[4] b	Temps d'achèvement avec combinaison ord./plan.
A_4	R_3	$67.3s$	R_4	$60.2s$
A_7	R_5	$44.7s$	R_7	$41.5s$
A_8	R_2	$59.2s$	R_1	$43.6s$
A_{12}	R_2	$71.4s$	R_1	$55.5s$
A_{13}	R_1	$65.1s$	R_3	$64.5s$
A_{17}	R_2	$73.2s$	R_3	$47s$
A_{20}	R_6	$59.8s$	R_7	$55.1s$

On remarque que si l'ordonnancement et la planification de trajectoire sont découplés, il y a une perte de performances en termes de temps d'achèvement. En effet, la combinaison permet à ces agents de gagner en moyenne $10.47s$ sur le temps d'achèvement de l'opération. De plus, ce découplage ne permet pas aux agents A_4 et A_8 de respecter l'échéance donnée car les temps d'achèvement ct_{nlb} de ces agents donné sur le tableau 6.3 sont supérieurs aux échéance donnés sur le tableau 6.2 (pour l'agent 4, $ct_{nlb} = 67.3 > odd_{nl} = 64.4$ et pour l'agent 8, $ct_{nlb} = 59.2 > odd_{nl} = 58.6$). Ainsi, on remarque l'avantage de la combinaison permettant aux agents de choisir une ressource plus appropriée pour achever l'opération. Pour finir, on remarque que certains agents changent plusieurs fois de destination pour, au final, arriver à la ressource initialement choisie. Ces changements successifs, illustrés sur la Fig. 6.5, se produisent à cause du phénomène de myopie de l'architecture hétérarchique que nous avions mentionné dans le Chapitre 3. En effet, comme les agents n'ont qu'une connaissance locale,

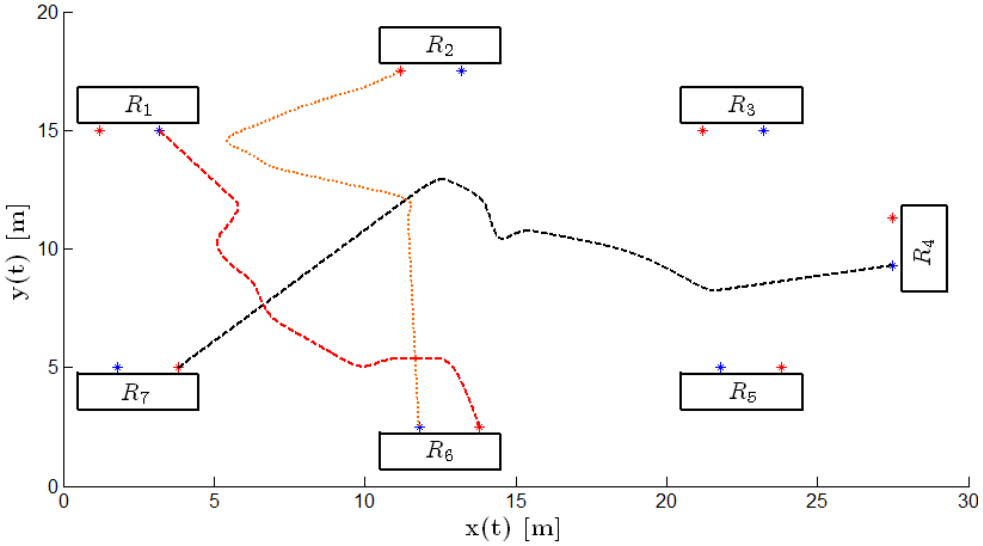


FIGURE 6.5 – Trajectoire des agents (approche hétérarchique) ayant un comportement myope.

ils choisissent uniquement la ressource avec ces données locales. Si les données locales changent, ils doivent s'adapter pour éviter les collisions mais aussi considérer les choix de ressource des voisins ayant un niveau de priorité plus haut. Par conséquent, si les données locales changent régulièrement, un phénomène d'oscillations de trajectoire se produit à cause du changement successif de ressource. On retrouve donc bien la description de la myopie donnée dans le Chapitre 1.

6.2.2 Résultats de la planification : architecture supervisée

D'une manière semblable, le scénario proposé pour tester l'approche supervisée est inspiré d'un atelier flexible basé sur AGV. La configuration du système de production pour ce scénario est similaire à celle présentée sur la Fig. 2.3 où 5 ressources immobiles effectuent certaines opérations sur des produits. L'environnement de production n'est pas le même que précédemment afin de montrer la détection de conflits par le superviseur. En effet, l'environnement de production de ce scénario a, en son centre, plus de chance d'amener des conflits de collision. Pour ce scénario, rappelons que chaque agent démarre d'une ressource dès que son opération précédente est achevée afin d'effectuer la suivante. Les données de chaque ressource sont fournies dans le tableau 6.4. Rappelons que, pour l'approche supervisée, les temps d'attente aux ressources ne sont pas fixes puisqu'ils sont connus par les agents à chaque mise à jour de trajectoire τ_k.

Pour ce scénario, treize agents naviguent dans l'environnement de production où les paramètres initiaux de chaque agent sont donnés dans le tableau 6.5. La portée de communication est $R_{com} = 8m$, la distance de sécurité est $d_{safe} = 0.8m$ et la période entre deux mises à jour de trajectoire est $T_c = 0.5$.

Le tableau 6.6 donne les résultats numériques correspondants à l'approche supervisée. Les trajectoires sans collision des agents sont illustrées sur la Fig. 6.6. On peut voir à l'aide de la Fig. 6.7 que les

Tableau 6.4 – Données des différentes ressources

Ressource	Position de départ	Position d'arrivée	Temps d'attente à $t = 0s$
R_1	$\begin{bmatrix} 16 & 30 \end{bmatrix}^T$	$\begin{bmatrix} 13.5 & 30 \end{bmatrix}^T$	$11.6s$
R_2	$\begin{bmatrix} 2.5 & 18.5 \end{bmatrix}^T$	$\begin{bmatrix} 2.5 & 16 \end{bmatrix}^T$	$9.8s$
R_3	$\begin{bmatrix} 27.5 & 16.5 \end{bmatrix}^T$	$\begin{bmatrix} 27.5 & 19 \end{bmatrix}^T$	$7.4s$
R_4	$\begin{bmatrix} 2.5 & 6 \end{bmatrix}^T$	$\begin{bmatrix} 2.5 & 3.5 \end{bmatrix}^T$	$14.2s$
R_5	$\begin{bmatrix} 27.5 & 4 \end{bmatrix}^T$	$\begin{bmatrix} 27.5 & 6.5 \end{bmatrix}^T$	$13.8s$

Tableau 6.5 – Paramètres initiaux des agents : approche supervisée

Agent	Temps de départ	Ressource de départ	Ressources possibles	Temps de traitement	Échéance
A_1	$14.2s$	R_4	$\{R_5\}$	$5.1s$	$61.8s$
A_2	$1.2s$	R_3	$\{R_2, R_4\}$	$5.8s$	$85.4s$
A_3	$5.7s$	R_1	$\{R_4, R_5\}$	$5.5s$	$70.5s$
A_4	$2.3s$	R_5	$\{R_1, R_2\}$	$4.9s$	$80.4s$
A_5	$9.8s$	R_2	$\{R_3, R_5\}$	$5.7s$	$63.9s$
A_6	$7.1s$	R_5	$\{R_2, R_4\}$	$5.6s$	$79.9s$
A_7	$0s$	R_1	$\{R_4\}$	$5.9s$	$61.1s$
A_8	$8.1s$	R_4	$\{R_1\}$	$5.1s$	$60.5s$
A_9	$7.4s$	R_3	$\{R_2\}$	$5.1s$	$76.0s$
A_{10}	$2.5s$	R_4	$\{R_1, R_3\}$	$6.0s$	$74.2s$
A_{11}	$13.8s$	R_5	$\{R_1, R_4\}$	$5.5s$	$91.7s$
A_{12}	$4.3s$	R_2	$\{R_3\}$	$5.9s$	$76.2s$
A_{13}	$11.6s$	R_1	$\{R_2, R_5\}$	$5.0s$	$95.1s$

contraintes de vitesse et d'évitement de collision sont respectées.

Les résultats de l'approche hétérarchique ont montré le gain de performances lorsque la planification de trajectoire est combinée avec l'ordonnancement. Ici, les temps finaux sont comparés avec leur borne inférieure. Rappelons que la borne inférieure correspond à la trajectoire directe générée avant que l'agent ne démarre et sans prendre en compte l'évitement de collision.

De la même manière que l'approche hétérarchique, la fonction d'ordonnancement reste combinée à la planification de trajectoire. Certains agents ont donc la possibilité de changer la ressource de destination lors de la navigation. Cependant, on remarque à l'aide du tableau 6.6 qu'uniquement l'agent A_3 a changé de façon permanente sa ressource de destination. Ceci est dû à l'ajout du superviseur qui, en détectant les conflits, permet aux agents d'avoir une connaissance à priori des conflits de collision qui peuvent se produire durant la navigation. La détection leur permet aussi d'éviter des zones de conflits comme illustré sur la Fig. 6.8. En effet, on remarque que l'agent 13, dont la trajectoire est en bleu,

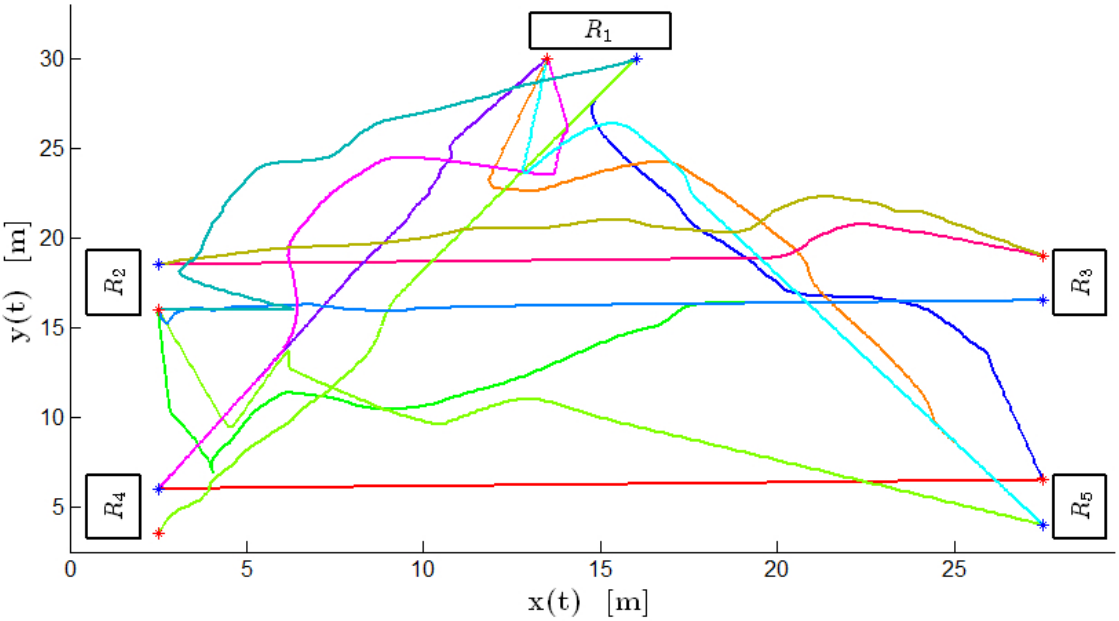
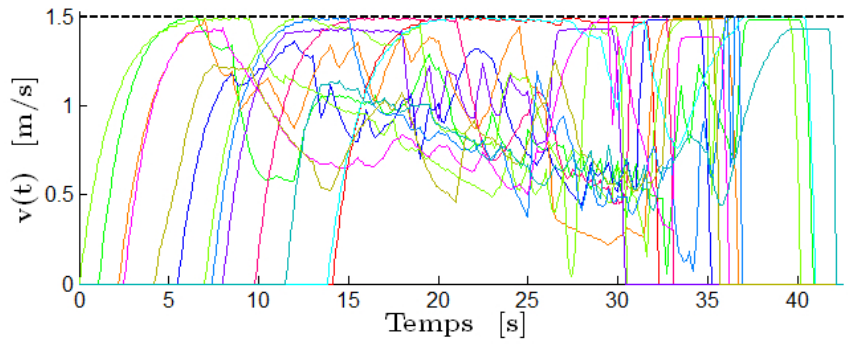
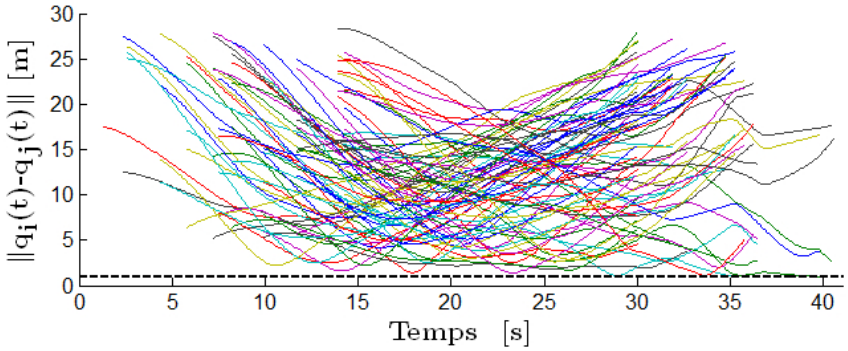


FIGURE 6.6 – Trajectoires des agents (approche supervisée).



(a)



(b)

FIGURE 6.7 – Vérification des contraintes de l'approche supervisée pour l'atelier flexible : (a) vitesse, (b) évitement de collision

Tableau 6.6 – Résultats numériques pour chaque agent : approche supervisée

Agent	Ressource initiale optimale	Borne inférieure du temps final	Temps final	Temps d'achèvement	Ressource choisie
A_1	R_5	$32.2s$	$32.2s$	$37.3s$	R_5
A_2	R_2	$20.3s$	$40.9s$	$53.4s$	R_2
A_3	R_4	$28.1s$	$35.2s$	$41.9s$	R_5
A_4	R_1	$24.5s$	$36.8s$	$46.8s$	R_1
A_5	R_3	$28.6s$	$33.0s$	$38.7s$	R_3
A_6	R_2	$27.9s$	$40.1s$	$47.6s$	R_2
A_7	R_4	$22.4s$	$30.5s$	$36.4s$	R_4
A_8	R_1	$28.1s$	$30.4s$	$35.5s$	R_1
A_9	R_2	$26.2s$	$36.9s$	$42.0s$	R_2
A_{10}	R_1	$22.4s$	$35.9s$	$41.9s$	R_1
A_{11}	R_1	$36.1s$	$41.0s$	$52.3s$	R_1
A_{12}	R_3	$23.1s$	$35.7s$	$44.6s$	R_3
A_{13}	R_2	$26.3s$	$42.2s$	$58.4s$	R_2

anticipe le conflit représenté par un cercle bleu. De plus, l'agent 10 (trajectoire de couleur magenta) anticipe les conflits représentés par le cercle en rouge. On remarque que ce conflit concerne les agents 5 et 12 qui partent de la ressource R_2 pour arriver à la ressource R_3. Ces agents engendre des conflits pour l'agent 10 car l'agent 5 procède à une manoeuvre lui permettant d'arriver à la ressource R_3 avant l'agent 12. Malgré l'anticipation des conflits, on remarque que le phénomène de myopie reste présent car

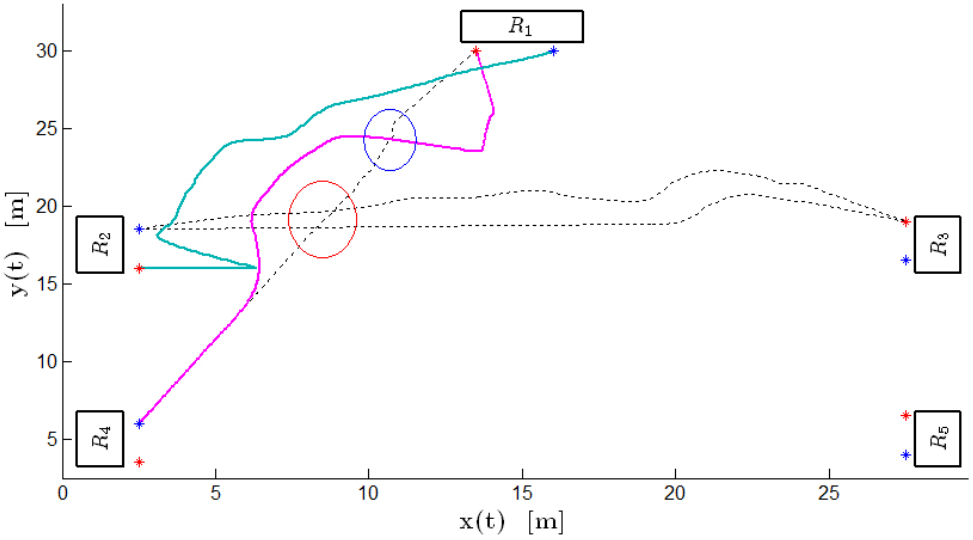


FIGURE 6.8 – Exemples d'agents anticipant les conflits de collision.

les agents 2 et 6 changent plusieurs fois de destinations comme le montre la Fig. 6.9. D'abord, rappelons que l'ajout du superviseur permet de réduire la myopie mais pas de la supprimer. Ce comportement

des agents est lié aux conflits de ressource. En effet, ces agents ont tous les deux la possibilité de se diriger vers les ressources R_2 et R_4 et ont des échéances proches. De plus, ces agents doivent aussi prendre en compte les autres agents ayant un niveau de priorité plus haut. Comme les agents A_2 et A_6 ont des performances similaires, ils entrent en compétition et donc leurs niveaux de priorité respectifs changent régulièrement. Par conséquent, ils essayent au mieux de choisir leur destination, amenant ainsi un phénomène d'oscillation. D'une manière générale, la myopie rencontrée pour cette approche

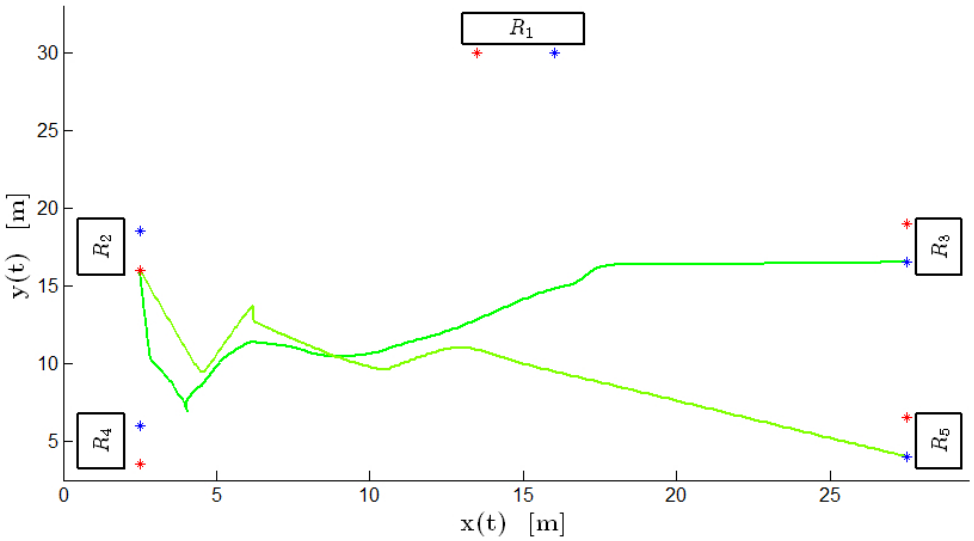


FIGURE 6.9 – Trajectoire des agents (approche supervisée) ayant un comportement myope.

supervisée est due aux conflits de ressource. De ce fait, on en vient à s'interroger sur la manière dont on pourrait éviter ce type de comportement sans pour autant changer l'algorithme de navigation. Pour cela, plusieurs points sont à discuter, principalement sur certaines hypothèses données dans le Chapitre 2. La plupart des hypothèses seront discutées dans la Section 6.3.2 permettant de voir l'impact de ces hypothèses sur le comportement des agents.

6.2.3 Comparaison architectures hétérarchique/supervisée

Dans les chapitres 3 et 4, deux algorithmes de navigation combinant planification de trajectoire et ordonnancement ont été proposés pour résoudre un même objectif : planifier une trajectoire vers une ressource à choisir afin d'achever l'opération en cours du produit transporté par l'agent le plus tôt possible. Nous avons vu dans ces chapitres qu'une architecture supervisée (Chapitre 4) est préférable à une architecture hétérarchique (Chapitre 3) pour réduire le phénomène de myopie. Ici, les deux méthodes seront comparées afin de montrer le gain en performances que l'ajout du superviseur apporte.

Le scénario proposé pour la comparaison considère 3 ressources et 4 agents. Les ressources R_1, R_2 et R_3 se situent aux positions $[1.5, 0.5]$, $[5, 0]$ et $[8, 0.25]$, ont les temps de traitement $1.1s$, $1s$ et $1s$ et les temps d'attente 6.1, $10s$ and $9.8s$, respectivement. Rappelons que les temps d'attente aux

ressources sont supposés fixes pour l'approche hétérarchique lors de la navigation (au vu du manque de communication avec le niveau M.O.R.M.). Par conséquent, ces temps d'attente seront supposés non variables lors de la navigation pour les deux approches afin de pouvoir comparer celles–ci. Les paramètres initiaux liés aux agents/produits sont donnés dans le tableau 6.7.

Tableau 6.7 – Paramètres initiaux des agents pour le scénario comparatif

	A_1	A_2	A_3	A_4
Position initiale $q_{i,init}$	$[2,9]^T$	$[8,9]^T$	$[6,11]^T$	$[4,10]^T$
Échéance de l'opération odd_{nl}	$90s$	$22s$	$17s$	$43s$
Possibilités de ressource $\mathcal{R}_{nl}$	$\{R_3\}$	$\{R_1\}$	$\{R_2\}$	$\{R_2, R_3\}$

Chaque agent i démarre à $T_{i,init} = 0$ de sa position $q_{i,init}$ à vitesse nulle (de la même manière qu'un agent démarre de sa ressource suite à la réalisation de l'opération précédente). La portée de communication est $R_{com} = 2m$ et la distance de sécurité est $d_{safe} = 0.5m$. Les résultats de comparaison entre les deux approches pour ce scénario sont donnés dans le tableau 6.8. Notons que AH et AS se réfèrent respectivement aux résultats des approches hétérarchique et supervisée. La Fig. 6.10 donne

Tableau 6.8 – Résultats numériques et comparatifs pour les deux approches

	A_1		A_2		A_3		A_4	
Borne inférieure du temps final	$7.96s$		$8.02s$		$8.09s$		$7.54s$	
Meilleure ressource initiale	R_3		R_1		R_2		R_2	
Approche	AS	AH	AS	AH	AS	AH	AS	AH
Temps final	$10.46s$	$11.11s$	$8.68s$	$10.01s$	$8.13s$	$8.09s$	$9.89s$	$9.91s$
Temps d'achèvement	$21.46s$	$22.01s$	$15.88s$	$17.21s$	$19.13s$	$19.09s$	$20.79s$	$20.81s$
Ressource choisie	R_3	R_3	R_1	R_1	R_2	R_2	R_3	R_3
Déviation par rapport à la trajectoire directe [m]	160.1	218.5	65.8	93.87	1.4	0	126.51	162.25

les trajectoires des agents pour chaque approche. Les Fig. 6.11–6.12 montrent que les contraintes de vitesse et d'évitement de collision sont respectées pour chaque agent et pour chaque approche. Comme l'agent 3 a le niveau de priorité le plus haut, il n'a pas à dévier de sa trajectoire directe et sa trajectoire est donc une ligne droite. L'agent 4, qui commence à se diriger vers la même ressource que l'agent 3, change de destination dans les deux approches grâce à la fonction d'ordonnancement. Ceci lui permet d'achever son opération plus tôt que s'il effectuerait son opération à la ressource R_2. Les agents 4 et 1 se dirigent tous deux vers la ressource R_3. Bien que ces deux agents se dirigent vers la même ressource,

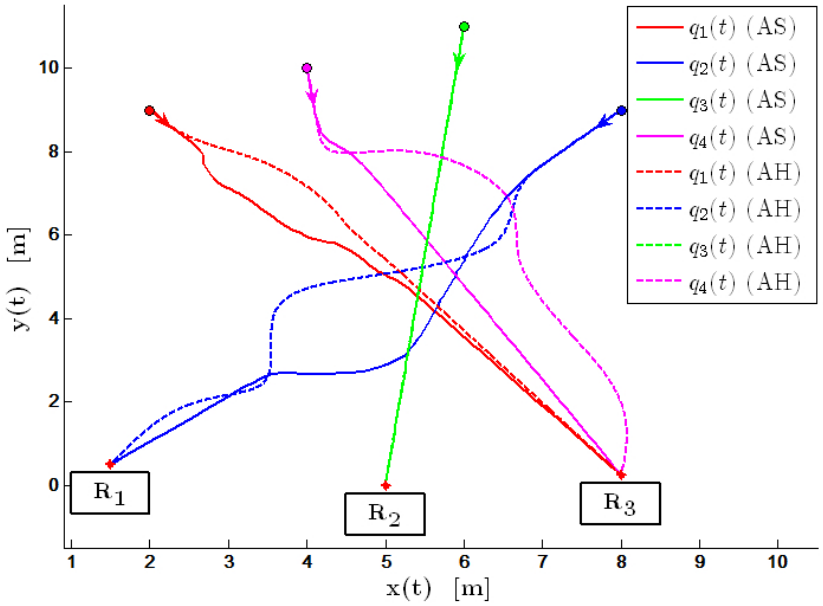


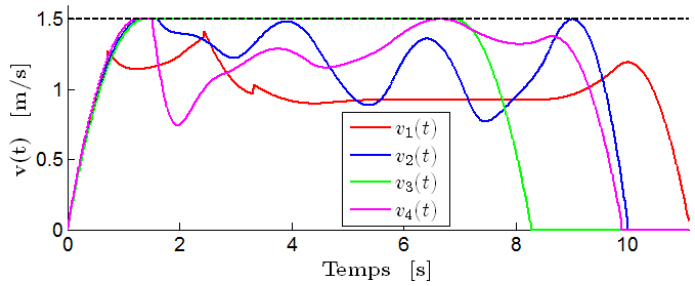
FIGURE 6.10 – Trajectoires des agents pour les approches hétérarchique (AH) et supervisée (AS)

l'agent 4 arrive à cette ressource avant que la collision ne se produise.

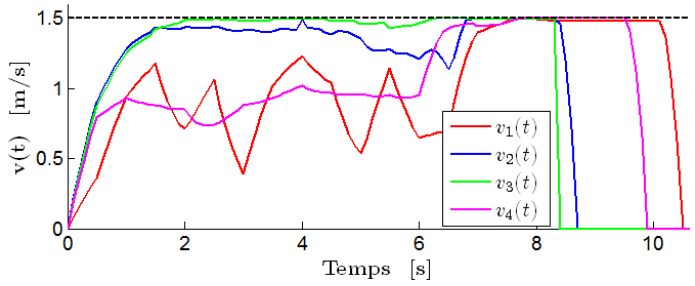
Selon le tableau 6.8, on remarque que l'approche supervisée permet à la plupart des agents d'arriver à leur ressource plus tôt par rapport à l'approche hétérarchique. Ce gain leur permet de gagner en moyenne $0.47s$ sur le temps d'achèvement. Enfin, on peut voir que la déviation par rapport à la trajectoire directe (i.e. $\int_0^{T_{i,fin}} \|q_i(t) - q^*_{ib}(t,0)\|\ dt, b = cr_{i0}$) est plus faible pour l'approche supervisée. Cette déviation est très bien illustrée sur la Fig. 6.10 avec l'agent 2. En effet, pour l'approche hétérarchique, les mises à jour de trajectoires le font "osciller" alors que pour l'architecture supervisée, il dévie au fur et à mesure pour retourner ensuite sur la trajectoire directe.

Remarque 6.1 *On peut voir sur le tableau 6.8 que la déviation de l'agent* 3 *n'est pas égale à 0 pour l'approche supervisée alors qu'elle l'est pour l'approche hétérarchique. Ceci est normal car pour l'approche hétérarchique, la trajectoire directe est gardée et il n'y a aucune mise à jour. Pour l'approche supervisée, la trajectoire directe est mise à jour graduellement au cours du temps. Bien que les trajectoires générées lors des mises à jour soient directes, il y a un petit écart par rapport à la trajectoire initiale directe. Cet écart est dû à l'approximation de la trajectoire via la courbe spline, qui ne permet pas d'obtenir exactement la trajectoire directe générée à l'instant initial.*

Les résultats sont meilleurs pour l'approche supervisée pour deux raisons. D'une part, nous avons vu dans le Chapitre 4 que l'algorithme PSO permet de générer des trajectoires plus optimales que l'algorithme utilisé pour l'approche hétérarchique. D'autre part, l'algorithme d'optimisation seul ne suffit pas à expliquer l'écart entre les déviations des deux approches. En effet, l'approche supervisée permet aux agents d'avoir des informations globales sur les conflits de collision, ce qui leur permet de

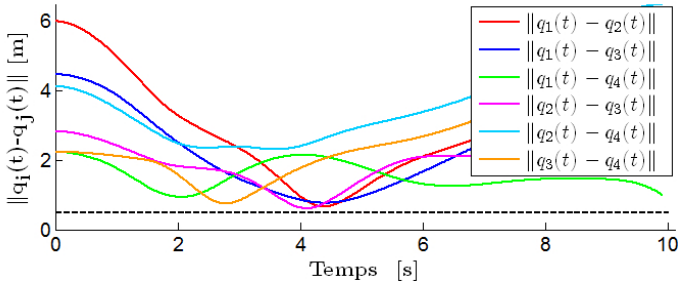


(a)

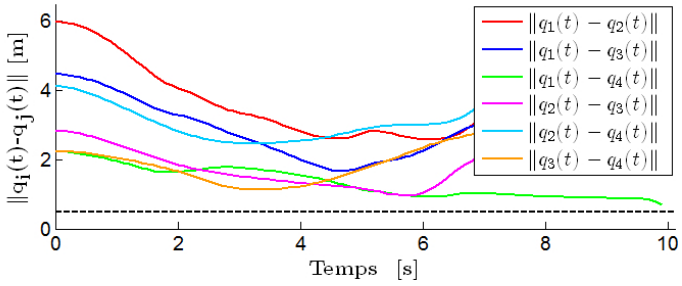


(b)

FIGURE 6.11 – Vérification de la contrainte de vitesse des deux approches : (a) hétérarchique, (b) supervisée



(a)



(b)

FIGURE 6.12 – Vérification de la contrainte de vitesse des deux approches : (a) hétérarchique, (b) supervisée

les anticiper, bien avant que les agents ne communiquent. Cet anticipation nous permet d'affirmer que le superviseur donne un aspect de proactivité dans la navigation des agents. De plus, elle permet aux agents d'entreprendre la déviation plus tôt afin d'éviter de dévier au dernier moment.

Pour finir, notons que le temps de calcul de l'approche supervisée est considérablement réduit (approximativement $118s$ pour l'approche supervisée et $1436s$ pour l'approche hétérarchique) malgré le nombre plus élevé de mises à jour de trajectoire. Ceci est principalement dû à l'utilisation de l'optimisation par essaims particulaires.

6.2.4 Résultats du problème de rendez–vous

Ici, les résultats numériques du protocole de commande distribuée décrit dans le Chapitre 5 sont présentés. Pour cela, considérons le système multi–agent décrit par (5.3.1) avec $N = 6$ agents suiveurs et un meneur virtuel référencé par 0. La topologie de communication, donnée sur la Fig. 6.13 est fixe. Le graphe de communication est connecté où la matrice correspondante H est donnée par

$$H = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & -1 \\ -1 & 3 & -1 & 0 & 0 & 0 \\ 0 & -1 & 3 & -1 & 0 & -1 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -1 \\ -1 & 0 & -1 & 0 & -1 & 3 \end{bmatrix}$$

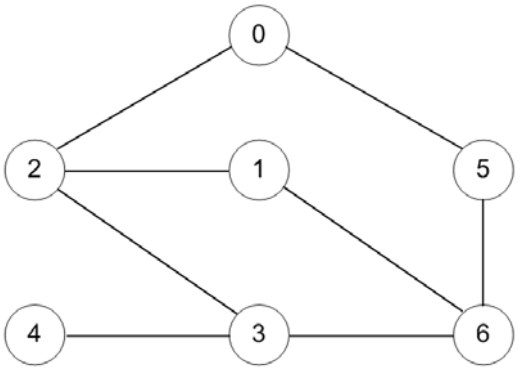


FIGURE 6.13 – Topologie de communication.

On peut voir que les agents 1, 3, 4 et 6 ne reçoivent par d'information du meneur (i.e. configuration désirée). La configuration de ce meneur est $q_0 = \left(6, -4, \frac{\pi}{6}\right)^T$.

L'objectif de commande est de résoudre le problème de rendez–vous vers un meneur immobile en un temps fixe. Pour chaque agent non holonome n ($n = 1, \ldots, 6$), de dynamique (5.3.1), la commande

Tableau 6.9 – États initiaux des agents suiveurs non holonomes

Agents	Configurations initiales	États transformés
1	$q_1 = \left(-8, -1, \frac{-\pi}{3}\right)^T$	$\xi_1 = \left(\frac{-\pi}{3}, 7.43, -3.13\right)^T$
2	$q_2 = \left(-2, 8, \frac{-2\pi}{3}\right)^T$	$\xi_2 = \left(\frac{-2\pi}{3}, 5.73, -5.93\right)^T$
3	$q_3 = \left(4, 9, \frac{-5\pi}{6}\right)^T$	$\xi_3 = \left(\frac{-5\pi}{6}, 5.79, -7.96\right)^T$
4	$q_4 = \left(6, 5, \frac{-7\pi}{6}\right)^T$	$\xi_4 = \left(\frac{-7\pi}{6}, 7.33, -2.7\right)^T$
5	$q_5 = \left(-10, 2, \frac{-5\pi}{6}\right)^T$	$\xi_5 = \left(\frac{-5\pi}{6}, 6.73, 7.66\right)^T$
6	$q_6 = \left(-7, 7, \frac{-\pi}{2}\right)^T$	$\xi_6 = \left(\frac{-\pi}{2}, 7, -7\right)^T$

u_n sera conçue à partir des informations locales (obtenues par les voisins) telle que :

$$q_n(t) - q_0 = 0 \tag{6.2.1}$$

Comme mentionné dans le Chapitre 5, la transformation (5.3.3) est appliquée dans la conception de la commande basée sur la forme chaînée (5.3.4). Dans ce cas, l'état transformé du meneur virtuel devient $\xi_0 = \left(\frac{\pi}{6}, 6.46, 3.2\right)^T$. Les configurations initiales des agents suiveurs, de type unicycle, sont fournies dans le tableau 6.9.

Rappelons que le système (5.4.1) ne vérifie pas les conditions nécessaires sur l'existence d'un retour d'état continu statique garantissant la stabilité asymptotique à l'équilibre [Brockett et al., 1983]. Par conséquent, une stratégie par commutation est appliquée.

A partir du Théorème 5.2, la stratégie de commutation (5.5.5) assure que le problème de rendez–vous à temps fixe vers une configuration fixe soit résolue. Les différents gains de commande sont réglés comme suit : $\alpha_1 = 32$, $\alpha_2 = 1600$, $\alpha_3 = 6.5$, $\alpha_4 = 0.1$, $\alpha_5 = 6.5$, $\beta_1 = 32$, $\beta_2 = 1600$, $\beta_3 = 1.2$, et $\beta_4 = 1.2$. Ainsi, on obtient les temps de commutation, qui sont indépendants des conditions initiales des agents, à l'aide de (5.5.6), comme suit :

$$\begin{cases} T_1 &= 1.1s \\ T_2 &= 2.1s \end{cases}$$

Contrairement aux protocoles de commande existants, une estimation explicite du temps de stabilisation est ici fournie sans une connaissance à priori des conditions initiales des agents, à partir de (5.5.7), i.e. :

$$T_{max} = 3.1s$$

Rappelons que le temps de stabilisation est obtenu en réglant les gains de commande α_i $(i = 1, 2, 3, 5)$

et β_i $(i = 1, \ldots, 4)$ à l'aide des équations (5.5.6)-(5.5.7). Les résultats de simulation sont illustrés par les Fig. 6.14-6.17. Les trajectoires de chaque agent, avec les coordonnées originelles (resp. coordonnées sous forme chaînée) sont données sur la Fig. 6.14 (resp. Fig. 6.15). Afin d'améliorer la compréhension des figures, les différentes étapes du protocole de commande par commutations sont les suivantes :

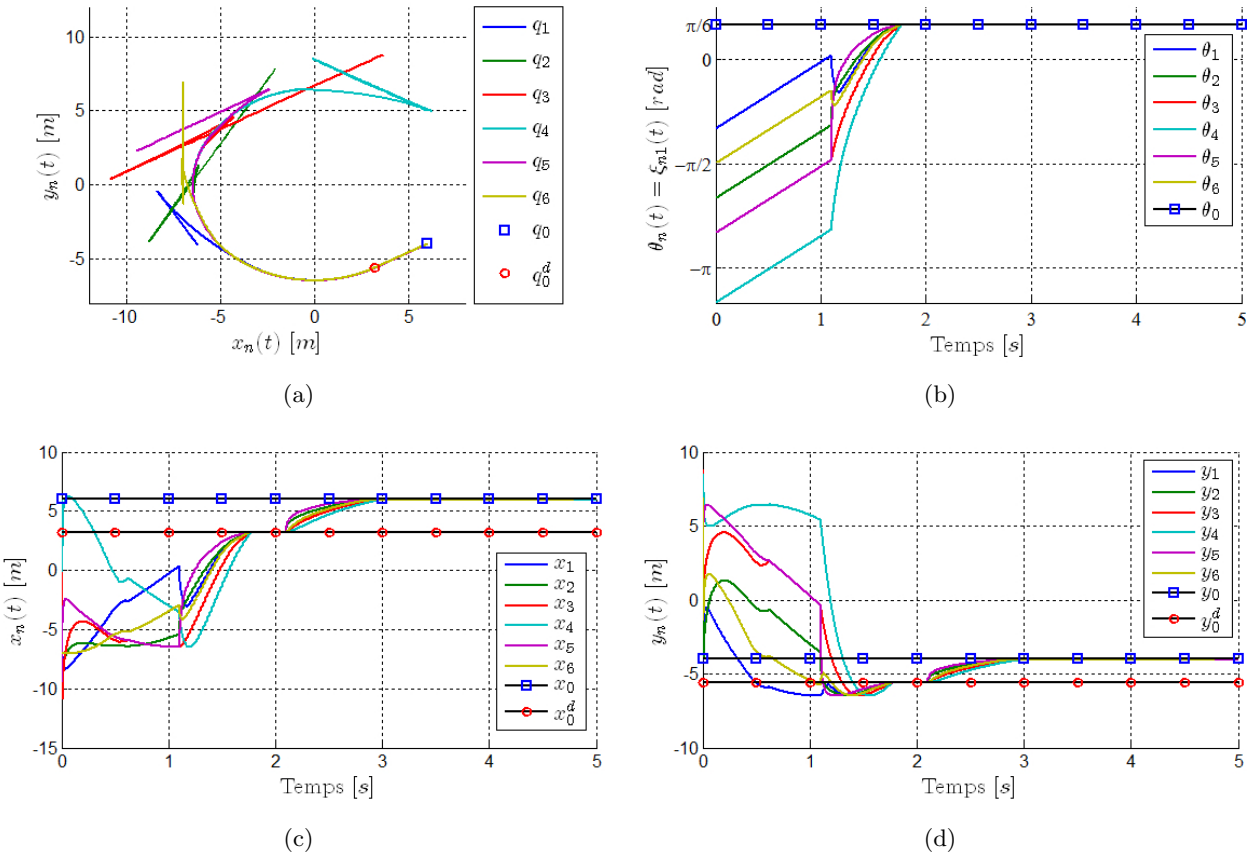
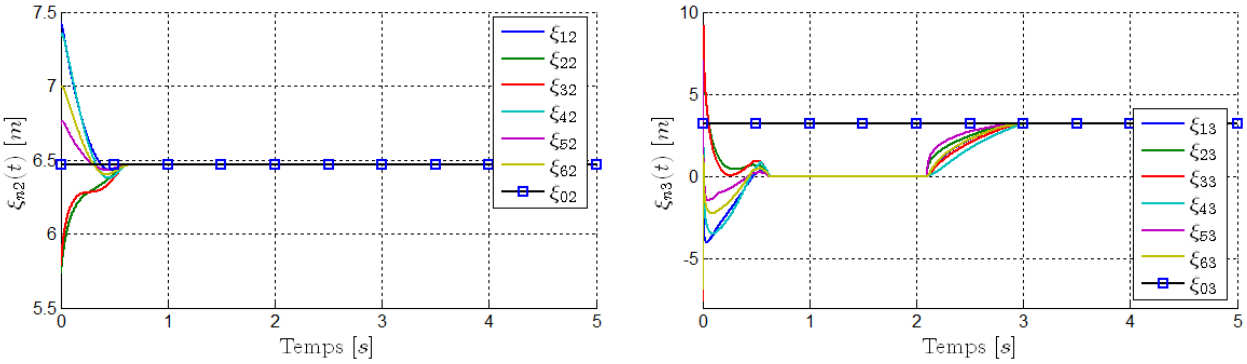


FIGURE 6.14 – Evolution des trajectoires du système pour chaque agent en coordonnées originelles (5.3.1). (a) trajectoires dans le plan (2D). (b) $\theta_n = \xi_{n1}(t)$. (c) $x_n(t)$ (d) $y_n(t)$.

- La première étape, i.e. $t \leq T_1$, est appliquée pour stabiliser en temps fixe les états ξ_{n2} et ξ_{n3} vers ξ_{02} et 0, respectivement. On remarque à l'aide des Fig. 6.15(a) et 6.15(b) que la stabilisation est achevée à $t = 0.85s$ qui est inférieur à T_1. De plus, on peut voir sur la Fig. 6.16 que la surface de glissement $s(\widetilde{\xi}_{n2}(t), \widetilde{\xi}_{n3}(t))$ converge vers zéro en un temps $2/\sqrt{\alpha_2} + 2/\sqrt{\beta_2} = 0.1s$. Notons qu'après $t = T_1$, la surface de glissement n'est plus égale à zéro car les états ξ_{n2} et ξ_{n3} évoluent du fait des commutations de la commande. Enfin, en utilisant la Fig. 6.14(a), on peut assimiler cette première étape de commande à une manoeuvre de stationnement due à la contrainte de non holonomie.

- La seconde étape, pour $t \in [T_1, T_2]$, les états ξ_{n2} et ξ_{n3} restent constants, c'est–à–dire que les agents se déplacent sur un cercle. L'état ξ_{n1} (i.e. l'orientation θ_n) converge vers θ_0 en un temps



(a) (b)

FIGURE 6.15 – Evolution des trajectoires du système pour chaque agent sous forme chaînée (5.3.4). (a) $\xi_{n2}(t)$. (b) $\xi_{n3}(t)$.

1.8s comme le montre la Fig. 6.14(b). On peut noter que l'estimation du temps de stabilisation est assez précise. Il est évident, en utilisant la Fig. 6.14(a), que durant cette étape, chaque agent se déplace sur un cercle centré en 0 et de rayon $\sqrt{(x_0^d)^2 + (y_0^d)^2}$ jusqu'à avoir résolu le problème de rendez–vous à la configuration q_0^d en un temps fixe.

- La dernière étape achève la stabilisation de ξ_{n3} vers ξ_{03} (voir Fig. 6.15(b)) tout en maintenant constant les états ξ_{n1} et ξ_{n2} en un temps 2.9s (voir Fig. 6.14(b)-6.15(a), respectivement). Dans ce cas, la dynamique de l'état transformé est réduite à (5.5.13). Ceci signifie que les agents se déplace en ligne droite pour achever le rendez–vous à la configuration du meneur q_0 comme le montre la Fig. 6.14(a).

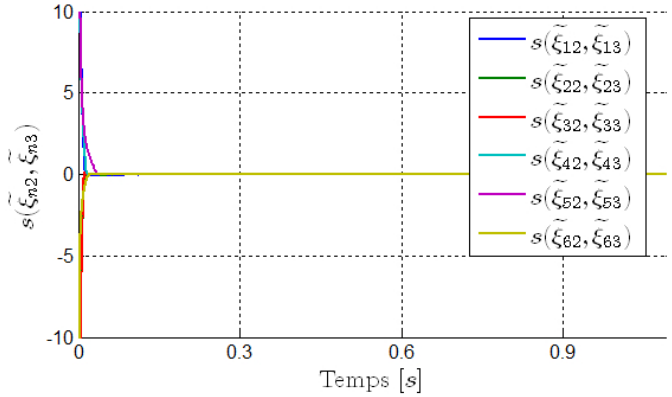
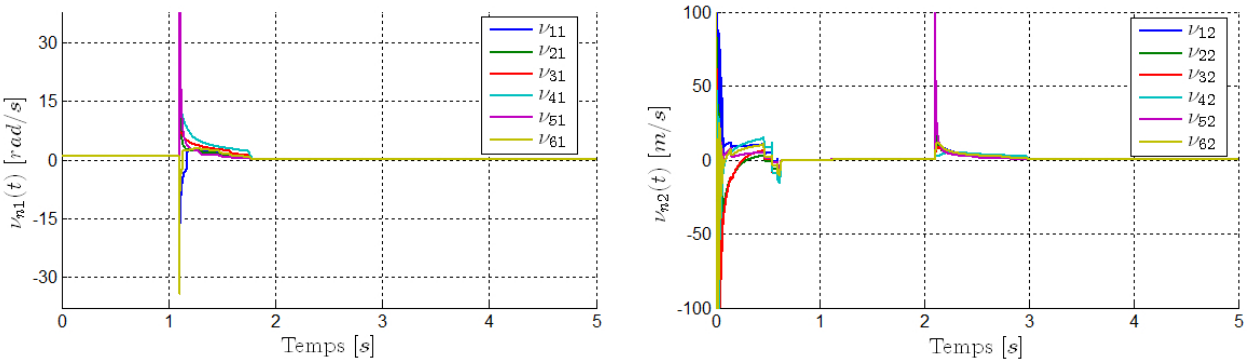


FIGURE 6.16 – Evolution de la surface de glissement $s(\widetilde{\xi}_{n2}(t), \widetilde{\xi}_{n3}(t))$ pour chaque agent (5.3.1).

Les entrées ν_{n1} et ν_{n2} pour chaque agent sont illustrés sur la Fig. 6.17. Notons que les amplitudes des entrées peuvent prendre des valeurs élevées lors des transitions entre les étapes pour permettre

une convergence rapide des surfaces de glissement $(s(\widetilde{\xi}_{n2}(t), \widetilde{\xi}_{n3}(t))$, $\widetilde{\xi}_{n3}$ et $\widetilde{\overline{x}}_n)$ données par les trois étapes du protocole de commande distribuée (5.5.5). Ces résultats montrent bien que le protocole de commande proposé assure un rendez–vous des agents vers un meneur en un temps fixe. Ceci confirme les résultats théoriques obtenues à partir du Théorème 5.2.



(a) (b)

FIGURE 6.17 – Evolution des entrée du système (a) $\nu_{n1}(t)$. (b) $\nu_{n2}(t)$.

Remarque 6.1 *Notons que les contraintes de non holonomie sont respectées au vu des trajectoires spécifiques du système obtenues par la stratégie de commutation proposée (voir Fig. 6.14(a)). Le déroulement du scénario de rendez–vous est donné à l'aide d'une vidéo disponible via le lien vidéo, permettant de monter l'évolution de ces trajectoires spécifiques ainsi que les mouvements des agents à chaque étape du protocole de commande. La vidéo permet aussi de souligner la convergence vers la configuration du meneur en un temps* $T_{max} = 3.1s$.

6.3 Résultats expérimentaux et discussion

Dans cette section, des résultats expérimentaux seront fournis pour montrer la faisabilité de l'algorithme de navigation avec architecture supervisée qui nous a donné de très bons résultats numériques.

6.3.1 Résultats expérimentaux

Afin de tester expérimentalement les algorithmes de planification de trajectoire, un scénario avec quatre agents mobiles et trois ressources (virtuelles) est mis en place. Ce scénario est semblable à celui utilisé pour la comparaison entre les approches (en termes de niveau de priorité et de choix de ressource des agents).

En guise d'agents, des LEGO Mindstorms, comme ceux représentés sur la Fig. 6.18, seront utilisés pour le scénario expérimental. Ces robots se déplacent à l'aide de chenilles utilisées à la place de roues

pour limiter les glissements où chaque chenille est commandée par un moteur à courant continu. L'entrée de ces moteurs est bornée et permet aux agents de se déplacer à une vitesse maximale $v_{max} = 0.15m/s$. La forme géométrique des robots nous donne une distance de sécurité égale à $d_{safe} = 0.2m$. La portée de communication est fixée à $R_{com} = 0.4m$. Le scénario expérimental mis en place permet aux agents

FIGURE 6.18 – Les robots Lego Mindstorms utilisés pour les résultats expérimentaux.

de naviguer sur une aire de $4m^2$ environ ($2m \times 1.8m$). Ce scénario est dans un premier temps testé en simulation. Les trajectoires des agents sont données sur la Fig. 6.19. A l'aide de la Fig. 6.20, on peut voir que les contraintes de vitesse maximale et d'évitement de collision sont satisfaites. Les paramètres des agents ainsi que les résultats de ce scénario expérimental sont fournis dans le tableau 6.10.

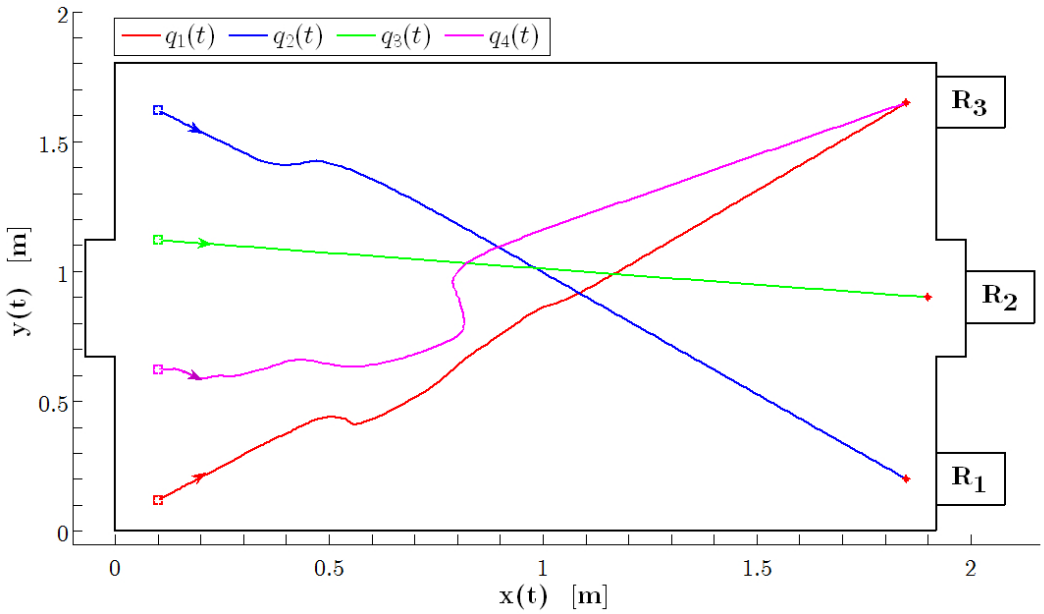
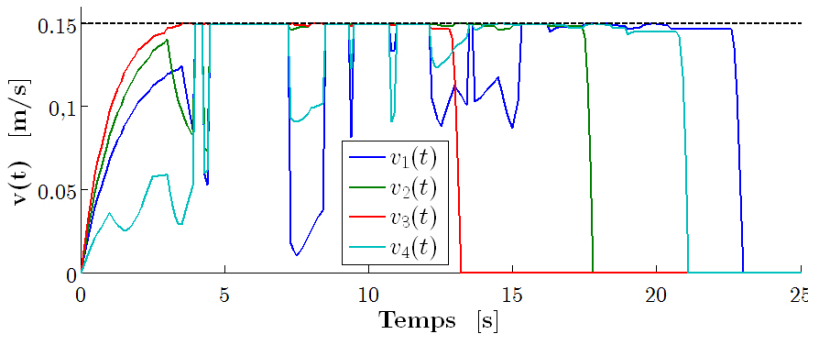
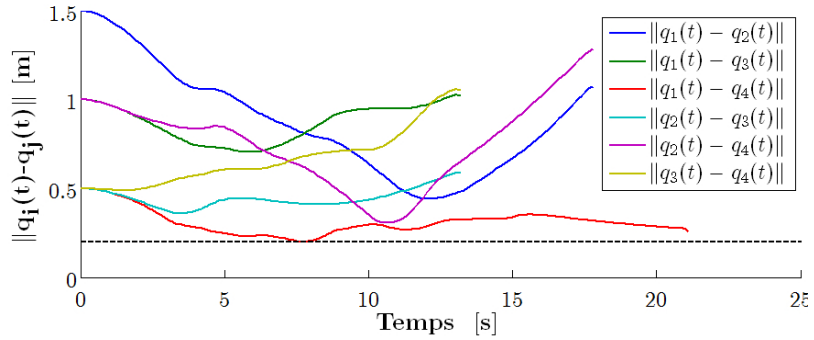


FIGURE 6.19 – Trajectoires des agents en simulation pour le scénario expérimental.



(a)



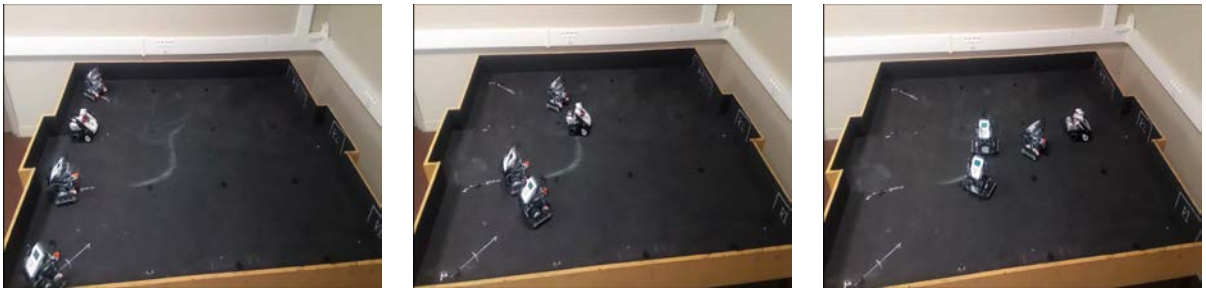
(b)

FIGURE 6.20 – Vérification des contraintes en simulation pour le scénario expérimental : (a) vitesse, (b) évitement de collision

Tableau 6.10 – Paramètres initiaux des agents et résultats.

	Position initiale	Possibilité de ressource	Temps de traitement	Échéance opération	Temps final	Temps d'achèvement
Robot 1	$[0.1, 0.125]$	$\{R_3\}$	$5.4s$	$76.1s$	$25.44s$	$31.12s$
Robot 2	$[0.1, 0.625]$	$\{R_1\}$	$4.3s$	$30.8s$	$17.76s$	$22.07s$
Robot 3	$[0.1, 1.125]$	$\{R_2\}$	$5s$	$23.8s$	$13.14s$	$18.14s$
Robot 4	$[0.1, 1.625]$	$\{R_2, R_3\}$	$4.7s$	60.2	$21.02s$	$25.72s$

Les trajectoires planifiées, permettant aux agents d'atteindre leur ressource respective, sont calculées pour chaque robot. Les explications des résultats expérimentaux sont données dans la vidéo disponible sur le lien suivant : Lien. Quelques instantanés du déroulement de cette vidéo sont illustrés par la Fig. 6.21. On peut voir les configurations initiales et finales de chaque agent sur les instantanés (a) et (f). En comparant les trajectoires réelles des agents sur la vidéo et les trajectoires obtenues en simulation, on remarque qu'il peut y avoir quelques écarts. Ceux–ci sont justifiables car la contrainte de non holonomie n'est pas prise en compte dans les algorithmes de planification de trajectoire, ce qui rend ces trajectoires moins précises par moment comme nous l'avons fait remarquer dans le Chapitre 2. Cependant, les collisions sont tout de même évitées, ce qui nous permet de démontrer la faisabilité de l'algorithme de navigation.



(a) Départ (b) Après 5s. (c) Après 10s.

(d) Après 15s. (e) Après 20s. (f) Arrivée

FIGURE 6.21 – Instantanés des robots se dirigeant vers leur ressource en évitant les collisions.

6.3.2 Discussion : implémentation des AGVs en milieu industriel

Dans le Chapitre 1, nous avons présenté les principaux avantages lorsque des AGVs naviguent librement dans un système de production. Nous avons aussi vu que ceci nécessite des algorithmes de navigation où il est préférable d'inclure la fonction d'ordonnancement pour diverses raisons. Dans les Chapitres 3 et 4, deux algorithmes de navigation, qui combinent planification de trajectoire et ordonnancement, ont été proposés. Les résultats donnés pour ces algorithmes ont montré l'avantage de la fonction d'ordonnancement mais aussi que les approches peuvent être sensibles au phénomène de myopie.

Cependant, les deux algorithmes de navigation ont été soumis à un certain nombre d'hypothèses données dans le Chapitre 2. Afin que de tels algorithmes puissent être utilisés dans des ateliers de production réels, il est nécessaire de discuter sur ces hypothèses et de voir leur impact sur les comportement des agents. De plus, les capacités des AGVs doivent aussi être discutées puisqu'elles interviennent dans leur comportement pour un système de production. Cette section est, par conséquent, divisée en deux parties. D'une part, nous discuterons des hypothèses qui influent sur le comportement des agents, principalement sur celles qui jouent un rôle sur l'ordonnancement. D'autre part, une discussion sur les capacités des AGVs, nécessaire pour le déploiement de ceux–ci dans un système de production, sera présentée.

Discussion sur les hypothèses

Dans le Chapitre 2, un ensemble d'hypothèses a été donné dont certaines jouent un rôle crucial dans l'ordonnancement réalisé par les agents lors de la navigation. Nous proposons ici d'en rappeler les plus importantes, c'est–à–dire celles qui peuvent rendre l'ordonnancement des agents plus ou moins efficace.

- **Les approches proposées sont centrées sur l'achèvement d'une seule opération et non du produit transporté :** Cette hypothèse a été posée pour se focaliser sur les algorithmes de navigation. Cependant, ne traiter qu'une seule opération peut mener au problème de minimum local. En effet, une ressource peut être optimale (en termes de temps pour l'achever) pour l'opération en cours, mais pas nécessairement pour l'ensemble des opérations. Par exemple, lorsqu'une autre ressource peut effectuer deux opérations successives ou lorsque la ressource choisie est éloignée de celles pouvant effectuer l'opération suivante.

  Pour éviter ce problème de minimum local, il serait utile d'informer les agents pour les opérations suivantes. Cependant, ceux–ci ne pourraient pas savoir a priori le temps de transport pour effecter l'ensemble des opérations suivantes ainsi que le temps d'attente aux ressources (à cause de la myopie temporelle). Par conséquent, il serait utile que le haut–niveau (le superviseur par exemple) leur donne des indications pour qu'ils puissent mieux choisir la ressource.

- **Chaque produit a une séquence d'opérations à suivre où les opérations sont traitées une à une :** Cette hypothèse stipule que les opérations se font dans un certain ordre. Si elles ne l'étaient pas, l'achèvement du produit pourrait être plus rapide car l'agent pourrait faire plusieurs opérations à une même ressource, ce qui réduirait le nombre de déplacement entre ressources. Cependant, le problème d'ordonnancement à traiter par les agents serait bien plus complexe puisqu'ils devraient choisir entre un plus grand nombre de ressources.

- **Le cahier des charges fourni aux agents, incluant l'ensemble des ressources et l'échéance de l'opération, est supposé faisable et n'est pas modifié sur l'intervalle de temps considéré :** Pour cette hypothèse, le cahier des charges est supposé faisable, c'est–à–dire qu'au départ de l'agent, l'opération peut être achevée avant l'échéance. La raison pour laquelle il est supposé faisable repose sur le fait qu'il ne peut pas être modifié (et donc, s'il est non faisable initialement, il le restera toujours). Dans cette situation, l'agent gère l'achèvement avant l'échéance de son opération à l'aide du mécanisme de priorité. Par contre, si beaucoup d'AGVs sont déployés dans le système de production (plus de conflits lors du transport, temps d'attente aux ressources plus long), ou si certaines ressources ne sont pas disponibles, il est possible que l'échéance ne soit plus respectée. En ajoutant la possibilité de modifier le cahier des charges, l'agent pourrait demander une mise à jour de son échéance, principalement si celle–ci est impossible à respecter. De plus, si une ressource tombe en panne, le niveau M.O.R.M. pourrait enlever cette même ressource de l'ensemble des possibilités.

- **Les files d'attente des ressources sont supposées être à capacité infinies :** Ici, il est supposé que le nombre d'agents à chaque ressource n'est pas limité, en faisant implicitement l'hypothèse que l'agent ne se dirigera pas vers une ressource ayant un long temps d'attente. Dans les atelier de production, les files d'attente sont limitées, ce qui impacte l'ordonnancement puisqu'un agent doit se renseigner sur la disponibilité de la ressource avant de pouvoir s'y rendre. De plus, le mécanisme de priorité joue aussi un rôle important puisqu'il permettrait de savoir quel agent/produit a le droit de réquisitionner la ressource. Pour traiter une file d'attente limitée, il serait donc préférable de rendre les ressources actives et les faire coopérer avec les agents pour améliorer leur ordonnancement.

- **Les perturbations liées aux ressources (pannes machine, maintenance) ne sont pas considérées :** De la même manière que dans les ateliers à cheminements multiples, les perturbations jouent un rôle important sur la fonction d'ordonnancement. En effet, une panne machine impacte les agents qui l'ont choisie. Ceux–ci doivent choisir une autre ressource. De plus, il est possible qu'en choisissant une autre ressource, l'opération du produit transporté par l'agent ne puisse plus être achevée avant l'échéance. Par conséquent, en considérant les perturbations aux ressources, il faut aussi pouvoir modifier le cahier des charge des agents le cas échéant.

- **La gestion de l'énergie de la batterie des agents (charge, décharge) n'est pas traitée :** La contrainte énergétique des AGVs a un fort impact sur l'ordonnancement. Dans un premier temps, il faut mettre en oeuvre des moyens assurant la recharge de la batterie des agents, en ajoutant par exemple des ressources de recharge ou de changement de batterie (battery swapping). Ensuite, il faut assurer que le niveau de batterie soit suffisant pour effectuer une opération à une ressource. Il est nécessaire de savoir si l'état de charge de la batterie peut permettre deux transports successifs à chaque moment. En effet, il faut que l'agent soit capable d'atteindre la ressource pour achever l'opération et par la suite rejoindre la ressource de recharge d'énergie. Par conséquent, la contrainte énergétique peut être considérée comme un indice de performance, ce qui change le mécanisme de priorité. Ainsi, un agent peut avoir un niveau de priorité plus haut si par exemple, son niveau de batterie est plus faible. Enfin, la contrainte énergétique change la manière dont le cahier des charges est conçu car il est nécessaire de savoir à priori si le niveau de charge de la batterie de l'AGV est suffisant pour achever le produit. Sinon, il faut adapter ce cahier des charges en incluant l'opération de recharge dans la séquence d'opérations du produit.

Discussion sur les capacités des AGVs

Les capacités des AGVs sont très importantes lorsqu'on parle de navigation car un ensemble de fonctions est nécessaire (planification de trajectoire, suivi, etc.). On parle alors d'*embarquabilité*. Dans la littérature, il n'y a pas de définition précise de l'embarquabilité [Malinowski et Yu, 2011]. Dans la plupart des dictionnaires, sa définition est souvent donnée comme “capacité à embarquer” ou encore

"caractère de ce qui est susceptible d'être embarqué". Nous proposons la définition suivante :

> *L'**embarquabilité** est la capacité de pouvoir embarquer des dispositifs de calcul, de communication et d'énergie suffisamment puissants dans un AGV afin de pouvoir accomplir les tâches qui lui sont assignées.*

De cette définition, on remarque que l'embarquabilité dépend des tâches demandées, montrant le lien avec les fonctions des AGVs permettant d'accomplir les tâches d'une manière intelligente. Lorsque les AGVs naviguent dans un système de production, il y a un certain nombre de problèmes que les AGVs doivent traiter par eux–même.

- **La navigation** est, bien entendu, la principale fonction car elle nécessite d'autres sous-fonctions permettant de traiter la localisation, la perception, la stratégie de commande, etc. Ces fonctions sont requises afin que les algorithmes de navigation conçus soient faisables, en termes d'évitement de collisions et de respect des contraintes physiques, et robustes face aux perturbations et/ou incertitudes.

- **La communication** est très importante dans un cadre multi–AGVs car les informations entre ceux–ci doivent être transmises très rapidement. De plus, l'application aux systèmes manufacturiers implique des communications entre le système et les AGVs, permettant de changer les ordres de fabrication ou donner des retours sur la production par exemple.

- **L'énergie** a un impact majeur sur les AGVs, surtout lorsqu'ils sont utilisés pour faire de la production en masse. Il est nécessaire de prendre en compte la charge/décharge des batteries lors du transport des produits.

- **Les facteurs humains** peuvent aussi avoir de l'influence sur le comportement des AGVs et sur les décisions qu'ils prennent. En effet, lors d'opérations de maintenance ou autre, il est possible que les opérateurs se déplacent parmi les AGVs. La sécurité des opérateurs doit donc être assurée.

Tous ces points doivent être pris en compte par les AGVs à l'aide de fonctions qui doivent être suffisamment intelligentes. Cependant, l'embarquabilité peut empêcher les AGVs d'améliorer cette intelligence et le nombre de fonctions qu'ils peuvent assurer peut être limité à cause de raisons techniques voire économiques.

On remarque qu'il existe un compromis entre l'intelligence des AGVs et l'embarquabilité. Il peut être illustré par une balance où l'intelligence et l'embarquabilité, dépendant de façon respective aux fonctions des AGVs et de leur capacité, sont de chaque côté comme le montre la Fig. 6.22. On remarque sur cette figure qu'améliorer l'intelligence de l'AGV revient à augmenter le poids du côté gauche de la balance. Cependant, il est nécessaire de compenser ce poids en augmentant les capacités de l'AGV. De nos jours, les capacités en termes de calcul pour les systèmes embarqués peuvent être très élevées.

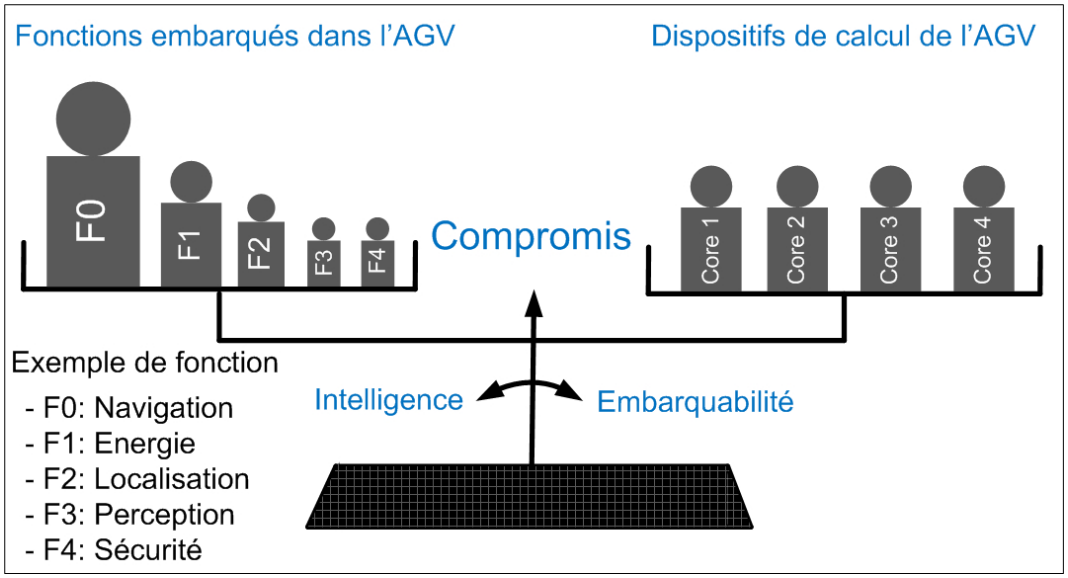


FIGURE 6.22 – Illustration du compromis entre intelligence et embarquabilité.

De ce fait, on pourrait croire que la seule limite à l'intelligence est liée à l'embarquabilité technique, c'est-à-dire dépendant uniquement de l'évolution des dispositifs pouvant être embarqués. Cependant, le point de vue économique a une plus grande importance car le déploiement des AGVs en industrie a un coût. L'amélioration des capacités des AGVs les rendent donc plus coûteux. Par conséquent, ce coût a une grande influence car le déploiement des AGVs en industrie doit être économiquement rentable, permettant un bon retour sur investissement.

Ce compromis a une grande influence sur la manière dont les AGVs peuvent être utilisés. En effet, lorsque l'on permet un haut niveau d'intelligence, on peut traiter plus facilement des situations complexes. Cependant, il est possible que les AGVs n'aient pas la capacité d'avoir cette intelligence, les empêchant d'appliquer les fonctions correctement et pouvant mener à une surcharge de calcul qui rend l'approche infaisable. Réciproquement, ayant des capacités limitées, les AGVs ne peuvent être utilisés que pour effectuer des tâches simples. Ce manque d'intelligence les empêche de traiter des solutions plus complexes.

Ces propos sont très bien illustrés par la manière dont les chercheurs et les industriels traitent ce compromis. En effet, les industriels préfèrent utiliser des fonctions simples et facilement embarquables, sans pour autant utiliser les AGVs d'une manière très intelligente. A l'inverse, les chercheurs se focalisent plutôt sur la manière de rendre les AGVs intelligents pour traiter des problèmes pouvant être très complexes. Cependant, la complexité de résolution est parfois difficilement compensable par l'évolution technique ou économique.

6.4 Conclusion

Dans ce chapitre, les résultats sur la planification de trajectoire avec architecture hétérarchique (Chapitre 3) et architecture supervisée (Chapitre 4) ainsi que sur la commande coopérative (Chapitre 5) ont été présentés.

D'abord, les résultats numériques de la planification de trajectoire ont permis dans un premier temps de montrer les avantages de la combinaison entre planification de trajectoire et ordonnancement. Pour l'approche hétérarchique, nous avons remarqué que la myopie est très présente, surtout dans le choix de la ressource. En effet, certains agents changent de manière successive de ressource pour au final revenir sur le choix initial. Les résultats pour l'architecture supervisée ont permis de montrer que l'anticipation donnée aux agents par le superviseur réduit cette myopie. Cependant, cette myopie reste présente. De plus, un scénario comparatif a permis de montrer l'amélioration des performances des agents pour l'architecture supervisée. Pour la commande coopérative, un scénario de rendez–vous a été proposé. Les résultats ont montré que le protocole de commande proposé assure bien le rendez–vous des agents vers un meneur en un temps fixe, ce qui a permis de justifier les résultats théoriques développés dans le Chapitre 5.

Ensuite, des résultats expérimentaux ont été proposés sur la planification de trajectoire où l'architecture supervisée a été testée à l'aide de robots LEGO Mindstorms. Ceux–ci ont permis de montrer la faisabilité de l'approche supervisée mais aussi de voir le manque de précision provenant de la simplification du modèle dynamique des agents. Malgré ce manque de précision, les contraintes d'évitement de collision sont respectées. Enfin, des discussions sur deux points ont été proposées. D'une part, une discussion sur les différentes hypothèses a permis de montrer les changements dans les comportements des agents, plus particulièrement sur l'ordonnancement. D'autre part, nous avons discuté sur les capacités des AGVs qui engendrent aussi plusieurs problématiques.

Conclusion générale et perspectives

Les travaux présentés dans ce mémoire sont dévolus au problème de la navigation de véhicules guidés autonomes (AGVs) dans un environnement manufacturier. La prise en compte de l'aspect manufacturier a plusieurs conséquences sur la navigation des AGVs car ceux–ci sont contraints de suivre un cahier des charges qui impose des délais en relation avec la production. Le problème de navigation considéré est donc très ambitieux puisqu'il couvre deux thématiques de recherche, à savoir la navigation de systèmes multi–agents et l'étude des systèmes de production (utilisant des outils de la recherche opérationnelle). L'intérêt majeur étant de parvenir à une mutualisation, nous avons cherché des solutions pouvant combiner ou harmoniser certains outils de ces deux thématiques.

- Dans le premier chapitre, nous avons introduit les principaux concepts de chaque domaine de recherche. Pour les systèmes de production, la notion de pilotage, ses différentes fonctions ainsi que ses architectures ont été présentées. De plus, nous avons donné un aperçu des travaux sur le pilotage des systèmes de production pour plusieurs types d'ateliers. Ensuite, nous avons donné un rapide tour d'horizon des outils permettant la navigation des AGVs, principalement sur la planification de trajectoire et la commande coopérative. Enfin, nous avons introduit la mutualisation, qui se définit comme la manière dont les outils de chaque domaine peuvent s'harmoniser ou se combiner. De plus, nous avons vu les avantages de cette mutualisation tels que l'amélioration de la flexibilité ou la réduction des temps de transport entre ressources, ainsi que certains impacts qui nous ont menés à combiner la planification de trajectoire avec l'ordonnancement.

- Le chapitre 2 est consacré à la formalisation du problème de planification de trajectoires des AGVs qui est résolu à l'aide de deux algorithmes différents dans les chapitres 3 et 4. L'objectif des AGVs et les principales hypothèses y ont été présentés. De plus, nous avons introduit la combinaison entre la planification de trajectoire et l'ordonnancement. Cette combinaison permet de mettre à jour la trajectoire lors de la navigation tout en choisissant la ressource permettant d'achever le produit au plus tôt. Ce chapitre s'est achevé en présentant les courbes splines utilisées pour décrire les trajectoires sous forme paramétrique.

- Dans le chapitre 3, une première solution du problème de planification de trajectoires des AGVs a été proposée. Pour celle–ci, une architecture hétérarchique a été présentée où chaque AGV doit de lui–même planifier une trajectoire, ordonnancer le produit pour l'opération en cours tout en coopérant avec ses voisins pour résoudre les conflits locaux. Pour pouvoir achever l'opération d'une manière efficace, la planification de trajectoire est combinée avec l'ordonnancement afin que l'AGV puisse changer de destination et mettre à jour sa trajectoire selon l'évènement rencontré. Le mécanisme permettant la coopération entre agents consiste à négocier l'ordre de priorité à l'aide d'une approche sociale calculée à partir d'un indice de performance lié à l'opération du produit transporté par l'AGV.

- Comme les architectures hétérarchiques souffrent du problème de myopie, le chapitre 4 est focalisé sur la manière de la réduire à l'aide d'une architecture supervisée. Pour cette seconde solution du problème de planification de trajectoires, un superviseur a été proposé pour d'une part résoudre les conflits à l'arrivée aux ressources et d'autre part, pour permettre aux AGVs d'anticiper les conflits pouvant engendrer des collisions. De plus, le superviseur donne aux agents l'autorisation d'ordonnancer leur produit à chaque mise à jour. En effet, pour cette approche, les mises à jour ne se font pas en présence d'évènements ; tous les agents mettent à jour leur trajectoire en même temps et graduellement au cours du temps. Pour cela, une planification de trajectoire en deux étapes a été proposée. La première étape permet de planifier les intentions de l'AGV à l'aide des informations relatives aux conflits données par le superviseur. Ces informations sont très utiles car elles permettent de mieux choisir la ressource lorsque l'ordonnancement est permis. La seconde étape est locale et consiste à assurer l'évitement de collision avec les AGVs avoisinants. Au vu du nombre de trajectoires à planifier à chaque mise à jour, une résolution du problème par optimisation par essaims particulaires a été proposée pour réduire les temps de calcul.

- Le chapitre 5 a été consacré à la navigation par commande coopérative où les AGVs se donnent rendez-vous à une certaine configuration en temps fixe, c'est–à–dire qui ne dépend pas des configurations initiales des AGVs. Après avoir rappelé l'ensemble des outils et concepts nécessaires, le problème de rendez–vous a été formulé. Il consiste à faire converger l'ensemble des robots à une configuration donnée. Comme ce type de commande est de bas niveau, un modèle d'agent non holonome a été choisi afin d'avoir une dynamique plus réaliste. En contrepartie, une stratégie par commutations a dû être développée. Avant de présenter le protocole de commande, la stabilisation d'un agent mis sous forme chaînée a été introduite. Celle–ci a ensuite été étendue dans le développement théorique permettant de résoudre le problème de rendez–vous pour une flottille d'agents.

- Dans le chapitre 6, nous avons donné les résultats numériques associés aux chapitres 3, 4 et 5 pour montrer la faisabilité des approches proposées. Pour la planification de trajectoire, ils ont aussi montré les avantages de l'aspect d'ordonnancement combiné à la planification mais aussi les impacts de la myopie dans la navigation des agents, surtout pour l'approche hétérarchique. Une comparaison entre les architectures hétérarchique et supervisée a permis de montrer le gain de performances induit par la détection des conflits par le superviseur. De plus, des résultats expérimentaux ont été donnés pour montrer la faisabilité de l'algorithme de navigation en considérant l'approche supervisée. Enfin, nous sommes revenus sur les principales hypothèses données dans le chapitre 2 ainsi que sur les capacités des AGVs.

Perspectives

Malgré les résultats prometteurs obtenus à l'aide des algorithmes de navigation proposés, il reste un long chemin à parcourir avant de pouvoir appliquer de telles méthodologies dans l'industrie, particulièrement pour la planification de trajectoire. La section de discussion donnée dans le chapitre 6 a montré l'importance des capacités des AGVs dans la conception d'un système de production flexible basé sur AGVs.

La conception d'un tel système dépend avant tout des objectifs de production et du cahier des charges imposé. Pour répondre à ces prérogatives, les AGVs doivent traiter une ou plusieurs des problématiques énumérées dans le chapitre 6. Afin d'améliorer la démarche de conception, il serait préférable de suivre une procédure spécifique orientée vers le compromis intelligence/embarquabilité. En se basant sur les travaux de cette thèse, une démarche de conception est proposée sur la Fig. 6.23 pour orienter nos perspectives de recherche.

Cette procédure contient plusieurs étapes de conception et de validation par tests où chaque étape, décrites ci–après, a son propre rôle.

- *Mise en place des objectifs des AGVs et hypothèses* : Pour cette première étape, le problème est mis en place en définissant l'objectif des AGVs et en posant les différentes hypothèses qui rendent le problème plus simple. Ces hypothèses peuvent être reconsidérées dans les prochaines étapes pour pouvoir traiter de plus amples problèmes.

- *Assignement des fonctions des AGVs et conception de l'architecture* : La seconde étape consiste à définir les différentes fonctions des AGVs permettant de répondre à l'objectif donné. Les fonctions que l'AGV ne peut pas couvrir doivent être traitées par les autres entités du système de production (le superviseur ou les ressources par exemple). De plus, l'architecture de pilotage doit être conçue pour empêcher les conflits entre entités.

- *Conception détaillée des fonctions intelligentes des AGVs* : Les fonctions des AGVs, incluant celles permettant leur navigation, sont détaillées dans cette étape. Si les tests d'embarquabilité ne

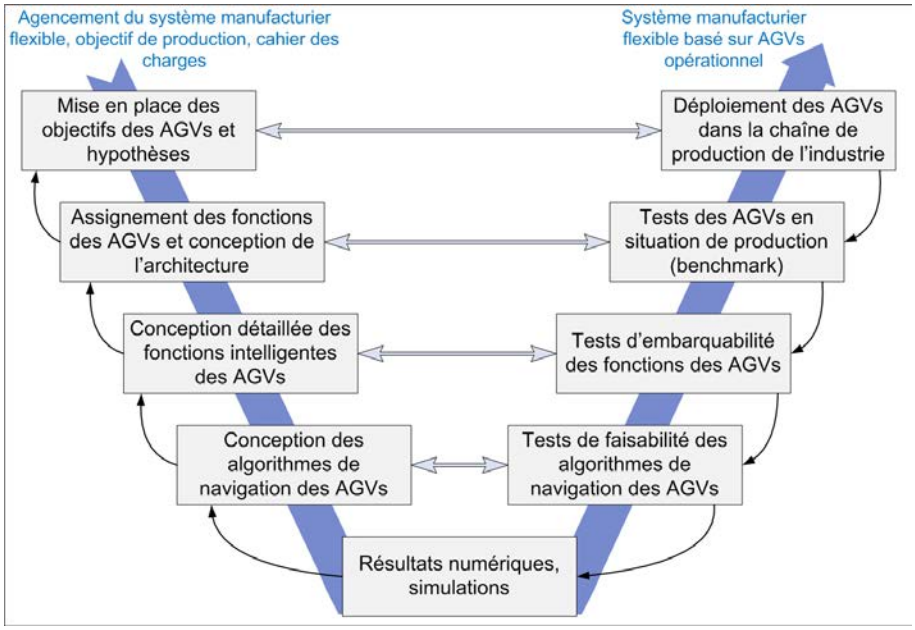


FIGURE 6.23 – Diagramme de la procédure de conception proposée

sont pas satisfaisants (i.e. mauvais compromis intelligence/embarquabilité), certaines fonctions de l'AGV devront être déportées sur d'autres entités, incluant celles du haut–niveau de pilotage.

- *Conception des algorithmes de navigation des AGVs* : Après avoir défini les fonctions des AGVs, les outils de navigation doivent être conçus comme le planificateur de trajectoire, les lois de commande, les algorithmes utilisés (par exemple algorithme A^*, méta heuristiques, ...)

- *Résultats numériques, simulations* : Ces résultats préliminaires permettent de donner un aperçu de la faisabilité de l'approche conçue, de l'intelligence des AGVs et des performances qu'ils permettent. Lorsque les résultats montrent que l'approche est infaisable, une analyse en profondeur doit être faite pour trouver quelles étapes doivent être modifiées (hypothèses, conceptions des fonctions de l'AGV, ...). Si l'intelligence des AGVs ou les performances qu'ils apportent ne sont pas suffisants pour répondre aux besoins, l'assignement des fonctions ou les algorithmes doivent être modifiés.

- *Tests de faisabilité des algorithmes des AGVs* : Dans cette étape, les algorithmes de navigation sont testés pour vérifier si les trajectoires fournies par l'algorithme de navigation sont faisables en utilisant des AGVs, des robots ou des prototypes. Si elles ne le sont pas, les algorithmes de navigation doivent être améliorés en réitérant les deux dernières étapes.

- *Tests d'embarquabilité des fonctions des AGVs* : Après avoir codé et embarqué toutes les fonctions des AGVs, l'embarquabilité est testée. Un scénario expérimental doit être conçu pour tester les fonctions des AGVs en les laissant accomplir leur mission. S'ils n'en sont pas capables, les fonctions doivent être modifiées.

- *Tests des AGVs en situation de production* : Lorsque l'embarquabilité est vérifiée, les fonctions des AGVs doivent être testées sur de plus grandes instances. Pour cela, un scénario de production complet (par exemple un benchmark [Trentesaux et al., 2013]) doit être mis en place. Ce scénario peut nécessiter d'autres fonctions pour couvrir d'autres problèmes non traités dans les tests précédents comme l'énergie des AGVs. Il est donc possible que certaines étapes doivent être améliorées une nouvelle fois.

- *Déploiement des AGVs dans la chaîne de production de l'industrie* : Pour effectuer cette étape, l'ensemble des problèmes en relation avec le système complet doit être pris en compte. Ceci peut mener à reconsidérer certaines hypothèses ou à simplifier l'approche proposée. Les autres entités (comme le superviseur par exemple) doivent aussi être conçue pour prendre en compte l'intervention des humains dans le système. Par exemple, des interfaces utilisateurs peuvent être utilisées pour permettre à l'humain de prendre en main le système ou pour comprendre le comportement des AGVs. De plus, le coût des AGVs doit être évalué par rapport à leur capacité souhaitée.

Les travaux de cette thèse ont été consacrés aux différentes étapes de conception (partie gauche du diagramme de la Fig. 6.23). Nous avons aussi proposé des résultats numériques à l'aide de simulations et le test de faisabilité des trajectoires a été concluant. En suivant ce diagramme, la prochaine étape consiste à valider l'embarquabilité. Pour cela, il est nécessaire d'améliorer les algorithmes de résolution (comme l'optimisation par essaims particulaires) afin de réduire au maximum les temps de calcul pour la génération des trajectoires. On pourrait par exemple utiliser des méta heuristiques parallèles [Alba, 2005] pour réduire le temps de calcul lorsque la planification de trajectoires et l'ordonnancement sont combinés.

En supposant que les tests d'embarquabilité soient concluants, la prochaine étape consisterait à mettre les AGVs en situation de production. Cependant, ceci nécessite une modification de plusieurs hypothèses. D'abord, il faudrait que les AGVs considèrent l'ensemble des opérations du produit sans pour autant rendre plus complexe les algorithmes de navigation proposés. Ensuite, l'aspect d'énergie des AGVs devra aussi être pris en compte car la production en masse de produits à l'aide d'AGVs nécessite des déplacements entre ressources successifs, et donc très coûteux en énergie. Enfin, il faudrait aussi considérer que les files d'attentes aux ressources sont limitées. Ceci a plusieurs impacts dans l'allocation des ressources mais aussi sur la possibilité de changer le cahier des charges de production lors de la navigation du produit.

En considérant juste les tests d'embarquabilité et de mise en situation de production des AGVs, on remarque qu'il pourrait y avoir plusieurs modifications dans l'algorithme de navigation. Ces modifications peuvent mener à un problème d'embarquabilité si on force les AGVs à tout traiter par eux–même. Dans ce travail, nous avons plutôt cherché à concevoir des approches donnant de l'intelligence aux AGVs. Par conséquent, pour pouvoir implémenter les méthodes proposées, il serait

préférable que les modifications apportées à nos méthodes suivent une procédure basée sur la conception pour l'embarquabilité [Kuo et al., 2001]. En effet, en utilisant le paradigme de la balance illustré sur la Fig. 6.22, des poids sont positionnés de chaque côté de la balance. D'un côté, les poids correspondent aux fonctions des AGVs selon leur complexité. De l'autre, les poids correspondent aux capacités de calcul des AGVs où par exemple un poids correspond à un coeur du processeur. Des poids importants donnent de meilleures capacités mais le coût sera aussi plus élevé. Par conséquent, en suivant une conception permettant l'embarquabilité, il serait judicieux de considérer les capacités des AGVs et leur conséquence en termes de coût avant de modifier les algorithmes proposés. Lorsque les capacités sont définies, les nouvelles fonctions des AGVs peuvent être conçues afin de garder un bon compromis. Bien entendu, ceci peut amener une réduction de l'intelligence des AGVs. Cependant, la coopération avec les autres entités du système de production (superviseur, ressources, ...), peut permettre de compenser ce manque d'intelligence des AGVs.

Les différentes perspectives proposées jusqu'à maintenant sont principalement focalisées sur le problème de planification de trajectoire. Cependant, il y a aussi des perspectives pour la commande coopérative. En effet, nous avons vu que le protocole de commande proposé implique plusieurs manoeuvres. Celle–ci peuvent être gênantes lorsque les AGVs naviguent dans le système de production. De plus, nous avons vu que l'amplitude des entrées peut prendre des valeurs élevées lors des transitions pour permettre une convergence rapide. Cependant, il est possible que les AGVs ne soient pas capables d'assurer ces commandes car les entrées sont bornées. De ce fait, il est nécessaire que le temps de stabilisation donné soit faisable pour les robots. Par conséquent, une idée serait de concevoir un algorithme d'optimisation, qui agirait sur les temps de commutation et sur le temps de stabilisation, permettant de minimiser les manoeuvres tout en respectant les entrées bornées de chaque robot.

Ensuite, comme le graphe de communication dépend de la position actuelle des agents, on pourrait approfondir le protocole pour considérer les commutations dans le graphe au lieu de le considérer comme fixe. Enfin, le dernier aspect concerne l'évitement de collisions pour le problème de rendez–vous. En effet, il est physiquement impossible pour des robots d'atteindre la même configuration. Par conséquent, une autre perspective serait de faire en sorte que les agents atteignent la configuration en respectant une certaine distance de sécurité. De plus, il serait préférable que le protocole permette l'évitement de collision lors de la navigation.

Références bibliographiques

[Adibi et al., 2010] Adibi, M., Zandieh, M., et Amiri, M. (2010). Multi-objective scheduling of dynamic job shop using variable neighborhood search. Expert Systems with Applications, 37(1), pp. 282–287.

[Ahuactzin et al., 1993] Ahuactzin, J. M., Talbi, E.-G., Bessiere, P., et Mazer, E. (1993). Using genetic algorithms for robot motion planning. Dans Geometric Reasoning for Perception and Action, pp. 84–93. Springer.

[Aissani, 2010] Aissani, N. (2010). Pilotage adaptatif et réactif pour un système de production à flux continu : application à un système de production pétrochimique. Thèse de Doctorat, Ecole doctorale de Lille ; Université d'Oran, Algérie.

[Alba, 2005] Alba, E. (2005). Parallel metaheuristics : a new class of algorithms, volume 47. John Wiley & Sons.

[Allahverdi et Al-Anzi, 2006] Allahverdi, A. et Al-Anzi, F. S. (2006). A pso and a tabu search heuristics for the assembly scheduling problem of the two-stage distributed database application. Computers & Operations Research, 33(4), pp. 1056–1080.

[Asano et al., 1985] Asano, T., Asano, T., Guibas, L., Hershberger, J., et Imai, H. (1985). Visibility-polygon search and euclidean shortest paths. Dans Foundations of Computer Science, 1985., 26th Annual Symposium on, pp. 155–164. IEEE.

[Banaszak et Krogh, 1990] Banaszak, Z. A. et Krogh, B. H. (1990). Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows. Robotics and Automation, IEEE Transactions on, 6(6), pp. 724–734.

[Barbosa et al., 2011] Barbosa, J., Leitão, P., Trentesaux, D., et Adam, E. (2011). Enhancing adacor with biology insights towards reconfigurable manufacturing systems. Dans IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society, pp. 2746–2751. IEEE.

[Bekrar et al., 2011] Bekrar, A., Chaabane, S., Trentesaux, S., Bornschlegell, A., Pellé, J., et Harmand, S. (2011). Hybrid PSO-tabu search for constrained nonlinear optimization problems. Dans ICSI 2011 : International Conference on Swarm Intelligence.

[Bertrand et Wortmann, 1992] Bertrand, J. W. M. et Wortmann, J. (1992). Information systems for production planning and control : developments in perspective. Production planning & Control, 3(3), pp. 280–289.

[Bhat et Bernstein, 2005] Bhat, S. P. et Bernstein, D. S. (2005). Geometric homogeneity with applications to finite-time stability. Mathematics of Control, Signals and Systems, 17(2), pp. 101–127.

[Bland, 1993] Bland, J. (1993). Nonlinear optimization of constrained functions using tabu search. International Journal of Mathematical Education in Science and Technology, 24(5), pp. 741–747.

[Bocewicz et al., 2014] Bocewicz, G., Nielsen, I., et Banaszak, Z. (2014). Automated guided vehicles fleet match-up scheduling with production flow constraints. Engineering Applications of Artificial Intelligence, 30, pp. 49–62.

[Boggs et Tolle, 1995] Boggs, P. T. et Tolle, J. W. (1995). Sequential quadratic programming. Acta numerica, 4, pp. 1–51.

[Böhnlein et al., 2011] Böhnlein, D., Schweiger, K., et Tuma, A. (2011). Multi-agent-based transport planning in the newspaper industry. International Journal of Production Economics, 131(1), pp. 146–157.

[Bolton et Tyler, 2008] Bolton, R. et Tyler, S. (2008). Pqli engineering controls and automation strategy. Journal of Pharmaceutical Innovation, 3(2), pp. 88–94.

[Bongaerts et al., 1995] Bongaerts, L., Valckenaers, P., Van Brussel, H., et Wyns, J. (1995). Schedule execution for a holonic shop floor control system. Advanced Summer Institute (ASI'95) on Life Cycle Approaches to Production Systems, Lisbon, Portugal.

[Brand et al., 2010] Brand, M., Masuda, M., Wehner, N., et Yu, X.-H. (2010). Ant colony optimization algorithm for robot path planning. Dans Computer Design and Applications (ICCDA), 2010 International Conference on, volume 3, pp. V3–436. IEEE.

[Brockett et al., 1983] Brockett, R. W. et al. (1983). Asymptotic stability and feedback stabilization. Differential geometric control theory, 27(1), pp. 181–191.

[Buzacott et Yao, 1986] Buzacott, J. A. et Yao, D. D. (1986). Flexible manufacturing systems : a review of analytical models. Management science, 32(7), pp. 890–905.

[Camalot, 2000] Camalot, J.-P. (2000). Aide à la décision et à la coopération en gestion du temps et des ressources. Thèse de Doctorat, Institut National des sciences appliquées de Toulouse.

[Canny, 1988] Canny, J. (1988). The complexity of robot motion planning. MIT press.

[Canny et Reif, 1987] Canny, J. et Reif, J. (1987). New lower bound techniques for robot motion planning problems. Dans Foundations of Computer Science, 1987., 28th Annual Symposium on, pp. 49–60. IEEE.

[Cao et Ren, 2010a] Cao, Y. et Ren, W. (2010a). Multi-vehicle coordination for double-integrator dynamics under fixed undirected/directed interaction in a sampled-data setting. International Journal of Robust and Nonlinear Control, 20(9), pp. 987–1000.

[Cao et Ren, 2010b] Cao, Y. et Ren, W. (2010b). Sampled-data discrete-time coordination algorithms for double-integrator dynamics under dynamic directed interaction. International Journal of Control, 83(3), pp. 506–515.

[Cao et Ren, 2012] Cao, Y. et Ren, W. (2012). Distributed coordinated tracking with reduced interaction via a variable structure approach. Automatic Control, IEEE Transactions on, 57(1), pp. 33–48.

[Cao et al., 2010] Cao, Y., Ren, W., et Meng, Z. (2010). Decentralized finite-time sliding mode estimators and their applications in decentralized finite-time formation tracking. Systems & Control Letters, 59(9), pp. 522–529.

[Cao et al., 2013] Cao, Y., Yu, W., Ren, W., et Chen, G. (2013). An overview of recent progress in the study of distributed multi-agent coordination. Industrial Informatics, IEEE Transactions on, 9(1), pp. 427–438.

[Cardin et al., 2013] Cardin, O., Mebarki, N., et Pinot, G. (2013). A study of the robustness of the group scheduling method using an emulation of a complex fms. International Journal of Production Economics, 146(1), pp. 199–207.

[Cardin et al., 2016] Cardin, O., Ounnar, F., Thomas, A., et Trentesaux, D. (2016). Future industrial systems : Best practices of the intelligent manufacturing & services systems (ims 2) french research group. IEEE Transactions on Industrial Informatics.

[Cardin et al., 2015] Cardin, O., Trentesaux, D., Thomas, A., Castagna, P., Berger, T., et El-Haouzi, H. B. (2015). Coupling predictive scheduling and reactive control in manufacturing hybrid control architectures : state of the art and future challenges. Journal of Intelligent Manufacturing, pp. 1–15.

[Carriker et al., 1990] Carriker, W. F., Khosla, P. K., et Krogh, B. H. (1990). The use of simulated annealing to solve the mobile manipulator path planning problem. Dans Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on, pp. 204–209. IEEE.

[Chatterjee et Siarry, 2006] Chatterjee, A. et Siarry, P. (2006). Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. Computers & Operations Research, 33(3), pp. 859–871.

[Chen et Li, 2006] Chen, X. et Li, Y. (2006). Smooth path planning of a mobile robot using stochastic particle swarm optimization. Dans Mechatronics and Automation, Proceedings of the 2006 IEEE International Conference on, pp. 1722–1727. IEEE.

[Chen et al., 1999] Chen, Y., Peng, Y., Finin, T., Labrou, Y., Cost, S., Chu, B., Sun, R., et Willhelm, R. (1999). A negotiation-based multi-agent system for supply chain management. Working Notes of the agents, 99.

[Cho et Lazaro, 2010] Cho, S. et Lazaro, A. (2010). Control theoretic model using pid controller for just-in-time production scheduling. The International Journal of Advanced Manufacturing Technology, 51(5-8), pp. 699–709.

[Choset, 1996] Choset, H. (1996). Sensor based motion planning : The hierarchical generalized Voronoi graph. Thèse de Doctorat, Citeseer.

[Chu et al., 2014] Chu, Y., You, F., et Wassick, J. M. (2014). Hybrid method integrating agent-based modeling and heuristic tree search for scheduling of complex batch processes. Computers & Chemical Engineering, 60, pp. 277–296.

[Clerc, 1999] Clerc, M. (1999). The swarm and the queen : towards a deterministic and adaptive particle swarm optimization. Dans Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, volume 3. IEEE.

[Coffman Jr et al., 1990] Coffman Jr, E., Yannakakis, M., Magazine, M., et Santos, C. (1990). Batch sizing and job sequencing on a single machine. Annals of Operations Research, 26(1), pp. 135–147.

[Conway et al., 2012] Conway, R. W., Maxwell, W. L., et Miller, L. W. (2012). Theory of scheduling. Courier Corporation.

[Cortés, 2006] Cortés, J. (2006). Finite-time convergent gradient flows with applications to network consensus. Automatica, 42(11), pp. 1993–2000.

[Cox et Durfee, 2003] Cox, J. S. et Durfee, E. H. (2003). Discovering and exploiting synergy between hierarchical planning agents. Dans Proceedings of the second international joint conference on Autonomous agents and multiagent systems, pp. 281–288. ACM.

[De Boor et al., 1978] De Boor, C., De Boor, C., De Boor, C., et De Boor, C. (1978). A practical guide to splines, volume 27. Springer-Verlag New York.

[De Gennaro et Jadbabaie, 2006] De Gennaro, M. C. et Jadbabaie, A. (2006). Formation control for a cooperative multi-agent system using decentralized navigation functions. Dans American Control Conference, 2006, pp. 6–pp. IEEE.

[Defoort, 2007] Defoort, M. (2007). Contributions à la planification et à la commande pour les robots mobiles coopératifs. Thèse de Doctorat, Ecole Centrale de Lille.

[Defoort et al., 2011] Defoort, M., Doniec, A., et Bouraqadi, N. (2011). Decentralized robust collision avoidance based on receding horizon planning and potential field for multi-robots systems. Dans Informatics in Control Automation and Robotics, pp. 201–215. Springer.

[Defoort et al., 2008] Defoort, M., Floquet, T., Kokosy, A., et Perruquetti, W. (2008). Sliding-mode formation control for cooperative autonomous mobile robots. Industrial Electronics, IEEE Transactions on, 55(11), pp. 3944–3953.

[Defoort et al., 2005] Defoort, M., Floquet, T., Perruquetti, W., et Kökösy, A. M. (2005). Tracking of a unicycle-type mobile robot using integral sliding mode control. Dans ICINCO, volume 5, pp. 106–111.

[Defoort et al., 2009a] Defoort, M., Kokosy, A., Floquet, T., Perruquetti, W., et Palos, J. (2009a). Motion planning for cooperative unicycle-type mobile robots with limited sensing ranges : A distributed receding horizon approach. Robotics and Autonomous Systems, 57(11), pp. 1094–1106.

[Defoort et al., 2009b] Defoort, M., Palos, J., Kokosy, A., Floquet, T., et Perruquetti, W. (2009b). Performance-based reactive navigation for non-holonomic mobile robots. Robotica, 27(02), pp. 281–290.

[Defoort et al., 2007] Defoort, M., Palos, J., Kokosy, A., Floquet, T., Perruquetti, W., et Boulinguez, D. (2007). Experimental motion planning and control for an autonomous nonholonomic mobile robot. Dans Robotics and Automation, 2007 IEEE International Conference on, pp. 2221–2226. IEEE.

[Defoort et al., 2015] Defoort, M., Polyakov, A., Demesure, G., Djemai, M., et Veluvolu, K. (2015). Leader-follower fixed-time consensus for multi-agent systems with unknown non-linear inherent dynamics. Control Theory & Applications, IET, 9(14), pp. 2165–2170.

[DeGroot, 1974] DeGroot, M. H. (1974). Reaching a consensus. Journal of the American Statistical Association, 69(345), pp. 118–121.

[Demesure et al., 2014] Demesure, G., Defoort, M., Bekrar, A., Trentesaux, D., et Djemaï, M. (2014). Cooperation mechanisms in multi-agent robotic systems and their use in distributed manufacturing control : Issues and literature review. Dans Industrial Electronics Society, IECON 2014-40th Annual Conference of the IEEE, pp. 2538–2543. IEEE.

[Deneubourg et al., 1994] Deneubourg, J. L., Clip, P. L., et Camazine, S. S. (1994). Ants, buses and robots-self-organization of transportation systems. Dans From Perception to Action Conference, 1994., Proceedings, pp. 12–23. IEEE.

[Digani et al., 2014] Digani, V., Caramaschi, F., Sabattini, L., Secchi, C., et Fantuzzi, C. (2014). Obstacle avoidance for industrial agvs. Dans Intelligent Computer Communication and Processing (ICCP), 2014 IEEE International Conference on, pp. 227–232. IEEE.

[Do et Pan, 2007] Do, K. et Pan, J. (2007). Nonlinear formation control of unicycle-type mobile robots. Robotics and Autonomous Systems, 55(3), pp. 191–204.

[Dorigo et al., 1996] Dorigo, M., Maniezzo, V., et Colorni, A. (1996). Ant system : optimization by a colony of cooperating agents. Systems, Man, and Cybernetics, Part B : Cybernetics, IEEE Transactions on, 26(1), pp. 29–41.

[Eberhart et al., 1995] Eberhart, R. C., Kennedy, J., et al. (1995). A new optimizer using particle swarm theory. Dans Proceedings of the sixth international symposium on micro machine and human science, volume 1, pp. 39–43. New York, NY.

[Eberhart et Shi, 2001] Eberhart, R. C. et Shi, Y. (2001). Particle swarm optimization : developments, applications and resources. Dans evolutionary computation, 2001. Proceedings of the 2001 Congress on, volume 1, pp. 81–86. IEEE.

[ElMaraghy, 2005] ElMaraghy, H. A. (2005). Flexible and reconfigurable manufacturing systems paradigms. International journal of flexible manufacturing systems, 17(4), pp. 261–276.

[Erol et al., 2012] Erol, R., Sahin, C., Baykasoglu, A., et Kaplanoglu, V. (2012). A multi-agent based approach to dynamic scheduling of machines and automated guided vehicles in manufacturing systems. Applied Soft Computing, 12(6), pp. 1720–1732.

[Fattahi et Fallahi, 2010] Fattahi, P. et Fallahi, A. (2010). Dynamic scheduling in flexible job shop systems by considering simultaneously efficiency and stability. CIRP Journal of Manufacturing Science and Technology, 2(2), pp. 114–123.

[Fillippov, 1988] Fillippov, A. P. (1988). Differential equations with discontinuous right-hand side. Dortrecht, The Netherlands : Kluwer.

[Ganesharajah et al., 1998] Ganesharajah, T., Hall, N. G., et Sriskandarajah, C. (1998). Design and operational issues in agv-served manufacturing systems. Annals of Operations Research, 76, pp. 109–154.

[Gazi et Passino, 2004] Gazi, V. et Passino, K. M. (2004). Stability analysis of social foraging swarms. Systems, Man, and Cybernetics, Part B : Cybernetics, IEEE Transactions on, 34(1), pp. 539–557.

[Gilmour, 2003] Gilmour, K. (2003). Amazon warehouse, amazon adventure. Internet Magazine.

[Girault et al., 2013] Girault, J., Loiseau, J. J., et Roux, O. H. (2013). Synthèse en ligne de superviseur compositionnel pour flotte de robots mobiles. European Journal of Automation, MSR, 13, pp. 1–3.

[Glover, 1989] Glover, F. (1989). Tabu search-part i. ORSA Journal on computing, 1(3), pp. 190–206.

[Goldberg et Holland, 1988] Goldberg, D. E. et Holland, J. H. (1988). Genetic algorithms and machine learning. Machine learning, 3(2), pp. 95–99.

[Graves, 1981] Graves, S. C. (1981). A review of production scheduling. Operations research, 29(4), pp. 646–675.

[Guo et Parker, 2002] Guo, Y. et Parker, L. E. (2002). A distributed and optimal motion planning approach for multiple mobile robots. Dans Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on, volume 3, pp. 2612–2619. IEEE.

[Hayes-Roth, 1985] Hayes-Roth, B. (1985). A blackboard architecture for control. Artificial intelligence, 26(3), pp. 251–321.

[Heragu et al., 2002] Heragu, S. S., Graves, R. J., Kim, B.-I., et Onge, A. S. (2002). Intelligent agent based framework for manufacturing systems control. Systems, Man and Cybernetics, Part A : Systems and Humans, IEEE Transactions on, 32(5), pp. 560–573.

[Herrera, 2011] Herrera, C. (2011). Cadre générique de planification logistique dans un contexte de décisions centralisées et distribuées. Thèse de Doctorat, Université Henri Poincaré-Nancy I.

[Herrero-Perez et Martinez-Barbera, 2008] Herrero-Perez, D. et Martinez-Barbera, H. (2008). Petri nets based coordination of flexible autonomous guided vehicles in flexible manufacturing systems. Dans Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on, pp. 508–515. IEEE.

[Herrero-Perez et Matinez-Barbera, 2008] Herrero-Perez, D. et Matinez-Barbera, H. (2008). Decentralized coordination of autonomous agvs in flexible manufacturing systems. Dans Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, pp. 3674–3679. IEEE.

[Ho et Tay, 2008] Ho, N. B. et Tay, J. C. (2008). Solving multiple-objective flexible job shop problems by evolution and local search. Systems, Man, and Cybernetics, Part C : Applications and Reviews, IEEE Transactions on, 38(5), pp. 674–685.

[Hu et Eberhart, 2002] Hu, X. et Eberhart, R. (2002). Solving constrained nonlinear optimization problems with particle swarm optimization. Dans Proceedings of the sixth world multiconference on systemics, cybernetics and informatics, volume 5, pp. 203–206. Citeseer.

[Huang et Tsai, 2011] Huang, H.-C. et Tsai, C.-C. (2011). Global path planning for autonomous robot navigation using hybrid metaheuristic ga-pso algorithm. Dans SICE Annual Conference (SICE), 2011 Proceedings of, pp. 1338–1343. IEEE.

[Ishibuchi et Murata, 1998] Ishibuchi, H. et Murata, T. (1998). A multi-objective genetic local search algorithm and its application to flowshop scheduling. Systems, Man, and Cybernetics, Part C : Applications and Reviews, IEEE Transactions on, 28(3), pp. 392–403.

[Janabi-Sharifi et Vinke, 1993] Janabi-Sharifi, F. et Vinke, D. (1993). Integration of the artificial potential field approach with simulated annealing for robot path planning. Dans Intelligent Control, 1993., Proceedings of the 1993 IEEE International Symposium on, pp. 536–541. IEEE.

[Jiang et Wang, 2009] Jiang, F. et Wang, L. (2009). Finite-time information consensus for multi-agent systems with fixed and switching topologies. Physica D : Nonlinear Phenomena, 238(16), pp. 1550–1560.

[Jimenez et al., 2016] Jimenez, J.-F., Bekrar, A., Trentesaux, D., et Leitao, P. (2016). A switching mechanism framework for optimal coupling of predictive scheduling and reactive control in manufacturing hybrid control architectures. International Journal of Production Research.

[Jones et al., 1999] Jones, A., Rabelo, L. C., et Sharawi, A. T. (1999). Survey of job shop scheduling techniques. Wiley Encyclopedia of Electrical and Electronics Engineering.

[Kadera et Tichy, 2009] Kadera, P. et Tichy, P. (2009). Plan, commit, execute protocol in multi-agent systems. Dans Holonic and multi-agent systems for manufacturing, pp. 155–164. Springer.

[Kavraki et al., 1996] Kavraki, L. E., Švestka, P., Latombe, J.-C., et Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. Robotics and Automation, IEEE Transactions on, 12(4), pp. 566–580.

[Khoo et al., 2009] Khoo, S., Xie, L., et Man, Z. (2009). Robust finite-time consensus tracking algorithm for multirobot systems. IEEE/ASME transactions on mechatronics, 14(2), pp. 219–228.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Vecchi, M. P., et al. (1983). Optimization by simmulated annealing. science, 220(4598), pp. 671–680.

[Kraus, 2001] Kraus, S. (2001). Automated negotiation and decision making in multiagent environments. Dans Multi-agent systems and applications, pp. 150–172. Springer.

[Künhe et al., 2005] Künhe, F., Gomes, J., et Fetter, W. (2005). Mobile robot trajectory tracking using model predictive control. Dans II IEEE latin-american robotics symposium.

[Kuo et al., 2001] Kuo, T.-C., Huang, S. H., et Zhang, H.-C. (2001). Design for manufacture and design for 'x' : concepts, applications, and perspectives. Computers & Industrial Engineering, 41(3), pp. 241–260.

[Latombe, 2012] Latombe, J.-C. (2012). Robot motion planning, volume 124. Springer Science & Business Media.

[LaValle et Kuffner, 2001] LaValle, S. M. et Kuffner, J. J. (2001). Randomized kinodynamic planning. The International Journal of Robotics Research, 20(5), pp. 378–400.

[Leitao, 2004] Leitao, P. J. P. (2004). An agile and adaptive holonic architecture for manufacturing control. Thèse de Doctorat, University of Porto.

[Lenstra et Kan, 1981] Lenstra, J. K. et Kan, A. (1981). Complexity of vehicle routing and scheduling problems. Networks, 11(2), pp. 221–227.

[Li et al., 2011] Li, S., Du, H., et Lin, X. (2011). Finite-time consensus algorithm for multi-agent systems with double-integrator dynamics. Automatica, 47(8), pp. 1706–1712.

[Lin et al., 2009] Lin, J.-H., Huang, L.-R., et al. (2009). Chaotic bee swarm optimization algorithm for path planning of mobile robots. Dans Proceedings of the 10th WSEAS international conference on evolutionary computing, pp. 84–89. World Scientific and Engineering Academy and Society (WSEAS).

[Lin et al., 2005] Lin, Z., Francis, B., et Maggiore, M. (2005). Necessary and sufficient graphical conditions for formation control of unicycles. Automatic Control, IEEE Transactions on, 50(1), pp. 121–127.

[Linn et Zhang, 1999] Linn, R. et Zhang, W. (1999). Hybrid flow shop scheduling : a survey. Computers & industrial engineering, 37(1), pp. 57–61.

[Liu et al., 2005] Liu, S. Q., Ong, H., et Ng, K. (2005). Metaheuristics for minimizing the makespan of the dynamic shop scheduling problem. Advances in Engineering Software, 36(3), pp. 199–205.

[Lozano-Perez, 1983] Lozano-Perez, T. (1983). Spatial planning : A configuration space approach. Computers, IEEE Transactions on, 100(2), pp. 108–120.

[Lu et al., 2012] Lu, X., Austin, F., et Chen, S. (2012). Formation control for second-order multi-agent systems with time-varying delays under directed topology. Communications in Nonlinear Science and Numerical Simulation, 17(3), pp. 1382–1391.

[Lv et Feng, 2006] Lv, N. et Feng, Z. (2006). Numerical potential field and ant colony optimization based path planning in dynamic environment. Dans Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on, volume 2, pp. 8966–8970. IEEE.

[Lynch, 1996] Lynch, N. A. (1996). Distributed algorithms. Morgan Kaufmann.

[Malinowski et Yu, 2011] Malinowski, A. et Yu, H. (2011). Comparison of embedded system design for industrial applications. Industrial Informatics, IEEE Transactions on, 7(2), pp. 244–254.

[Martínez-Barberá et Herrero-Pérez, 2010] Martínez-Barberá, H. et Herrero-Pérez, D. (2010). Autonomous navigation of an automated guided vehicle in industrial environments. Robotics and Computer-Integrated Manufacturing, 26(4), pp. 296–311.

[Masehian et Amin-Naseri, 2008] Masehian, E. et Amin-Naseri, M. R. (2008). Sensor-based robot motion planning-a tabu search approach. Robotics & Automation Magazine, IEEE, 15(2), pp. 48–57.

[Masehian et Sedighizadeh, 2007] Masehian, E. et Sedighizadeh, D. (2007). Classic and heuristic approaches in robot motion planning-a chronological review. World Academy of Science, Engineering and Technology, 23, pp. 101–106.

[Mataric, 1992] Mataric, M. J. (1992). Minimizing complexity in controlling a mobile robot population. Dans Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on, pp. 830–835. IEEE.

[Maturana et al., 1999] Maturana, F., Shen, W., et Norrie, D. H. (1999). Metamorph : an adaptive agent-based architecture for intelligent manufacturing. International Journal of Production Research, 37(10), pp. 2159–2173.

[Min et al., 2005] Min, H.-Q., Zhu, J.-H., et Zheng, X.-J. (2005). Obstacle avoidance with multi-objective optimization by pso in dynamic environment. Dans Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on, volume 5, pp. 2950–2956. IEEE.

[Mintzberg, 1989] Mintzberg, H. (1989). The structuring of organizations. Dans Readings in Strategic Management, pp. 322–352. Springer.

[Mohamad et al., 2005] Mohamad, M. M., Dunnigan, M. W., et Taylor, N. K. (2005). Ant colony robot motion planning. Dans Computer as a Tool, 2005. EUROCON 2005. The International Conference on, volume 1, pp. 213–216. IEEE.

[Nickabadi et al., 2011] Nickabadi, A., Ebadzadeh, M. M., et Safabakhsh, R. (2011). A novel particle swarm optimization algorithm with adaptive inertia weight. Applied Soft Computing, 11(4), pp. 3658–3670.

[Noguchi et Terao, 1997] Noguchi, N. et Terao, H. (1997). Path planning of an agricultural mobile robot by neural network and genetic algorithm. Computers and electronics in agriculture, 18(2), pp. 187–204.

[Novas et al., 2013] Novas, J. M., Van Belle, J., Saint Germain, B., et Valckenaers, P. (2013). A collaborative framework between a scheduling system and a holonic manufacturing execution system. Dans Service orientation in holonic and multi agent manufacturing and robotics, pp. 3–17. Springer.

[Olfati-Saber, 2006] Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems : Algorithms and theory. Automatic Control, IEEE Transactions on, 51(3), pp. 401–420.

[Olfati-Saber et Murray, 2002] Olfati-Saber, R. et Murray, R. M. (2002). Distributed cooperative control of multiple vehicle formations using structural potential functions. 15(1), pp. 242–248.

[Olfati-Saber et Murray, 2004] Olfati-Saber, R. et Murray, R. M. (2004). Consensus problems in networks of agents with switching topology and time-delays. Automatic Control, IEEE Transactions on, 49(9), pp. 1520–1533.

[Ottaway et Burns, 2000] Ottaway, T. et Burns, J. (2000). An adaptive production control system utilizing agent technology. International Journal of Production Research, 38(4), pp. 721–737.

[Ou-Yang et Lin, 1998] Ou-Yang, C. et Lin, J. (1998). The development of a hybrid hierarchical/heterarchical shop floor control system applying bidding method in job dispatching. Robotics and Computer-Integrated Manufacturing, 14(3), pp. 199–217.

[Pach, 2013] Pach, C. (2013). ORCA : Architecture hybride pour le contrôle de la myopie dans le cadre du pilotage des Systèmes Flexibles de Production. Thèse de Doctorat, Université de Valenciennes et du Hainaut-Cambresis.

[Pach et al., 2014a] Pach, C., Berger, T., Bonte, T., et Trentesaux, D. (2014a). Orca-fms : a dynamic architecture for the optimized and reactive control of flexible manufacturing scheduling. Computers in Industry, 65(4), pp. 706–720.

[Pach et al., 2014b] Pach, C., Berger, T., Sallez, Y., Bonte, T., Adam, E., et Trentesaux, D. (2014b). Reactive and energy-aware scheduling of flexible manufacturing systems using potential fields. Computers in Industry, 65(3), pp. 434–448.

[Parker et al., 1989] Parker, J. K., Khoogar, A. R., et Goldberg, D. E. (1989). Inverse kinematics of redundant robots using genetic algorithms. Dans Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on, pp. 271–276. IEEE.

[Parsegov et al., 2012] Parsegov, S., Polyakov, A., et Shcherbakov, P. (2012). Nonlinear fixed-time control protocol for uniform allocation of agents on a segment. Dans 2012 IEEE 51st IEEE Conference on Decision and Control (CDC), pp. 7732–7737. IEEE.

[Parsegov et al., 2013] Parsegov, S., Polyakov, A., et Shcherbakov, P. (2013). Fixed-time consensus algorithm for multi-agent systems with integrator dynamics. Dans 4th IFAC Workshop on Distributed Estimation and Control in Networked Systems, pp. 110–115.

[Parsopoulos et Vrahatis, 2005] Parsopoulos, K. E. et Vrahatis, M. N. (2005). Unified particle swarm optimization for solving constrained engineering optimization problems. Dans Advances in natural computation, pp. 582–591. Springer.

[Parunak, 1985] Parunak, V. D. (1985). Fractal actors for distributed manufacturing control. The engineering of knowledge-based systems, pp. 653–660.

[Petkova et van Wezel, 2006] Petkova, B. et van Wezel, W. (2006). Disentangling manufacturing flexibility. Dans Preprints of the Fourteenth International Working Seminar on Production Economics, pp. 287–296.

[Pinedo, 2012] Pinedo, M. L. (2012). Scheduling : theory, algorithms, and systems. Springer Science & Business Media.

[Polyakov, 2012a] Polyakov, A. (2012a). Nonlinear feedback design for fixed-time stabilization of linear control systems. Automatic Control, IEEE Transactions on, 57(8), pp. 2106–2110.

[Polyakov, 2012b] Polyakov, A. (2012b). Nonlinear feedback design for fixed-time stabilization of linear control systems. IEEE Transactions on Automatic Control, 57(8), pp. 2106–2110.

[Qin et al., 2011] Qin, J., Gao, H., et Zheng, W. X. (2011). Second-order consensus for multi-agent systems with switching topology and communication delay. Systems & Control Letters, 60(6), pp. 390–397.

[Raileanu et al., 2012] Raileanu, S., Parlea, M., Borangiu, T., et Stocklosa, O. (2012). A jade environment for product driven automation of holonic manufacturing. Dans Service Orientation in Holonic and Multi-Agent Manufacturing Control, pp. 265–277. Springer.

[Rajendran et Ziegler, 2004] Rajendran, C. et Ziegler, H. (2004). Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. European Journal of Operational Research, 155(2), pp. 426–438.

[Ren, 2008] Ren, W. (2008). Consensus tracking under directed interaction topologies : Algorithms and experiments. Dans American Control Conference, 2008, pp. 742–747. IEEE.

[Ren et al., 2005] Ren, W., Beard, R. W., et al. (2005). Consensus seeking in multiagent systems under dynamically changing interaction topologies. IEEE Transactions on automatic control, 50(5), pp. 655–661.

[Reveliotis et Roszkowska, 2010] Reveliotis, S. A. et Roszkowska, E. (2010). On the complexity of maximally permissive deadlock avoidance in multi-vehicle traffic systems. IEEE Transactions on Automatic Control, 55(7), pp. 1646–1651.

[Rey et al., 2013] Rey, G. Z., Pach, C., Aissani, N., Bekrar, A., Berger, T., et Trentesaux, D. (2013). The control of myopic behavior in semi-heterarchical production systems : A holonic framework. Engineering Applications of Artificial Intelligence, 26(2), pp. 800–817.

[Rolón et Martínez, 2012] Rolón, M. et Martínez, E. (2012). Agent-based modeling and simulation of an autonomic manufacturing execution system. Computers in Industry, 63(1), pp. 53–78.

[Rolstadas, 2010] Rolstadas, A. (2010). Ims 2020 roadmap for sustainable manufacturing research. Proceedings from the IMS2020 Summer School on Sustainable Manufacturing, pp. 26–28.

[Sallez et al., 2010] Sallez, Y., Berger, T., Deneux, D., et Trentesaux, D. (2010). The lifecycle of active and intelligent products : The augmentation concept. International Journal of Computer Integrated Manufacturing, 23(10), pp. 905–924.

[Sallez et al., 2009] Sallez, Y., Berger, T., et Trentesaux, D. (2009). A stigmergic approach for dynamic routing of active products in fms. Computers in industry, 60(3), pp. 204–216.

[Sankar et al., 2005] Sankar, S. S., Ponnambalam, S., Rathinavel, V., et Visveshvaren, M. (2005). Scheduling in parallel machine shop : an ant colony optimization approach. Dans Industrial Technology, 2005. ICIT 2005. IEEE International Conference on, pp. 276–280. IEEE.

[Saska et al., 2006] Saska, M., Macaš, M., Přeučil, L., et Lhotska, L. (2006). Robot path planning using particle swarm optimization of ferguson splines. Dans Emerging Technologies and Factory Automation, 2006. ETFA'06. IEEE Conference on, pp. 833–839. IEEE.

[Scattolini, 2009] Scattolini, R. (2009). Architectures for distributed and hierarchical model predictive control–a review. Journal of Process Control, 19(5), pp. 723–731.

[Scholten, 2007] Scholten, B. (2007). The road to integration : A guide to applying the ISA-95 standard in manufacturing. Isa.

[Senehi et Kramer, 1998] Senehi, M. et Kramer, T. R. (1998). A framework for control architectures. International Journal of Computer Integrated Manufacturing, 11(4), pp. 347–363.

[Sha et Hsu, 2008] Sha, D. et Hsu, C.-Y. (2008). A new particle swarm optimization for the open shop scheduling problem. Computers & Operations Research, 35(10), pp. 3243–3261.

[Shi et al., 2016] Shi, S., Yu, X., et Khoo, S. (2016). Robust finite-time tracking control of nonholonomic mobile robots without velocity measurements. International Journal of Control, 89(2), pp. 411–423.

[Shi et al., 2015] Shi, S., Yu, X., et Liu, G. (2015). Finite-time consensus algorithm for multiple nonholonomic disturbed systems with its application. Mathematical Problems in Engineering, 2015.

[Shi-Cai et al., 2007] Shi-Cai, L., Da-Long, T., et Guang-Jun, L. (2007). Robust leader-follower formation control of mobile robots based on a second order kinematics model. Acta Automatica Sinica, 33(9), pp. 947–955.

[Song et al., 2010] Song, Q., Cao, J., et Yu, W. (2010). Second-order leader-following consensus of nonlinear multi-agent systems via pinning control. Systems & Control Letters, 59(9), pp. 553–562.

[Sousa et Ramos, 1999] Sousa, P. et Ramos, C. (1999). A distributed architecture and negotiation protocol for scheduling in manufacturing systems. Computers in industry, 38(2), pp. 103–113.

[Srivastava et al., 2008] Srivastava, S. C., Choudhary, A. K., Kumar, S., et Tiwari, M. (2008). Development of an intelligent agent-based agv controller for a flexible manufacturing system. The International Journal of Advanced Manufacturing Technology, 36(7-8), pp. 780–797.

[Sucan et al., 2012] Sucan, I. A., Moll, M., et Kavraki, L. E. (2012). The open motion planning library. Robotics & Automation Magazine, IEEE, 19(4), pp. 72–82.

[Taghezout et Zaraté, 2008] Taghezout, N. et Zaraté, P. (2008). Negotiation process for multi-agent dss for manufacturing system. Dans Collaborative Decision Making : Perspectives and Chalenges, pp. 49–60.

[Tangour, 2007] Tangour, F. (2007). Ordonnancement dynamique dans les industries agroalimentaires. Thèse de Doctorat, Ecole Centrale de Lille.

[Taousser et al., 2016] Taousser, F., Defoort, M., et Djemai, M. (2016). Consensus for linear multi-agent system with intermittent information transmissions using the time-scale theory. International Journal of Control, 89(1), pp. 210–220.

[Tarokh, 2008] Tarokh, M. (2008). Hybrid intelligent path planning for articulated rovers in rough terrain. Fuzzy Sets and Systems, 159(21), pp. 2927–2937.

[Tawegoum et al., 1994] Tawegoum, R., Castelain, E., et Gentina, J. (1994). Hierarchical and dynamic production control in flexible manufacturing systems. Robotics and computer-integrated manufacturing, 11(4), pp. 327–334.

[Trelea, 2003] Trelea, I. C. (2003). The particle swarm optimization algorithm : convergence analysis and parameter selection. Information processing letters, 85(6), pp. 317–325.

[Trentesaux, 2002] Trentesaux, D. (2002). Pilotage hétérarchique des systèmes de production. Habilitation à diriger des recherches, Université de Valenciennes et du Hainaut-Cambresis.

[Trentesaux, 2009] Trentesaux, D. (2009). Distributed control of production systems. Engineering Applications of Artificial Intelligence, 22(7), pp. 971–978.

[Trentesaux et al., 2013] Trentesaux, D., Pach, C., Bekrar, A., Sallez, Y., Berger, T., Bonte, T., Leitão, P., et Barbosa, J. (2013). Benchmarking flexible job-shop scheduling and control systems. Control Engineering Practice, 21(9), pp. 1204–1225.

[Trentesaux et al., 1998] Trentesaux, D., Tahon, C., et Ladet, P. (1998). Hybrid production control approach for jit scheduling. Artificial Intelligence in Engineering, 12(1), pp. 49–67.

[Tsitsiklis et al., 1984] Tsitsiklis, J. N., Bertsekas, D. P., et Athans, M. (1984). Distributed asynchronous deterministic and stochastic gradient optimization algorithms. Dans 1984 American Control Conference, pp. 484–489.

[Ueda et al., 2001] Ueda, K., Markus, A., Monostori, L., Kals, H., et Arai, T. (2001). Emergent synthesis methodologies for manufacturing. CIRP Annals-Manufacturing Technology, 50(2), pp. 535–551.

[Valckenaers et al., 2007] Valckenaers, P., Van Brussel, H., Verstraete, P., Saint Germain, B., et al. (2007). Schedule execution in autonomic manufacturing execution systems. Journal of manufacturing systems, 26(2), pp. 75–84.

[Wang et al., 2012] Wang, J., Qiu, Z., Zhang, G., et Yang, W. (2012). Finite-time consensus problem for multiple non-holonomic mobile agents. Dans Intelligent Control and Automation (WCICA), 2012 10th World Congress on, pp. 1739–1744. IEEE.

[Wang et al., 2006] Wang, L., Liu, Y., Deng, H., et Xu, Y. (2006). Obstacle-avoidance path planning for soccer robots using particle swarm optimization. Dans Robotics and Biomimetics, 2006. ROBIO'06. IEEE International Conference on, pp. 1233–1238. IEEE.

[Wang et Hong, 2008] Wang, X. et Hong, Y. (2008). Finite-time consensus for multi-agent networks with second-order agent dynamics. Dans IFAC World Congress, volume 56, pp. 15185–15190.

[Wang et al., 2009] Wang, Z., Li, S., et Fei, S. (2009). Finite-time tracking control of a nonholonomic mobile robot. Asian Journal of Control, 11(3), pp. 344–357.

[Watanabe et al., 2001] Watanabe, M., Furukawa, M., et Kakazu, Y. (2001). Intelligent agv driving toward an autonomous decentralized manufacturing system. Robotics and computer-integrated manufacturing, 17(1), pp. 57–64.

[Wurman et al., 2008] Wurman, P. R., D'Andrea, R., et Mountz, M. (2008). Coordinating hundreds of cooperative, autonomous vehicles in warehouses. AI magazine, 29(1), pp. 9.

[Wyns, 1998] Wyns, J. (1998). Reference architecture for Holonic Manufacturing Systems-the key to support evolution and reconfiguration. Thèse de Doctorat, Katholieke Universiteit Leuven.

[Xiao et al., 2009] Xiao, F., Wang, L., Chen, J., et Gao, Y. (2009). Finite-time formation control for multi-agent systems. Automatica, 45(11), pp. 2605–2611.

[Yang et Kim, 1999] Yang, J.-M. et Kim, J.-H. (1999). Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots. Robotics and Automation, IEEE Transactions on, 15(3), pp. 578–587.

[Yang et al., 2007] Yang, T., Ma, J., Hou, Z.-G., Peng, G., et Tan, M. (2007). A multi-agent architecture based cooperation and intelligent decision making method for multirobot systems. Dans Neural Information Processing, pp. 376–385. Springer.

[Yang et al., 2012] Yang, X., Wang, J., et Tan, Y. (2012). Robustness analysis of leader–follower consensus for multi-agent systems characterized by double integrators. Systems & Control Letters, 61(11), pp. 1103–1115.

[Yong et al., 2012] Yong, C., Huiyang, L., et Guangming, X. (2012). Finite-time containment control for multi-agent systems with single-integrator dynamics. Dans Control Conference (CCC), 2012 31st Chinese, pp. 6433–6438. IEEE.

[Yong-Zheng et Jiong, 2008] Yong-Zheng, S. et Jiong, R. (2008). Leader–follower consensus problems of multi-agent systems with noise perturbation and time delays. Chinese Physics Letters, 25(9), pp. 3493.

[Yu et al., 2011] Yu, F., Kaihara, T., et Fujii, N. (2011). A multi-agent based negotiation for supply chain network using game theory. Dans Advances in Production Management Systems. Value Networks : Innovation, Technologies, and Management, pp. 299–308. Springer.

[Zambrano et al., 2011] Zambrano, G., Pach, C., Aissani, N., Berger, T., et Trentesaux, D. (2011). An approach for temporal myopia reduction in heterarchical control architectures. Dans Industrial Electronics (ISIE), 2011 IEEE International Symposium on, pp. 1767–1772. IEEE.

[Zambrano Rey, 2014] Zambrano Rey, G. (2014). Réduction du comportement myope dans le contrôle des FMS : une approche semi-hétérarchique basée sur la simulation-optimisation. Thèse de Doctorat, Université de Valenciennes et du Hainaut-Cambresis.

[Zbib et al., 2012] Zbib, N., Pach, C., Sallez, Y., et Trentesaux, D. (2012). Heterarchical production control in manufacturing systems using the potential fields concept. Journal of Intelligent Manufacturing, 23(5), pp. 1649–1670.

[Zhang et Yang, 2013] Zhang, Y. et Yang, Y. (2013). Finite-time consensus of second-order leader-following multi-agent systems without velocity measurements. Physics Letters A, 377(3), pp. 243–249.

[Zhengxiong et Xinsheng, 2010] Zhengxiong, G. et Xinsheng, G. (2010). A particle swarm optimization for the motion planning of wheeled mobile robot. Dans Intelligent Control and Automation (WCICA), 2010 8th World Congress on, pp. 2410–2414. IEEE.

[Zhu et al., 2006] Zhu, Q., Yan, Y., et Xing, Z. (2006). Robot path planning based on artificial potential field approach with simulated annealing. Dans Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on, volume 2, pp. 622–627. IEEE.

[Zoghlami, 2014] Zoghlami, N. (2014). Stabilité et stabilisation en temps fini des systemes dynamiques interconnectés et probleme de consensus en temps fini. Thèse de Doctorat, Université d'Évry Val d'Essonne ; École nationale d'ingénieurs de Tunis (Tunisie).

[Zuo et al., 2014] Zuo, Z., Yang, W., Tie, L., et Meng, D. (2014). Fixed-time consensus for multi-agent systems under directed and switching interaction topology. Dans American Control Conference (ACC), 2014, pp. 5133–5138. IEEE.

Annexes

Chapitre A

Détails sur les architectures de pilotage hybrides

Dans la section 1.1.1 du chapitre 1, les architectures de pilotage hybrides sont introduites. Ces architectures sont divisées en quatre sous-classes selon les deux critères discutés : le dynamisme de pilotage pouvant être statique ou dynamique et l'homogénéité du pilotage qui est soit homogène soit hétérogène. Ces sous–classes sont les suivantes :

- **II–SHo, Statique et Homogène** : Un optimiseur de haut niveau gère l'ordonnancement de la production en agissant sur les entités de bas niveau. Ces entités sont actives et peuvent prendre des décisions et communiquer afin d'envoyer des requêtes à l'optimiseur pour recalculer un nouvel ordonnancement le cas échéant.

- **II–SHe, Statique et Hétérogène** : De la même manière que précédemment, l'ordonnancement est fait par un optimiseur global. Cependant, les entités de bas niveau sont capable de le moduler jusqu'à un certain point. Elles peuvent aussi proposer des modifications de l'ordonnancement (en cas de perturbation par exemple) qui devront être validées par l'optimiseur global.

- **II–DHo, Dynamique et Homogène** : Les entités subissent une hiérarchie forte et sont contrôlées par un optimiseur global tant qu'il n'y a pas de perturbation. En cas de perturbation, un mécanisme de basculement permet de passer de hiérarchie à hétérarchie où les entités prennent les décisions de pilotage. Le problème de cette sous-classe est le basculement d'hétérarchie vers hiérarchie. De plus, les entités non concernées par la perturbations doivent aussi prendre des décisions, ce qui diminue les performances globale du système.

- **II–DHe, Dynamique et Hétérogène** : Dans cette sous–classe similaire à la précédente, uniquement le basculement change. En effet, uniquement les entités concernées par la perturbations basculent en hétérarchie. Ainsi, les autres entités restent contrôlées par l'optimiseur global et les performances sont donc moins réduites en cas de perturbation.

Le tableau A.1, issu de [Pach, 2013] présente un ensemble d'exemples de ces sous-classes d'architectures hybrides de classe II utilisées dans la littérature. La seconde colonne représente le mécanisme d'optimisation globale permettant le contrôle de la myopie. La troisième colonne présente le mécanisme myope d'optimisation locale et la dernière colonne indique les références associées.

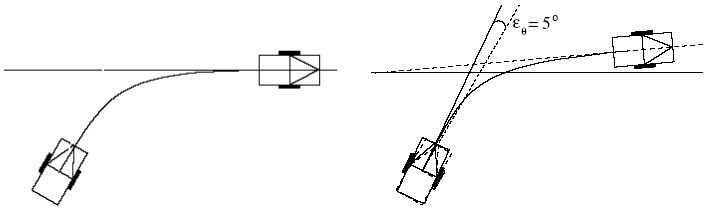
Tableau A.1 – Les sous–classes de l'architecture hybride et références

Sous classe	**Mécanisme d'optimisation globale contrôlant la myopie**	**Mécanisme myope d'optimisation locale**	**Référence**
II–SHo	Ordonnanceur de niveau haut	Demande de re-calcul des contrôleurs de sous-niveau	[Tawegoum et al., 1994]
	Agent superviseur	Demande de re-calcul des ressources	[Ottaway et Burns, 2000]
	Contrôleur de niveau usine	Enchère des contrôleurs de cellule	[Ou-Yang et Lin, 1998]
	Algorithme de synergie	Plan des agents de niveau bas	[Cox et Durfee, 2003]
	Agent coordinateur	Optimisation à base de colonie de fourmis	[Böhnlein et al., 2011]
	Ordonnancement basé sur l'exploration d'arbres	Négociation entre agents	[Chu et al., 2014]
II–SHe	Ordonnanceur global	Négociation de niveau bas	[Parunak, 1985]
	Agent médiateur	Négociation des ressources	[Maturana et al., 1999]
	Agent maitre	Mécanisme d'apprentissage des agents de niveau bas	[Yang et al., 2007]
	Ordonnanceur et coordinateur de niveau haut	Modification du plan par les entités de niveau bas	[Heragu et al., 2002]
	Ordonnancement de l'agent ordre	Modification du plan par les agents ressources	[Rolón et Martínez, 2012]
	Ordonnancement des ressources goulets	Allocation hétérarchique des autres ressources	[Trentesaux et al., 1998]
	Ordonnancement d'un groupe d'opérations	Flexibilité séquentielle des opérations dans un groupe	[Cardin et al., 2013]
	Séquencement de lots de production	Division éventuelle des lots par les agents produits	[Herrera, 2011]
II–DHo	Ordonnanceur de niveau haut	Approche par champs de potentiel	[Pach et al., 2014a]
	Holon staff	système multi–agents délégation	[Novas et al., 2013]
	Holon staff	Contrat net de niveau bas	[Raileanu et al., 2012]
II–DHe	Holon superviseur	Stigmergie de niveau bas	[Barbosa et al., 2011]
	Ordonnancement basé sur exploration d'arbres	Approche par champs de potentiel	[Zambrano et al., 2011]
	Ordonnanceur de niveau haut	Agents fourmis d'exploration et d'intention	[Valckenaers et al., 2007]

Chapitre B

Le suivi de trajectoire

Après avoir généré la trajectoire du robot mobile, il est nécessaire de commander celui–ci, via les actionneurs, afin de suivre au mieux cette trajectoire en prenant en compte sa dynamique. L'approche la plus basique est la commande en boucle ouverte où l'on applique directement les commandes planifiées. Cependant, cela pose problème en pratique car la dynamique du robot est souvent mal modélisée et des perturbations/incertitudes sont présentes. Ainsi, un simple glissement des roues (ou chenilles) du robot peut le faire dévier de sa trajectoire planifiée comme le montre la Fig. B.1 issue de [Defoort, 2007].



(a) Trajectoire planifiée (b) Trajectoire suivie en boucle ouverte

FIGURE B.1 – Problème du suivi de trajectoire en boucle ouverte

Ces inconvénients peuvent être évités par une commande en boucle fermée, en corrigeant la configuration du robot par rapport à sa trajectoire malgré la présence de perturbations et d'incertitudes. L'objectif général de la commande robuste vise donc à considérer les incertitudes paramétriques et les perturbations du système afin de garantir son bon fonctionnement selon un ensemble de critères pour réaliser une tâche à un certain niveau de précision. Beaucoup de travaux ont été proposés sur ce suivi de trajectoire dans le cadre mono–robot à l'aide de commandes predictive [Künhe et al., 2005], à mode glissant [Yang et Kim, 1999] avec action intégrale [Defoort et al., 2005] et aussi à temps fini [Wang et al., 2009].

Chapitre C

Les principales méta heuristiques

Dans cette annexe, les principales méta heuristiques, utilisées pour résoudre des problèmes de planification de trajectoire, sont présentées brièvement avec leur concept. Les méta heuristiques présentées seront, dans cet ordre, le recuit simulé, l'algorithme génétique, les colonies de fourmis, la recherche par tabou et l'optimisation par essaim particulaire. Tout d'abord, nous allons de façon brève, introduire des généralité sur ces méta heuristiques.

Méta heuristiques : généralité

Une méta heuristique est un algorithme d'optimisation qui vise à résoudre des problèmes complexes pour lesquels il n'existe pas de méthode classique efficace. Les méta heuristiques recherchent la meilleure solution d'un problème en manipulant une ou plusieurs solutions de tel sorte que l'on passe, après un certain nombre d'itérations, d'une solution mauvaise à une solution proche de l'optimale. Le principal problème est le choix de la méta heuristique en elle-même car aucune règle n'a été établie et on ne peut savoir à priori quelle méta heuristique choisir pour un problème donné. Afin de choisir, il est souvent recommandé de comparer plusieurs méta heuristiques pour un même problème.

D'une manière générale, elles fonctionnent sur beaucoup de problèmes différents car elles ne nécessitent pas de connaissances particulières sur le problème qu'elles traitent. De plus, on peut rencontrer ces méta heuristiques pour des problèmes à variables discrètes, continues voire mixtes. Les quatres notions auxquelles elles s'articulent sont le voisinage, l'exploration, l'exploitation et la mémoire. Les algorithmes sont dits évolutionnaires lorsqu'ils traitent un ensembles de solutions, appelé population. Pour finir, notons que ces méta heuristiques peuvent être combinées avec d'autres méthodes, on parlera dans ce cas d'hybridation.

Le recuit simulé

Le recuit simulé (*Simulated Annealing*), introduit par [Kirkpatrick et al., 1983], a été un des premiers outils méta heuristiques qui est basé sur le refroidissement des matériaux en métallurgie. A chaque étape, une nouvelle solution x' est définie comme position candidate, choisie de manière aléatoire dans le voisinage de la solution x. Cette nouvelle solution est acceptée sans condition si elle a un potentiel énergétique inférieure (i.e. un coût inférieur). Sinon, l'algorithme passe à l'étape suivante. Ceci est répété jusqu'à temps qu'une petite valeur proche de zéro soit atteinte ou lorsque l'on sort d'un minimum local.

Cette méthode a été utilisée sur un problème de navigation dans [Carriker et al., 1990]. Cependant, cette méthode a plutôt été hybridée avec des champs de potentiel pour éviter les minimums locaux [Zhu et al., 2006, Janabi-Sharifi et Vinke, 1993].

L'algorithme génétique

L'algorithme génétique (*Genetic Algorithm*) est évolutionnaire et a été introduit par [Goldberg et Holland, 1988] en se basant sur des phénomènes biologiques. Le but est, à partir de solutions initiales générées aléatoirement, appelées chromosomes, de faire des changements par mutation ou enjambement dans l'ensemble des solutions puis de faire une sélection. Cette sélection représente la ou les meilleures solutions. A chaque itération, des changements sont appliqués puis une sélection est faite, jusqu'à atteindre une solution optimale. Notons que l'algorithme génétique ne traite que des variable discrètes. Pour résoudre des problèmes continus, une discrétisation devra être faite.

Cet algorithme a été appliqué sur le problème de planification de trajectoire, en premier dans [Parker et al., 1989], puis s'est développé [Ahuactzin et al., 1993] jusqu'à être hybridé avec des réseaux de neurones [Noguchi et Terao, 1997] ou avec de la logique floue [Tarokh, 2008].

La colonie de fourmis

La colonie de fourmis (*Ant Colony*) est un algorithme évolutionnaire, plutôt centré sur l'exploration, développé par [Dorigo et al., 1996]. Cette méta heuristique est basée sur le comportement des fourmis en recherche de nourriture. L'objectif est qu'un ensemble de fourmis creusent (cherchent) en partant de leur nid jusqu'à la nourriture. Chaque fourmi dépose des phéromones sur le chemin (phénomène de mémoire) afin de mémoriser leur solution. Au fur et à mesure, certains chemins moins optimaux s'effacent alors que des chemins meilleurs ont de plus en plus de phéromones jusqu'à ce que le chemin optimal soit trouvé.

Les premières applications de cette méta heuristique sur le problème de navigation sont dans [Deneubourg et al., 1994]. Puis d'autres travaux ont été proposés en changeant l'initialisation des

phéromones [Brand et al., 2010] ou en hybridant avec d'autres algorithmes comme par exemple les champs de potentiel [Lv et Feng, 2006] ou autre [Mohamad et al., 2005].

La recherche tabou

La recherche tabou (*Tabu Search*), introduite par [Glover, 1989], est l'une des approches méta heuristiques ayant des mécanismes d'évitement de minimums locaux. En effet, l'idée consiste à rechercher localement (dans le voisinage) une solution qui minimise la fonction objectif et à interdire le retour vers des solutions déjà explorées. Les solutions interdites sont conservées dans une mémoire dont il est préférable de réduire car on ne garde généralement que les solutions minimisant le critère.

Dans le problème de navigation, la recherche par tabou a été utilisée en ligne dans [Masehian et Amin-Naseri, 2008]. La recherche par tabou peut être hybridée avec d'autres méthodes comme l'optimisation par essaims particulaires [Allahverdi et Al-Anzi, 2006, Bekrar et al., 2011] mais avec peu (voire sans) applications aux problèmes de navigation de robots.

L'optimisation par essaims particulaires

L'optimisation par essaims particulaires (*Particle Swarm Optimization*) est présentée plus amplement dans la section 4.5 du chapitre 4. Elle a été introduite par [Eberhart et al., 1995] en se basant sur les déplacement en groupe d'oiseaux ou de poissons. De plus, c'est aussi un algorithme à population où chaque solution est appelée particule. L'idée consiste que chaque particule se déplace de manière aléatoire vers une position locale optimale en regardant aussi les positions des autres particules (surtout la position de la meilleure). Après un certain nombre d'itérations, les particules se rapprochent plus ou moins les unes des autres jusqu'à atteindre une solution optimale ou proche.

Cette méta heuristique traite principalement des variables continues et est donc très utilisée pour la planification de trajectoire. Elle a été appliqué à des problèmes de robots à roues [Zhengxiong et Xinsheng, 2010] ou d'évitement d'obstacles [Min et al., 2005, Wang et al., 2006], de manière stochastique [Chen et Li, 2006] ou hybridée avec l'algorithme génétique [Huang et Tsai, 2011].

Chapitre D

Généralités sur l'optimisation sous contraintes et les pénalités

Les deux problèmes d'optimisation définis dans le chapitre 4 (section 4.5) sont des problèmes non linéaires sous contraintes. Ces problèmes sont souvent établis, d'une manière générale, comme suit :

$$\text{minimiser} \qquad h(x) \tag{D.0.1}$$

$$\text{sujet à} \quad g_p(x) \leq 0,\ p = 1, 2, .., N_c \tag{D.0.2}$$

où x est l'ensemble des variables de décision, $h(x)$ est la fonction objectif et $g_p(x)$ sont les N_c contraintes. Afin de prendre en compte les contraintes, les fonctions de pénalité permettent de les considérer en les ajoutant au critère. Ainsi, lorsqu'une contrainte n'est pas respectée, la valeur à minimiser augmente, se traduisant par :

$$\min_x \quad f(x) + \sum_{p=1}^{N_c} \alpha_p \cdot g_p(x) \tag{D.0.3}$$

avec

$$\alpha_p = \begin{cases} 0 & \text{if} \quad g_p(x) \leq 0 \\ N_p & \text{if} \quad g_p(x) > 0 \end{cases} \tag{D.0.4}$$

où N_p est un nombre grand permettant à une contrainte de pénaliser la solution si celle–ci n'est pas respectée. Notons qu'il est possible de favoriser certaines contraintes. Par exemple, pour deux contraintes $g_1(x)$ et $g_2(x)$, en prenant $N_1 > N_2$, la contrainte $g_1(x)$ sera plus pénalisée que l'autre. On peut donc donner plus ou moins d'importance à une contrainte, ce qui facilite aussi l'ajout d'inégalités valides.