Supervision d'un système de transport par AGVs bidirectionnels dans un contexte de défauts de capteurs

Samia MAZA[1] et Pascale MARANGE[1]
[1] Université de Lorraine, Centre de Recherche en Automatique de Nancy.
samia.maza@univ-lorraine.fr, pascale.marange@univ-lorraine.fr

Résumé

Les chariots autoguidés ou AGVs sont considérés parmi les moyens de transport les plus flexibles dans les systèmes automatisés de production (SAP). Un des problèmes importants que pose le déploiement d'une flotte d'AGVs bidirectionnels est celui d'une commande sûre garantissant un routage sans inter-blocages des véhicules. En partant d'une méthode de routage sans conflits basée en partie sur la théorie de contrôle par supervision (TCS) de Ramadge et Wonham, cet article propose de s'intéresser aux défaillances pouvant surgir dans certains capteurs. Dans la théorie de supervision classique, il est supposé que les événements impliqués sont observables. Nous proposons donc de considérer le cas où des capteurs peuvent défaillir, rendant certains événements non observables et le superviseur inefficient. L'objectif de cet article est d'une part de faire une étude fiabiliste de l'impact des défaillances capteurs sur la qualité de la supervision. D'autre part de proposer une architecture de supervision tolérante aux fautes qui intègre la fonction de diagnostic. Cette dernière permettra, lorsque le système d'AGVs vérifie la propriété de diagnosticabilité, d'éviter certains états de blocages provoqués par la perte d'information sur la position de certains véhicules. Le système sera modélisé par des automates temporisés stochastiques. Nous utiliserons le *Model Checking* et l'outil UPPAAL afin d'évaluer la fiabilité de l'architecture proposée.

1 Introduction

Du fait de leur flexibilité et efficacité, les véhicules autoguidés ou AGVs (en anglais *Automated Guided Vehicles*) sont largement utilisés pour la manutention de produits dans des installations telles

que les centres de distribution, les ateliers de production, les entrepôts et les terminaux (Maza et Castagna, 2007). Un AGV est un véhicule autonome qui se déplace en suivant des circuits de guidage matérialisés de diverses manières. Un système d'AGVs est dit bi-directionnel lorsque les véhicules sont autorisés à se déplacer dans les deux sens sur certaines voies de circulation (Figure 1). Si les systèmes bidirectionnels procurent plus d'avantages (Egbelu et Tanchoco, 86), leur commande est plus complexe à cause des nombreuses situations de conflits qui doivent être évitées (Figure 1). En effet, il est nécessaire de piloter la flotte et de coordonner les véhicules afin que ceux-ci puissent réaliser leurs missions de transport en évitant de s'inter-bloquer. Ce problème est connu sous le nom de routage sans conflits (Maza et Castagna, 07). Il consiste à calculer pour chaque AGV le chemin qu'il doit suivre pour atteindre une destination, souvent en minimisant un critère de performance donné (ex. le temps de transport, la distance,…) et sans entrer en conflit avec les autres véhicules. Si les collisions peuvent être évitées grâce aux capteurs et contrôleurs embarqués, les blocages eux doivent être anticipés et évités au niveau du pilotage pour éviter des états de famine et l'interruption du trafic. En effet, l'interruption du trafic nécessite une intervention externe pour résoudre les conflits ce qui a pour effet de dégrader les performances du système de transport (le temps de mission se trouvant rallongé). Dans cet article, nous nous intéressons aux blocages liées aux mouvements des véhicules et non ceux liées au problème d'affectation des missions de transport (i.e., le problème de *Dispatching*). Il existe plusieurs approches dans la littérature pour résoudre ce problème. Elles ont un point en commun : elles sont basées sur une division du circuit de guidage en plusieurs zones et contrôlent ainsi l'accès des véhicules à ces zones tout le long de leurs trajets vers leurs destinations. Certaines approches sont de nature prédictive. Elles permettent de planifier les AGVs et leurs trajets tout en optimisant un critère de performance. Par exemple dans (Kim et Tanchoco, 1993), les auteurs proposent une méthode d'ordonnancement temporel des AGVs sur les zones du circuit, pour les amener vers leurs destinations en un temps minimal et sans conflits. Une méthode similaire est proposée dans (Rajotia, Shankar, et Batra, 1998). Gamache, Grimard, et Cohen (2005) ont adapté cette approche pour gérer une flotte de véhicules dans une mine. Les auteurs de (Krishnamurthy et al, 1993) considèrent le problème d'optimisation conjointe du dispatching et du routage d'AGVs pour minimiser le temps de transport global (le *makespan*) par la programmation en nombres entiers.
La planification simultanée des grues de quai, de chantier et d'AGVs dans un terminal à conteneurs automatisé est étudiée dans (Yang et al.2018) en utilisant les algorithmes génétiques.
Ces approches peuvent être qualifiées de méthodes de routage en boucle ouverte. Elles planifient les véhicules dans le temps en se basant sur les hypothèses d'absence de perturbations et de fonctionnement selon les prédictions. Elles souffrent de manque de robustesse aux aléas qui peuvent survenir dans le système et empêcher le respect de l'ordonnancement temporel sans conflit.

D'autres approches que l'on peut qualifier en boucle fermée sont de nature plus réactive. Les décisions de routage sont prises de façon dynamique en fonction de l'état du système. Elles ont pour principe d'allouer une seule zone par AGV, en vérifiant l'absence de blocages avant l'allocation. Par exemple dans (Reveliotis, 2000), une adaptation de l'algorithme du banquier est utilisée pour résoudre ce problème d'allocation de ressources lié aux AGVs mais de façon non optimale. Des algorithmes basés sur la théorie des graphes et la caractérisation des interblocages sont proposés dans (Fanti, 2002). La simulation pro-active est utilisée dans (Felko, 2011) pour prédire les blocages et les éviter en temps réel et une approche de routage par algorithmes de consensus est proposée dans (Fanti et al. 2018).
L'inconvénient des approches réactives est que les performances ne sont pas considérées et optimisées a priori puisque les décisions sont prises en temps réel en considérant un horizon très court.

Pour allier les avantages des deux types d'approches, à savoir l'optimisation et la réactivité aux aléas, des méthodes qu'on peut qualifier d'hybrides, ont fait l'objet de plusieurs publications. Par exemple dans (Maza et Castagna, 2005a, 2005b), une approche de routage robuste à deux niveaux est proposée. Le premier niveau ordonnance les AGVs sur les zones du circuit, et le second évite les

conflits en présence de perturbations. Une méthode de routage basée sur la synthèse du superviseur le plus permissif avec recherche d'un chemin qui optimise un critère de performances est proposé dans (Arnaud et al. 2009). Celle-ci n'est adaptée que pour les systèmes de petite taille. Dans le but réduire la taille du superviseur, les auteurs de (Girault et al. 2016) proposent d'utiliser la méthode de γ-dépliage pour calculer le superviseur le plus permissif respectant une contrainte sur le coût, dont le seuil limite est γ.

Dans (Maza, 19), une architecture de routage basée sur deux niveaux est proposée. Un niveau de planification des trajets d'AGVs couplé à un second basé sur la théorie de supervision pour éviter les inter-blocages en temps réel. Cette architecture permet d'une part de minimiser un critère de performance, et d'autre part de réduire la sensibilité de la méthode de supervision à la taille du système (i.e., le problème d'explosion combinatoire) tout en permettant une meilleure robustesse aux perturbations. Elle est la base du travail proposé dans cet article et sera brièvement rappelée dans la section 3.

La supervision d'AGVs utilisée dans (Maza, 19), comme dans la théorie de Ramadge et Wonham de façon générale, est construite sur la base d'un ensemble d'événements observables. Pour le système d'AGVs, ces événements permettant de localiser les AGVs sur les zones du circuit de guidage, sont donc supposés observables. En pratique, cela correspond à l'existence de capteurs et de balises délimitant les zones du circuit de guidage et permettant de détecter le passage d'un AGV d'une zone à une autre. Cette information est indispensable pour que le superviseur prédise et empêche les états interdits et les états peu sûrs (Maza, 19). En réalité, ces événements peuvent devenir non-observables puisque les capteurs peuvent être sujets à des défauts ou défaillances. La perte d'observabilité de certains événements peut conduire à l'occurrence de séquences d'événements interdites. Il est donc important que le superviseur puisse utiliser les informations disponibles dans le système, afin d'éviter d'atteindre des états de blocage.

A notre connaissance, il y a peu de travaux qui considèrent la supervision dans un contexte d'événements non observables. On peut citer le travail de Paoli et Lafortune (Paoli et Lafortune, 2005) qui introduisent la propriété de diagnosticabilité sûre pour la réalisation d'une supervision tolérante aux pannes d'un système à événements discrets. Cette propriété empêcherait qu'un défaut local se développe en une défaillance globale avec un impact sur la sécurité. Dans le contexte des AGVs, on peut citer l'étude de (Li et al. 2016) qui tient compte des pannes sur les AGVs dans le problème de routage sans conflits, et propose des procédures de circulation d'urgence en complément aux règles de circulation normale.

L'objectif de cet article est de proposer une architecture de supervision tolérante aux fautes qui intègre la fonction de diagnostic des événements qui peuvent devenir non observables. Cette dernière permettra dans certaines situations d'éviter la violation de contrainte de sécurité (i.e., deux AGVs se retrouvant face-à-face).

L'article est organisé comme suit. L'approche de supervision du système d'AGVs bidirectionnels est présentée formellement dans la section 3 après quelques rappels sur les automates d'états et la théorie de supervision. La section 4 s'intéresse au problème du diagnostic des systèmes à événements discrets par automates et qui sera appliqué au système d'AGVs. La nouvelle architecture de supervision tolérante aux fautes y sera également présentée. Le *Model Checking* appliqué à la propriété d'absence de blocage sera utilisé en section 5 pour analyser l'efficacité de cette architecture qui sera modélisée par automates temporisés stochastiques. Une analyse de la fiabilité et de l'efficacité de cette architecture est effectuée en simulation avec l'outil UPPAAL et est présentée dans la section 6. Une conclusion et des perspectives feront l'objet de la section 7.

2 Contexte de l'étude et motivation

Comme mentionné plus haut, pour piloter efficacement et sans conflits les AGVs, le circuit de guidage est divisé en plusieurs zones souvent considérées comme non partageables. Les méthodes de routage établissent alors pour les AGVs les zones qu'ils doivent emprunter pour réaliser leur mission de transport ainsi que les règles d'accès à ces zones. Ces dernières garantissent un fonctionnement sécuritaire et performant. Cela fait des systèmes de transport d'AGVs des systèmes à événements discrets (SED). La plupart de ces méthodes de commande, supposent la connaissance certaine de l'information sur la position de chaque AGV, i.e., la zone où se trouve un chariot. Le routage est donc construit autour des événements représentant l'accès d'un AGV aux zones du circuit et du séquencement de ceux-ci pour éviter les blocages. Un blocage est une situation indésirable où des véhicules ne peuvent plus avancer et terminer leur mission, dégradant ainsi le temps de transport et la productivité du système (Figure 1).

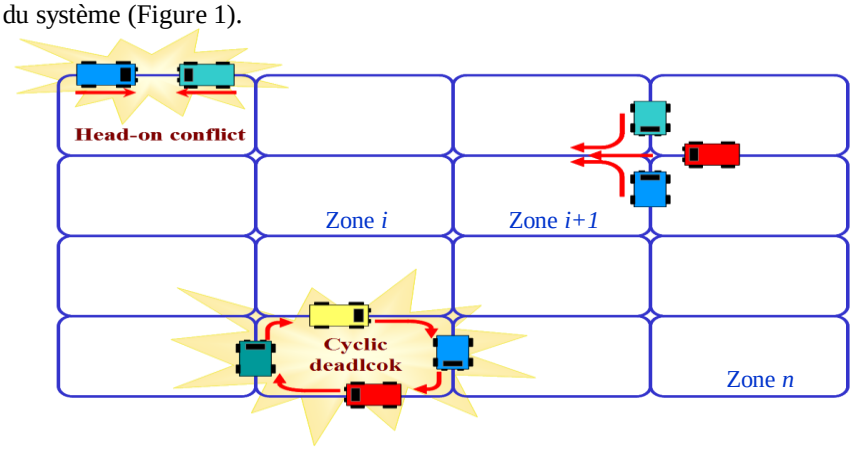


Figure 1 : Exemple d'un circuit bi-directionnel avec les différentes situations de blocage.

Dans la pratique, des capteurs peuvent être placés pour délimiter ces zones et détecter le passage des AGVs (Maza et Castagna, 2007). Par exemple des lecteurs *RFID* pour détecter et identifier les AGVs, équipés de patchs d'identification, cherchant à accéder à une zone. Les informations issues de ces capteurs représentent les événements qui seront utilisés par le superviseur. L'observabilité de ces événements est ainsi liée à la disponibilité, au sens fiabiliste, des capteurs qui les produisent.

Dans cet article, nous nous intéressons aux défaillances liées à l'instrumentation de l'infrastructure de guidage. Des défauts capteurs qui rendent la détection d'un AGV qui rentre dans une zone impossible et donc l'événement associé non observable. Si le superviseur central ne reçoit pas cet événement, il peut autoriser des séquences d'événements qui mèneraient le système dans un état défendu. Afin de rendre notre superviseur plus sûr et tolérant aux défauts de capteurs de positions, nous proposons d'étudier la propriété de diagnosticabilité du système d'AGVs et de réaliser, quand cela est possible le diagnostic des événements devenus non observables. Cette information sera alors utilisée par le superviseur pour empêcher l'occurrence de certaines séquences d'événements interdites contenant les événements non observables. Nous considérerons des défauts capteurs qui sont des défaillances passagères dans le sens ou un capteur en défaut peut ne pas détecter le passage d'un AGV (par exemple problème de salissure, de positionnement, d'environnement fortement perturbé, etc.). Une défaillance complète et définitive pourrait être prise en compte par une action de maintenance et/ou une reconfiguration du circuit et de ses zones. Nous analyserons ensuite l'impact de tels défauts sur la fiabilité de l'architecture de supervision.

3 Supervision d'un système d'AGVs bidirectionnels

Dans cette section, nous allons brièvement rappeler le principe de l'architecture de routage en deux niveaux proposée dans (Maza, 2019). Nous nous focaliserons sur le second niveau basé sur la théorie de supervision par Ramadge et Wonham (Ramadge et Wonham, 1987) dont voici quelques rappels.

3.1 Rappels sur la théorie de supervision par automates

Considérons un SED G modélisé par un automate à états fini $G=(X, \sum, \delta, x_0, X_m)$ où X est l'espace d'états, $\sum$ est l'ensemble d'événements, δ est la fonction de transition d'état, x_0 est l'état initial et X_m est l'ensemble des états marqués. L'occurrence d'un événement fait évoluer le système d'un état à un autre. Un mot $\sigma_1\sigma_2\ldots\sigma_n$ construit sur l'alphabet $\sum$ (i.e. $\sigma_i \in \sum$, avec $i=1, n$) est appelé une trace.

Le comportement du système est décrit par un langage préfixe-clos $\mathcal{L}(G)=\{s \in \sum^* \text{ tels que } \delta(x_0, s) \text{ est défini}\}$ avec $\sum^*$ l'ensemble de tous les mots sur l'alphabet $\sum$. Ce langage représente l'ensemble des traces qui peuvent être générées par l'automate. Certains événements peuvent être contrôlables, i.e., dont peut empêcher l'occurrence, et d'autres non contrôlables. Par conséquent, l'ensemble $\sum$ est partitionné en deux sous-ensembles $\sum=\sum_c \cup \sum_u$ où $\sum_c$ est l'ensemble des événements contrôlables et $\sum_u$ est l'ensemble des événements non contrôlables.
Dans la théorie du contrôle par supervision des SEDs, voir par exemple (Lafortune et Cassadras, 1999), on se donne un modèle G du système en boucle ouverte (non contrôlé) et un ensemble de spécifications sur le comportement désiré en boucle fermée. Généralement, G exprime l'interconnexion, via l'opération de composition parallèle (notée ||), d'un ensemble de composants en interaction dont les modèles sont désignés par (G_1, G_2, …G_n) et $G=\|_{i=1 \text{ à } n} G_i$. Le comportement du système, décrit par $\mathcal{L}(G)$, doit être restreint par le contrôle pour respecter les spécifications et en compatibilité avec l'ensemble des événements non contrôlables $\sum_u$. Un superviseur, représenté par un automate S, est donc synthétisé et connecté au modèle G du système, tel que le comportement du système supervisé $G^{Sup}=\|_{i=1 \text{ à } n} G_i \| S$ satisfasse les contraintes précédentes. La compatibilité avec les événements non-contrôlables de $\sum_u$ signifie que l'occurrence de ceux-ci ne fera évoluer le système que vers des états vérifiant les spécifications. C'est la contrôlablité.

3.2 Méthodologie de routage robuste sans conflits d'AGVs basée sur la supervision

Comme expliqué précédemment, nous allons brièvement décrire la méthode de routage proposée dans (Maza, 19), qui est basée sur deux niveaux : un de planification et un d'évitement de blocages réactif. Cette architecture hybride (Figure 2) a été proposée dans deux buts. D'une part afin d'allier l'optimisation d'une approche prédictive à la robustesse d'une approche réactive à base de supervision. Par robustesse, on entend la capacité du routage à éviter les blocages en présence des perturbations faisant écarter le comportement du système d'AGVs de ce qui a été planifié. D'autre part, de réduire la taille de l'espace d'états du système supervisé. En effet, il est connu que la faiblesse de la méthode de contrôle par supervision de Ramadge et Wonham est l'explosion combinatoire du nombre d'états en fonction de la taille du système (ici la taille de la flotte et le nombre de zones).

Voici quelques hypothèses sur lesquelles est basée cette méthode.

- Le circuit est divisé en zones, dont la dimension est supérieure à la taille maximale d'un AGV. Une zone est une ressource partagée entre plusieurs AGVs et a une capacité limitée.

- Chaque zone a un identifiant Z_i et est délimitée par deux nœuds, repérés par des capteurs de positions. Le passage d'un AGV k par un nœud n produit un événement $e_{n,k}$ (i, k et n quelconque). Ces événements forment un ensemble E.
- Une mission de transport consiste à aller d'un nœud de départ (source) à un nœud destination. Deux AGVs ne doivent pas avoir comme destination les nœuds où se trouve actuellement l'autre véhicule.

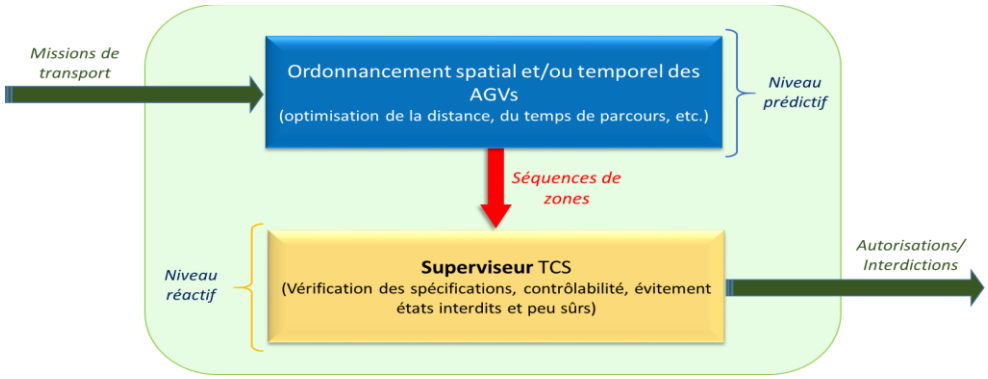


Figure 2 : Architecture de commande hybride.

A. Le niveau prédictif :

Dans ce niveau, une quelconque méthode de planification des AGVs qui optimise un critère donné peut être employée pour calculer les chemins de chaque AGV vers sa destination. Cela peut être une méthode qui anticipe et prédit les conflits et planifie ainsi le passage des AGVs par certaines zones à des dates précises qui garantissent l'évitement de blocages et la minimisation du temps de transport (voir par exemple Kim et Tanchoco, 1993). Une méthode qui calcule les chemins entre deux nœuds en minimisant la distance parcourue (ex. algorithme de *Dijskstra*) sans prédiction des blocages peut également être utilisée dans le niveau 1. Ces blocages seront gérés et évités par la supervision (niveau 2). Néanmoins comme indiqué dans la dernière hypothèse, il est indispensable de ne pas avoir deux AGVs qui auraient deux chemins identiques (la source de l'un est la destination de l'autre et inversement). Ainsi, la méthode de planification qui sera utilisée dans le premier niveau permettra d'optimiser un critère de performance et de délivrer au second niveau la séquence de zones que le véhicule doit emprunter pour accomplir sa mission de transport.

B. Le niveau réactif :

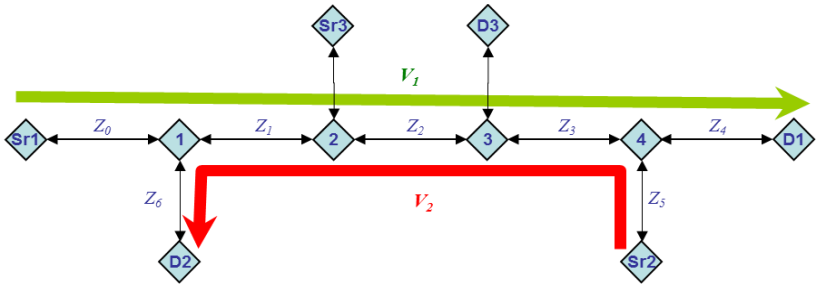


Figure 3 : Exemple d'un chemin commun à deux AGVs dans deux sens opposés.

Chaque mission de transport planifiée d'un AGV V_k est repérée par un mot dont la trace est la séquence de zones qu'il doit occuper en se déplaçant entre ses nœuds source et destination. Cette séquence, qu'on appellera mot-mission et qui sera notée S_k, est construite sur l'alphabet des identifiants de zones nommé $\sum_z$. Considérons les trajets planifiés des deux AGVs de la Figure 3 par exemple. Les mots-mission des véhicules V_1 et V_2 sont respectivement $S_1=Z_0Z_1Z_2Z_3Z_4$ et $S_2=Z_5Z_3Z_2Z_1Z_6$.

Voici brièvement les étapes de la méthode de supervision du niveau 2:

- ***Etape1*** : Les chemins communs partagés par deux véhicules V_k et V_j sont calculés à partir des sous-langages communs à leurs mots-mission S_k et S_j. La séquence commune, notée $S_{k,j}$, permet d'identifier les zones communes, les événements associés et les nœuds extrêmes du chemin commun (CC).
- ***Etape2*** : Définir une spécification pour chaque zone identifiée comme partagée entre deux AGVs ou plus. Celle-ci peut être permissive ou non mais n'autorise son parcours que dans un sens ou l'autre. La spécification est définie comme ayant un état initial et deux sous-ensemble d'états, appelés classes d'états : une classe d'états pour chaque sens de parcours et la taille d'une classe est égale à la capacité d'une zone (voir l'exemple de la Figure 4). Elle permet, dans le cas d'une spécification permissive, d'autoriser plusieurs AGVs à accéder à une zone s'ils sont planifiés dans le même sens et si la capacité de celle-ci le permet (Maza, 2019).
- ***Etape3*** : A partir de l'étape1, identifier pour chaque AGV V_k l'ensemble E_k des événements associés à son mot-mission. Tous les événements sont supposés observables. Pour les AGVs partageant des trajets commun, ces ensembles d'événements sont partitionnés en deux sous-ensembles E_k^c et E_k^u pour respectivement les événements contrôlables et non contrôlables. La non-contrôlabilité sera affectée aux événements associés aux nœuds à l'intérieur d'un chemin partagé dans deux sens opposés par deux AGVs, à l'exception des nœuds extrêmes.
- ***Etape4*** : Synthétiser le superviseur, en compatibilité avec les événements non-contrôlables, qui respecte ces spécifications.

Quelques remarques :

- Une spécification permet d'éviter les conflits de type « collision frontale » (en anglais *Head-on Conflict*).
- La qualification des événements internes à un chemin commun comme non-contrôlables permet d'éviter des états peu sûrs (c.à.d états où le blocage n'est pas imminent mais inévitable).

Exemple :

D'après la Figure 3, les nœuds internes du chemin commun (CC) où V_1 et V_2 s'opposent sont *2* et *3*. Tous les événements associés aux deux AGVs seront considérés contrôlables, à l'exception de ceux associés aux deux nœuds précédents. Ainsi, $E_1^u=\{e_{21}, e_{31}\}$ et $E_2^u=\{e_{22}, e_{32}\}$, où le premier indice est le numéro du nœud et le second celui du véhicule.

Supposons maintenant l'existence d'un troisième AGV V_3 se déplaçant entre Sr_3 et D_3 en empruntant Z_2 dans le même sens que V_1. Si la capacité de Z_2 le permet, un exemple de spécification de zone permissive associée à Z_2 et aux AGVs la traversant est donné à la Figure 4. Ici la notation $Z_n^{i,j}$ signifie que la zone Z_n est parcourue dans le sens i par j AGVs simultanément (où ***i*=1** dans le sens nœud *2* vers *3*, et ***i*=2** en sens inverse). L'état initial de la spécification, noté Z_n^0, signifie que la zone est inoccupée (dans cet exemple n=2). Sur la Figure 4 par exemple, la zone Z_2 peut-être parcourue par deux AGVs à la fois s'ils vont dans le même sens (sens 1). Ainsi, si V_1 s'engage sur Z_2 (état $Z_2^{1,1}$ est actif) le véhicule V_3 qui a le même sens de déplacement que V_1 est autorisé à s'engager à son tour dans cette zone avant que V_1 ne l'ait quitté (état $Z_2^{1,2}$ est actif). Lorsque V_1 quitte Z_2 on revient à l'état $Z_2^{1,1}$ puis à l'état Z_2^0 après départ de V_3. De même si V_3 est le premier à s'engager sur Z_2. En revanche dans l'autre sens (sens 2), seul V_2 peut s'engager et occuper la zone.

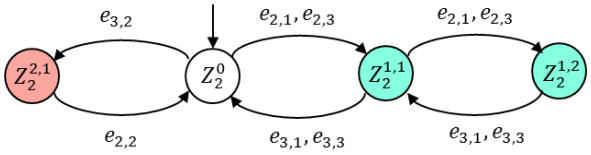


Figure 4 : Exemple d'une spécification permissive pour une zone partagée entre 3 AGVs

4 Le diagnostic dans un système d'AGVs

Comme indiqué précédemment, le contrôle par supervision est basé sur une hypothèse forte qu'est l'observabilité des événements. On peut alors se poser la question de ce qui arriverait si un événement devenait non observable suite à un défaut de son capteur (ici de position). L'objectif de cette section est d'utiliser une technique de diagnostic des SEDs basée sur les automates qui, en cas de diagnosticabilité, permet de communiquer au superviseur l'information sur l'occurrence de l'événement devenu non observable. Ainsi le superviseur pourra, dans certains cas, éviter des séquences interdites qui mènent vers un blocage. Le diagnostiqueur intégrera par la suite l'architecture de commande du système d'AGVs pour la rendre plus tolérante aux fautes.

4.1 Rappels sur le diagnostic des SEDs

Soit un système G partiellement observable, i.e., dont l'alphabet des événements $\sum$ peut être réparti en deux sous-ensembles disjoints $\sum=\sum^{o}\cup\sum^{no}$ avec $\sum^{o}$ est l'ensemble des événements observables et $\sum^{no}$ celui des événements non observables. Soit σ l'événement non observable à diagnostiquer ($\sum^{no}=\{\sigma\}$) et P la fonction de projection sur $\sum^{o}$ qui retire d'un mot les événements non observables. Formellement, $\forall s\in\sum^{*}$, avec $\sigma\notin s$ (i.e., le mot s ne contient pas σ), $P(s\sigma)=s$. La fonction projection inverse est notée P^{-1}. Soit $\psi(\sigma)$ l'ensemble des mots dans $L(G)$ se terminant par l'événement non observable σ. Rappelons la définition formelle de la diagnosticabilité selon (Sampath, 95).

Définition. *Un langage L préfixe-clos et vivant est dit diagnosticable par rapport à une fonction de projection P et σ si et seulement si* $(\exists n \in \mathbb{N})(\forall s \in \Psi(\sigma))(\forall t \in L \,/\, s)[\| t \| \geq n \Rightarrow D]$ *où la condition de diagnosticabilité* $\boldsymbol{D}$: $\omega \in P^{-1}P(st) \cap L \Rightarrow (\sigma \in \omega)$

Autrement dit, L est diagnosticable par rapport à σ ssi pour toute séquence d'événements s se terminant par l'événement σ et toute continuation t de s suffisamment longue ($\|t\|\geq n$), alors toute autre séquence d'événements ω ayant la même projection observable $(P(st) = P(\omega))$ contient nécessairement l'événement non observable σ.

4.2 Diagnostic d'événements non observables dans un système d'AGV

Considérons l'observabilité des deux types d'événements: ceux liés aux nœuds internes du CC et ceux associés aux nœuds extrêmes. Rappelons que la supervision (à travers les spécifications) ne concerne que les zones partagées entre plusieurs AGVs se déplaçant en sens inverse essentiellement. En effet, les zones non partagées ne sont pas problématiques. Il en est de même pour les événements qui leurs sont associés.

Cas 1 : si certains capteurs de positions internes sont sujets aux défauts rendant les événements associés non-observables, cela n'affecte pas l'efficacité de la supervision. En effet, le superviseur a

été calculé en considérant ces événements comme non-contrôlables. Il n'est donc pas nécessaire de diagnostiquer leur occurrence.

Cas 2 : Pour éviter les états indésirables et peu sûrs, le superviseur doit détecter et empêcher l'occurrence des événements contrôlables associés aux nœuds extrêmes du CC. En pratique, il interdira à un AGV d'accéder à son CC avec autre AGV en sens inverse si celui-ci s'y est déjà engagé car un blocage futur est inévitable (état peu sûr). L'observabilité de ces événements est donc plus cruciale pour la supervision et son efficacité. La perte de cette information sur l'engagement d'un AGV sur sa voie commune avec un autre AGV devient en effet un problème si ce dernier AGV s'engage à son tour.

Nous allons donc pour ce dernier cas essayer d'utiliser les informations disponibles sur les autres capteurs de positions pour diagnostiquer l'événement non-observable lié à un nœud extrême. Cette information sur le diagnostic, lorsqu'elle est disponible, sera alors intégrée dans le superviseur pour améliorer sa tolérance aux fautes et le rendre plus sûr.

Considérons deux AGVs V_i et V_k, modélisés par deux automates G_i et G_k, se déplaçant sur un circuit et partageant un CC en sens inverse. Les missions prévues pour ces AGVs sont décrites par deux séquences d'événements M_i et M_k qu'ils doivent produire pour atteindre leur destination. Soit $L_{i,k}$ le langage généré par l'automate de composition $G_i \| G_k$ qui décrit les séquences possibles sur l'AGVS. Supposons que les nœuds appartenant au CC sont numérotés de *1*,*2*,…*n* (*n* quelconque) avec *1* et *n* les nœuds externes de ce CC. Admettons que sur ce CC, V_i va dans le sens *1* à *n*, V_k fait le trajet inverse et que le capteur en *1* est sujet à des défauts rendant parfois les événements e_{1i} et e_{1k} non observables.

Pour le système considéré, et suivant la définition précédente, $L_{i,k}$ est diagnosticable par rapport rapport à E^{no} ssi aucune des séquences M_i et M_k ne se termine par un événement associé au nœud *1*. Autrement dit, aucun AGV n'a comme destination le nœud *1* et que sa mission contient au moins un nœud (avec un capteur opérationnel) après le nœud *1*. Le système d'AGVs est donc diagnosticable par rapport aux événements associés au nœud *1* si la suite de toute mission de transport passant par ce nœud est de longueur supérieure ou égale à un. Formellement, si $M_i = M_{i\_début} e_{1i} M_{i\_suite}$ alors $\|M_{i\_suite}\| \geq 1$ implique la diagnosticabilité de e_{1i}.

Proposition. *La condition nécessaire pour que le diagnostic de l'événement non-observable associé au nœud 1 du début du CC permette l'évitement de certains blocages est que le premier événement observable garantissant la diagnosticabilité soit associé à un nœud k interne au CC (i.e., $k \in \{2,..n-1\}$). On définira cette diagnosticabilité comme étant partiellement tolérante aux fautes.*

La diagnosticabilité est qualifiée de partiellement tolérante aux fautes car elle permet d'éviter le conflit sur le CC mais à condition qu'elle ait lieu avant qu'un autre AGV s'engage sur ce CC. Dans la suite de l'article, on supposera que le CC est constitué de plusieurs zones partagées.

4.3 Supervision avec prise en compte du diagnostic

Pour ne pas alourdir l'article, nous allons expliquer le principe de la supervision avec prise en compte du diagnostic pour le cas de l'événement e_{1i} mais le même raisonnement peut s'appliquer sur e_{1k}. Le premier, i.e., e_{1i}, étant plus problématique car il marque l'accès de V_i au CC alors que le second représente la sortie de V_k du CC. Selon (§4.2), e_{1i} est diagnoticable si M_{i_suite} est au moins de longueur un. Autrement dit pour toute séquence démarrant par l'événement e_{2i} associé au nœud *2* (avec $2 \in$CC). Rappelons que les nœuds ainsi que les zones appartenant au CC sont numérotés, selon le sens de parcours de V_i, de *1* à *n* (respectivement (*n*-1) pour les zones). Une zone *j* est délimitée par deux nœuds *j* et *j*+1 et son automate de spécification sera noté *Spec*(*j*). Voici les étapes de la méthode.

Etape1. Selon la méthode de supervision de (§3.2), l'événement non-observable e_{1i} apparait dans *Spec*(1) (Figure 5.a). L'événement diagnostiqueur e_{2i} qui permet de réinitialiser *Spec*(*1*), ne peut avoir lieu si e_{1i} n'est pas observé par *Spec*(*1*). e_{2i} est également présent dans *Spec*(*2*), car il marque la sortie

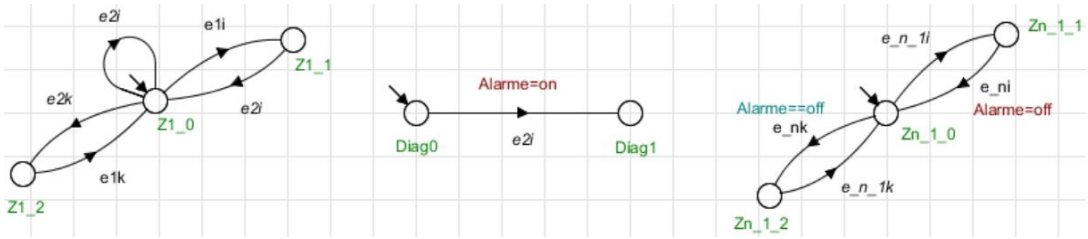
de la zone *1* (réinitialisation de *Spec*(*1*)) et l'entrée dans la zone *2*. Par conséquent, la synchronisation de *Spec*(*1*) et *Spec*(*2*) avec G_i va bloquer V_i qui ne pourra pas avancer dans la zone *2* (*Spec*(*1*) est bloquante).
Pour éviter cela, l'événement e_{2i} (qui permet de diagnostiquer e_{1i}) présent dans les deux spécifications doit-être rajouté à *Spec*(*1*) à travers une boucle sur l'état initial de celle-ci (Figure 5.a). Il indique que la zone *1* est libre mais après occurrence d'un événement non observé.

Etape2. Le diagnostiqueur sera simplement modélisé par un automate *Diag* à deux états. Celui-ci va émettre une alarme lorsque e_{2i} se produit (Figure 5.b). Le passage de l'état initial *Diag0* vers l'état *Diag1*, synchronisé sur e_{2i}, va produire une action sur une variable booléenne *Alarme* qui signifie que la détection de e_{1i} a bien eu lieu (*Alarme=On*).

Etape3. L'information issue du diagnostiqueur doit être incluse dans la spécification associée à l'autre zone extrême du CC et qui contient un événement contrôlable associé au nœud *n*, i.e., *Spec*(*n*-1) (Figure 5.c). En effet, si e_{1i} est détecté, alors V_k doit-être stoppé au nœud *n* et empêché par le superviseur d'entrer dans le CC. Pour cela, il suffit d'associer une garde à l'arc étiqueté par l'événement contrôlable e_{nk}. Cette garde porte sur la variable de détection *Alarme* qui doit-être nulle (*Off*). Cette même variable est mise à jour et réinitialisée dans *Spec*(*n*-1) par l'événement e_{ni} marquant le départ de V_i du CC.

Etape4. Faire la synthèse avec les nouvelles spécifications et le diagnostic en calculant ($G_i \parallel G_k \parallel Diag \parallel_{j=1,n-1} Spec(j)$).

Remarque : La méthode est présentée en considérant que le premier événement observable dans S_{i_suite} est e_{2i}, mais elle peut être généralisée à un autre premier événement observable e_{ji} avec $j \in \{2, n-1\}$. Les modifications précédentes concerneront alors les deux spécifications *Spec*(*j*-1) et *Spec*(*n*-1).



(a) Automate *Spec*(1) (b) Automate *Diagnostic* (c) Automate *Spec*(*n*-1)

Figure 5 : Modification des spécifications pour prendre en compte le diagnostic.

5 Modélisation dynamique et stochastique

Comme indiqué précédemment, nous souhaitons étudier notre système du point de vue de la sûreté de fonctionnement. Pour cela, nous avons construit des modèles comportementaux à base d'automates temporisés stochastiques. Le but étant d'une part d'évaluer l'impact des défaillances de certains capteurs sur la fiabilité du système d'AGVs et d'autre part, d'évaluer l'impact du diagnostic sur celle-ci. Nous définissons la fiabilité d'un système d'AGVs comme la probabilité que tous les AGVs remplissent leurs missions. Autrement dit, la probabilité d'absence de blocages empêchant les AGVs d'atteindre leurs destinations. Cette étude fiabiliste sera faite par *model-checking* statistique.

5.1 Les automates temporisés stochastiques

Afin de modéliser le comportement dynamique et temporel du système d'AGVs, comme le temps de parcours des zones, ainsi que le comportement aléatoire comme la défaillance de certains capteurs et les perturbations (obstacles fixes et mobiles, charge de la batterie, etc.) nous avons fait appel aux

automates temporisés stochastiques (ATS). Avant d'introduire le modèle, nous allons en rappeler la définition formelle.
Les automates temporisés stochastiques (ATS) sont une extension des automates temporisés définis par (Alur et Dill, 1994) et qui permettent le partage de variables, d'événements synchronisant et de modéliser des caractéristiques probabilistes (Alur et Dill, 1994). Formellement, un ATS est présenté comme le *n-uplet* $A=(L, V, E, C, Inv, P, T, L_m, l_0, v_0)$, où :

- L est un ensemble fini de localités.
- V est un ensemble fini de variables.
- E est un ensemble fini d'événements synchronisant avec $E= E_u \cup E_{nu.}$
  - E_u est un ensemble fini d'événements *urgents*. Un événement urgent permet de forcer l'automate à quitter une localité quand une transition peut être franchie.
  - E_{nu} est un ensemble fini d'événements *non urgents.*
- C est un ensemble fini d'horloges.
- Inv est un ensemble d'invariants.
- P est un ensemble de probabilités : ***i)*** discrète sur l'ensemble des transitions (à partir d'une localité, une transition peut conduire dans différentes localités l_i avec des probabilités p_i où $\sum p_i=1$ (noté par des arcs en pointillés). ***ii)*** continue sur des variables d'automates.
- T est un ensemble fini de transitions notées par $(l,e,g,m,l') \in (L \text{ x } E \text{ x } G \text{ x } M \text{ x } L)$ où l et l' sont respectivement les deux localités début et arrivée. Sur une transition trois éléments sont définis : ***i)*** Une garde $g \in G$, qui est une condition sur les variables pour valider le franchissement de la transition associée. ***ii)*** Une mise à jour $m \in M$, sur les variables ou les horloges. ***iii)*** L'événement de synchronisation $e \in E$.
- $L_m \subseteq L$ est l'ensemble des localités marquées.
- $l_0 \in L$ est la localité initiale de l'automate.
- v_0 est le vecteur d'initialisation des variables.

5.2 Modélisation du système d'AGVs par ATS

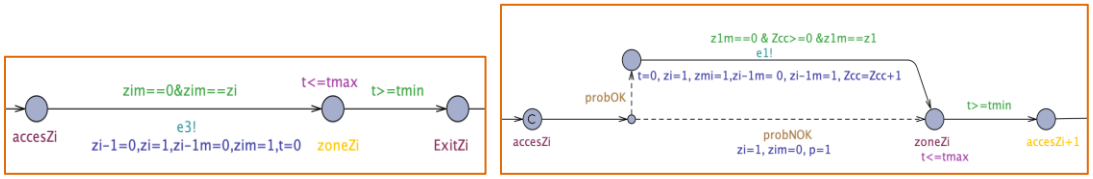
Le comportement logique de chaque AGV est modélisé par un ATS, selon le même principe expliqué précédemment au (§3.2) avec les automates à états finis. Néanmoins, il faut adapter le modèle afin de réaliser des opérations qui ne sont pas possibles avec les ATS et le logiciel UPPAAL. Des opérations comme la composition des automates, la définition des spécifications et la synthèse du superviseur.
Une spécification associée à une zone *Zi* sera modélisée par une variable globale qui prend la valeur de 1 si la zone est occupée et 0 sinon, ainsi que par des gardes qui seront associées aux transitions modélisant l'accès d'un AGV à cette zone. Ce sont également ces gardes qui synchroniseront les automates. Le comportement du superviseur qui, selon (§3.2), gère essentiellement l'accès des véhicules à leur chemin commun partagé en sens opposé, sera modélisé par une variable globale *Zcc* qui est nulle si ce chemin partagé est libre de tout AGV. Sinon, elle indique le nombre d'AGVs engagés dans un sens ou dans l'autre. Rappelons que nous utilisons le modèle de supervision avec les spécifications permissives (voir Section 3). La variable *Zcc* conditionnera les gardes associées aux transitions représentant les événements qualifiés de contrôlables dans le chemin commun (§3.2), autrement dit associés aux nœuds externes de ce chemin partagé.

La modélisation du comportement des AGVs se fait par l'intermédiaire de 3 modules élémentaires appelés patterns :

- Un pour un capteur considéré comme sûr et donc avec un événement observable (Figure 6.a).

- Deux patterns pour un capteur pouvant défaillir : (i) l'un avec un diagnostic et (ii) l'autre sans diagnostic (Figure 6.b).

Chaque pattern (Figure 6) est constitué de 3 localités {*accesZi*, *zoneZi*, *accesZi+1*} qui, pour un AGV donné, représentent respectivement l'attente devant la zone *Zi*, sa présence dans la zone *Zi*, et l'attente devant la zone suivante *Zi+1*.
Pour qu'un AGV accède à une zone appartenant à un trajet commun, il faut avoir l'autorisation du superviseur (représenté par la variable *Zcc*) qui vérifie que son chemin commun avec les autres AGVs est disponible (pas d'engagement en sens inverse). Un capteur peut subir une défaillance à tout moment provoquant la perte d'observabilité de l'événement qu'il mesure. Ainsi, la transition représentant le passage d'une zone à une autre qui est repérée par ce capteur non sûr est modélisée par deux arcs modélisant les deux possibilités de fonctionnement et ayant chacun une probabilité d'occurrence : **i)** accès à la zone avec un capteur fonctionnel (arc avec une probabilité *probOK*) ; **ii)** et accès à la zone avec un capteur défaillant (arc avec probabilité *probNOK*). Le premier cas impliquant l'observabilité de l'événement impliqué, toutes les variables de contrôle utilisées pour réaliser la supervision seront mises à jour en conséquence. Dans le second cas, l'événement associé n'étant pas observable, les variables de mesure et de contrôle restent inchangées (Figure 6). Dans le cas du pattern avec diagnostic, la variable de contrôle du chemin commun *Zcc* sera mise à jour dès que le système devient diagnosticable (voir Section 4) assurant ainsi une supervision plus sûre. Par exemple, lors du passage du capteur suivant.
Pour traverser une zone, un AGV met un temps aléatoire *t* compris dans un intervalle [*tmin*, *tmax*]. L'aspect aléatoire vient du fait que la vitesse de l'AGV dépend de son environnement et la présence d'éventuels obstacles comme des opérateurs qui traversent les pistes de guidages, l'état de charge de la batterie etc. Pour cela, des horloges sont mises en place et réinitialisées au moment de quitter et d'entrer dans une zone. Ainsi, un AGV pourra quitter sa zone actuelle (état *zoneZi*) et demander l'accès à sa zone suivante (état *accesZi+1*) uniquement s'il est resté dans l'état *zoneZi* au moins un temps égal à *tmin* qui correspond au temps de parcours minimal de cette zone. Ceci est modélisé par une garde [*t*≥*tmin*] associée à la transition modélisant la fin de parcours d'une zone (i.e., reliant *zoneZi* à *accesZi+1*). L'AGV ne doit pas non plus rester dans sa zone actuelle (i.e. état *zoneZi*) au-delà du temps maximale *tmax*. Ceci est modélisé par un invariant dans la localité *zoneZi* vérifiant que *t* ≤ *tmax*.



(a) Pattern d'un capteur sûr (observabilité) (b) pattern avec un capteur non sûr (perte d'observabilité)

Figure 6 : Patterns d'un déplacement élémentaire avec et sans observabilité

A partir de ces patterns, le modèle global du déplacement de l'AGV se fait simplement en concaténant le pattern à chaque capteur selon s'il est sûr ou non. La figure 7 donne l'exemple d'un ATS complet pour un trajet aller/retour avec 4 zones et 4 capteurs dont le premier peut être défaillant. Le model Checking sera par la suite utilisé pour l'évaluation des performances (Section 6).

Dans le domaine SED, un model-checker est couramment utilisé pour vérifier si un modèle satisfait ou non certaines propriétés. Ces propriétés peuvent être exprimées dans plusieurs langages logiques (CTL, PCTL, etc.). Pour évaluer le risque d'inter-blocage, nous utilisons dans la section suivante une extension du model-checking pour les modèles ATS qui est la logique PCTL (*Probabilistic Computation Tree Logic*) (Alur et al., 1999). Ce langage est une extension probabiliste de CTL (*Computation Tree Logic*). Ce type de logique permet d'exprimer des propriétés telle que "Quelle est la probabilité que le modèle soit dans l'état *A*, dans l'intervalle de temps [0,*T*]?". Cette question peut être transcrite en PCTL comme dans l'expression : *P*=?[*F*≤*T*"*A*"]. L'opérateur « *P*=? » exprime l'évaluation de probabilité, l'opérateur « *F* » signifie qu'il existe dans le futur un chemin où

l'automate est dans l'état « *A* ». Le Model Checking statistique (SMC) est utilisé pour obtenir cette probabilité sur un principe identique à celui de la simulation Monte Carlo (Bulychev et al., 2012).

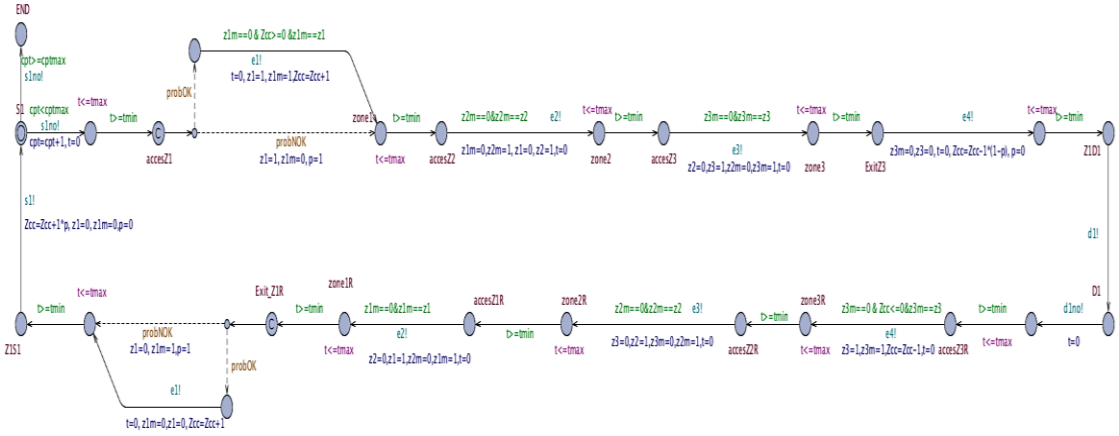


Figure 7 : Modèle ATS complet d'un AGV avec 4 capteurs

6 Etude en simulation

Nous avons établi deux modèles ATS d'un système de deux AGVs similaire à celui de la Figure 3, mais partageant 4 zones en commun. Le but est de comparer les performances fiabilistes des deux architectures de supervision, c.à.d sans et avec intégration du diagnostic de l'événement devenu non observable suite à la défaillance d'un capteur. Pour les deux modèles, nous avons considéré que les défauts pouvaient affecter un capteur associé à nœud extrême du chemin commun car plus critique comme expliqué en Section 4. Il s'agit du capteur *1*. La probabilité d'avoir absence de blocage est donnée par la propriété à vérifier par le model checker et exprimé par l'équation (1)

$$P=?[F\leq DureeMax\text{“}AGV1.END \;\&\&\; AGV2.END\text{ ”}] \qquad (1)$$

Ici *DureeMax* est le temps total nécessaire à la réalisation de toutes les missions en se plaçant dans le pire des cas où le temps de parcours de chaque zone est égal à t_{max}. Autrement dit, *DureeMax* est tel que : *DureeMax* ≥ (Nombre d'AGVs ∗ nombre de missions ∗ nombre de zones ∗ t_{max}).

Pour évaluer cette probabilité, il est nécessaire de définir une mission, par exemple effectuer un aller-retour et y associer un état. Dans la figure 7, les AGVs ont fini leur mission si la localité *END* est atteinte. Ainsi, la probabilité que tous les AGVs atteignent cet état (équation (1)) représente le fait qu'il n'y a pas eu d'inter-blocage entre AGVs et représente donc la fiabilité de l'AGVS.

Dans le but d'évaluer et d'analyser la fiabilité de l'AGVS, nous avons conduit des simulations sur les modèles ATS (Figure 7) pour différents taux de défaillance du capteur *1* et des durées de traversée de zones différentes. La figure 8 montre en exemple les résultats de simulation obtenus pour des temps de traversée de zones (i.e., des intervalles de séjour en localité-zone) de [*tmin*, tmax]=[14, 16].

Ces résultats montrent deux choses : d'une part la décroissance de la fiabilité du système d'AGVs avec l'augmentation de la probabilité de défaillance du capteur *1*. En effet, plus ce capteur est sujet aux défaillances plus le risque de conflit frontal sur une des zones partagées avec l'autre véhicule devient important. D'autre part, on voit l'amélioration de la fiabilité grâce à l'intégration du diagnostic dans la supervision. Cette amélioration est d'autant plus importante lorsque le capteur

devient plus défaillant. Par exemple, pour une probabilité de défaillance de 50% la fiabilité de la supervision sans diagnostic est de 49% alors qu'elle est de 65% quand le diagnostic est effectué.

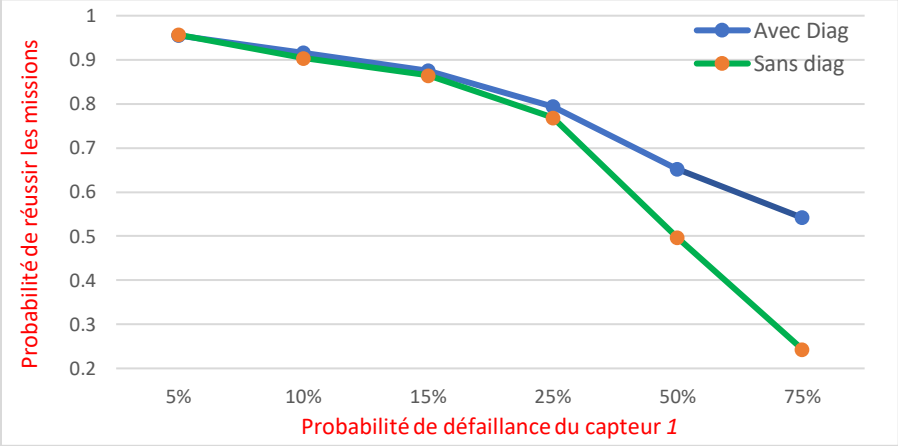


Figure 8 : Fiabilité du système d'AGVs sans et avec diagnostic.

7 Conclusion et perspectives

Dans cet article, nous nous sommes intéressées au problème de routage sans conflits d'un système d'AGVs bi-directionnels dans un contexte de défaillances de capteurs de position. L'objectif était d'une part d'étudier l'impact de ces défaillances sur la fiabilité du système, et d'autre part de proposer une architecture de supervision plus sûre. Pour ce faire, nous avons d'abord caractérisé la diagnosticabilité d'un système d'AGVs par rapport à des événements pouvant devenir non-observables suite à une défaillance capteur puis intégré la fonction de diagnostic dans l'architecture de supervision. Dans un but d'une évaluation fiabiliste des performances d'un tel système, nous avons modélisé celui-ci par automates temporisés stochastiques et utilisé le model checking pour évaluer la probabilité d'absence de blocage. L'analyse en simulation a démontré l'amélioration apportée par l'intégration du diagnostic dans la supervision. Cette amélioration dépend certainement de la méthode de diagnostic utilisée. Il est probable qu'un diagnostic basé sur le temps prévisionnel (méthode par template par exemple) donne de meilleurs résultats. C'est une des perspectives de ce travail. Une étude sur un système de taille plus importante avec des missions changeantes est également envisagée.

Références

Alur, R., & D. Dill. 1994. A theory of timed automata. *Theoretical computer science*, 126(2), 183-235.

Alur, R., L. De Alfaro., T.A. Henzinger & Y.C. Mang. 1999. Automating modular verification. In : *International Conference on Concurrency Theory*. Springer, Berlin, Heidelberg. pp. 82-97.

Arnaud, Y., J. Boimond., J. Cury., J. Loiseau., & C. Martinez. 2009. Using UPPAAL for the secure and optimal control of AGV fleets. In *7th Workshop on Advanced Control and Diagnosis ACD*, Poland.

Bulychev, P., A. David., K. Gulstrand Larsen, et al. 2012. UPPAAL-SMC: Statistical model checking for priced timed automata. *arXiv preprint arXiv:1207.1272*, 2012.

Cassandras C.G., & Lafortune S. 1999. Introduction to Discrete Event Systems. *Kluwer Academic Publishers*, Dordrecht, 1999, ISBN: 0-7923-8609-4.

Egbelu, P.J., & J.M.A. Tanchoco. 1986. Potentials for bi-directional guide-path for automated guided vehicle based systems. *International Journal of Production Research*, 24(5), pp. 1075-1097.

Fanti, P.M. 2002. Event-based controller to avoid deadlock and collisions in zone-control AGVS. *International Journal of Production Research*, 40(6), pp. 1453-1478.

Fanti, P.M., Mangini, A.M., G. Pedroncelli, & W. Ukovich. 2018. A decentralized control strategy for the coordination of AGV systems. *Control Engineering Practice*, (70) pp. 86-97.

Felko, I. 2011. Simulation-based deadlock avoidance and optimization in bidirectional AGVS. *Proceeding of the 4th International Conference on Simulation Tools and Techniques for Communication, Networks and Systems* (SIMUTOOLS'11), Barcelona.

Gamache, M., R. Grimard, & P. Cohen. 2005. A shortest-path algorithm for solving the fleet management problem in underground mines. *European Journal of Operational Research*, 166, pp. 497-506.

Girault, J., J-J. Loiseau, & O-H. Roux. 2016. On-line compositional controller synthesis for AGV. *Discrete Event Dynamic Systems*, Springer Verlag, 26(4), pp. 583-610.

Kim, C.W., Tanchoco. 1993. Operational control of bi-directional AGV routing. *International Journal of Production Research*, 29(12), pp. 2123-2138.

Krishnamurthy, N.N., R. Batta, & H. Karwan. 1993. Developing conflict-free routes for automated guided vehicles. *Operations Research*, 41(6), pp. 1077-1090.

Li, Q., A. Progmsky, T. Adriaansen, & J.T. Udding. 2016. A control of collision and deadlock avoidance for automated guided vehicles with fault-tolerance capability. *International Journal of Advanced Robotics Systems*. 13:64 / doi: 10.5772/62685.

Maza, S. 2019. Conflict-free fault-tolerant supervisory control of bi-directional AGV systems. *13th International Conference CIGI QUALITA*, Montreal, Canada, 25-28 June.

Maza, S., & P. Castagna. 2005a. A performance-based structural control policy for conflict-free routing of bi-directional Automated Guided Vehicles. *Computers in Industry*, 56(7), pp. 719-733.

Maza, S., & P. Castagna. 2005b. Sequence-based hierarchical conflict-free routing strategy of bi-directional automated guided vehicles. 16th *IFAC World Congress*, Prague, Czech Republic, 3-8 july.

Maza, S., & P. Castagna. 2007. On the use of automated guided vehicles in flexible manufacturing systems. *4th IFAC International Conference on Informatics in Control, Automation and Robotics*, Angers, France, 9-12 May.

Paoli, A., & S. Lafortune. 2005. Safe diagnosability for fault-tolerant supervision of discrete-event systems. *Automatica*, (41), pp.1335-1347.

Rajotia, S., K. Shanker, & J.L. Batra. 1998. A semi-dynamic time-window constrained routing strategy in an AGV system. *International Journal of Production Research*, 36(1), pp. 35-50.

Ramadge, P.J., & W.M. Wonham. 1987. Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25(1) pp. 260-230.

Reveliotis, S.A. 2000. Conflict resolution in AGV systems. *IIE Transactions*, 32(7), pp. 647-659.

Sampath, M., Sengupta, R., & Lafortune, S. 1995. Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, *40*(9), pp.1555–1575.

Yang, Y., M. Zhong., Y. Dessouky., & O. Postolache. 2018. An integrated scheduling method for AGV routing in automated container terminals. *Computers & Industrial Engineering*, 126, pp. 482-493.