Proposition d'une architecture logique d'un système de pilotage hétérarchique évolutif par apprentissage

Bousbia Salah

Thèse

Présentée pour l'obtention du titre de
Docteur de l'Université de Valenciennes et du Hainaut Cambrésis

Préparée au LAMIH : Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines, UMR CNRS 8530

Dans la spécialité Automatique et Informatique des Systèmes Industriels et Humains
Discipline : Automatique

Par
Salah Ben Hédi Bousbia
Ingénieur en Génie Textile (ENI, Monastir Tunisie)
DEA en Automatique et Informatique (UVHC Valenciennes)

Proposition d'une architecture logique d'un système de pilotage hétérarchique évolutif par apprentissage

Présentée publiquement le 12 Décembre 2006
Devant le jury composé de :

Alain Guinet	: Rapporteur	Professeur, Institut National des Sciences Appliquées de Lyon
Gérard Morel	: Rapporteur	Professeur, Centre de Recherche en Automatique de Nancy
Yannick Frein	: Examinateur	Professeur, École Nationale Supérieure de Génie Industriel de Grenoble
Patrick Pujo	: Examinateur	Maître de conférences, Laboratoire des Sciences de l'Information et des Systèmes, Université Aix-Marseille.
Philippe Thomin	: Examinateur	Maître de conférences, Université de Valenciennes et du Hainaut Cambrésis.
Christian Tahon	: Directeur de thèse	Professeur, Université de Valenciennes et du Hainaut Cambrésis.
Damien Trentesaux	: Codirecteur de thèse	Professeur, Université de Valenciennes et du Hainaut Cambrésis.

Dédicaces

Ce travail est le témoignage de ma reconnaissance et ma profonde gratitude envers mon père Hédi Bousbia, pour ses encouragements, sa patience, les principes qu'il m'a inculqués et qui, à aucun moment, n'a pas eu le moindre doute concernant l'aboutissement de ce travail : maintenant tu peux être pleinement fier de ton fiston !

À toute ma famille.

Remerciements

Le travail présenté dans ce mémoire a été réalisé au Laboratoire d'Automatique, de Mécanique et d'Informatique Industrielles et Humaines de l'université de Valenciennes et du Hainaut-Cambrésis, au sein de l'équipe Systèmes de production dirigée par le professeur Christian Tahon. Je tiens à lui exprimer toute ma gratitude pour m'avoir accueilli au sein de l'équipe.

Je tiens à exprimer mes vifs remerciements aux membres du jury d'avoir accepté d'évaluer le présent travail. Que Messieurs Alain Guinet et Gérard Morel trouvent ici le témoignage de ma reconnaissance d'avoir accepté de rapporter cette thèse. Je les remercie pour l'intérêt qu'ils ont porté à ce travail, le temps qu'ils y ont consacré et leurs remarques constructives qui m'ont aidé à poser un autre regard sur mon propre travail.

Mes sincères remerciements vont également à messieurs Yannick Frein et Patrick Pujo pour l'intérêt qu'ils ont porté à ce travail en acceptant de participer à mon jury de thèse en tant qu'examinateurs.

J'exprime ici toute ma reconnaissance et ma sympathie envers Philippe Thomin qui a contribué à la conception du simulateur informatique et pour avoir accepté d'examiner ce travail.

Je témoigne ici ma plus grande reconnaissance et profonde gratitude envers mon encadreur Damien Trentesaux pour son aide précieuse et les échanges scientifiques et amicaux que nous avons eus tout au long de cette collaboration. Son soutien, sa patience et ses conseils ont contribué à rendre ce travail très enrichissant sur le plan des connaissances et sur le plan personnel.

Que mes amis trouvent ici l'expression de toute ma gratitude, spécialement Mehdi Ben Tahar pour sa sympathie et ses encouragements continus, Nizar Aifaoui pour le soutien qu'il m'a toujours manifesté et Hervé Stachowiak pour son aide précieuse. Enfin, un grand merci du fond du cœur à mes amis Abderrazak, Karim, Bilel et Rabï pour leur précieuse aide morale et 'logistique' pendant ces derniers mois.

Table des matières

Table des figures

Table des tableaux

Introduction générale

La complexité et la variabilité du contexte actuel des Systèmes de Production de Biens et de Services (SPBS) imposent une vision dynamique des performances (techniques, économiques,...) de ces systèmes, c'est-à-dire des performances continuellement améliorées. Ce besoin d'une vision dynamique des performances des SPBS :

- est de plus en plus ressenti et exprimé par les acteurs de différents domaines de SPBS,
- s'inscrit dans le contexte plus global d'agilité,
- implique de mettre en œuvre des systèmes de pilotage adaptés.

D'une part, dans un contexte fortement concurrentiel, l'agilité devient une caractéristique clef de la prospérité d'un SPBS et symbolise l'actuel objet de compétition entre les SPBS. D'autre part, la mise en place d'un système de pilotage représente un moyen adéquat pour garantir l'agilité d'un SPBS. En effet, le pilotage, défini dans [Trentesaux, 02], 'consiste à décider dynamiquement des commandes pertinentes à donner à un système soumis à des perturbations pour atteindre un objectif donné décrit en termes de maîtrise de performances'. Cette définition précise en outre que la notion de maîtrise intègre non seulement celle de maintien d'un niveau de performance donné, mais également celle de progrès (évolution vers un niveau de performance souhaité ou amélioration continue). C'est sur cet objectif d'amélioration continue des performances des SPBS que nous nous focalisons dans cette thèse.

Cependant, les systèmes de pilotage actuels ne répondent pas efficacement à cette évolution permanente du contexte des SPBS et par conséquent l'objectif d'amélioration continue des performances reste difficilement atteignable. Plus précisément, les systèmes de pilotage sont généralement conçus pour répondre à un besoin spécifique et ce manque de généricité réduit fortement les propriétés d'agilité d'un SPBS.

Nos travaux de recherche s'inscrivent dans un cadre qui porte sur le développement d'un système de pilotage pour l'amélioration continue des performances des SPBS. En ce sens, grâce aux Technologies de l'Information et de la Communication (TIC), l'accès aux informations est de plus en plus simple et efficace. Cet accès offre l'opportunité de concevoir des systèmes de pilotage hétérarchiques qui favorisent la décentralisation des capacités décisionnelles.

L'objectif de ce mémoire est de proposer :

- une caractérisation du paradigme d'agilité des systèmes de production de biens et de services,
- un cadre de modélisation générique des systèmes de production de biens et de services,
- un système de pilotage hétérarchique évolutif par apprentissage pour l'amélioration continue des performances des SPBS.

Ce mémoire, composé de cinq chapitres, présente notre démarche avec le plan suivant :

Le premier chapitre est consacré à la présentation de notre travail (contexte, objectif et problématiques rencontrées). Nous présentons les différents concepts clefs relatifs à l'objectif d'amélioration continue des performances des SPBS. En particulier, nous proposons un cadre générique caractérisant l'agilité des SPBS. Un état de l'art montrera l'apport des différentes approches qui s'intéressent globalement ou partiellement à cet objectif et aux différentes problématiques.

En nous appuyant sur les analyses et constatations déduites du premier chapitre, nous établissons dans le deuxième chapitre un ensemble de spécifications d'un système de pilotage hétérarchique capable de garantir l'amélioration continue des performances des SPBS. Ces spécifications se résument globalement en trois points principaux : la diversification et la pertinence des données, l'évolutivité des prises des décisions et l'évaluation continue des performances.

À partir de cet ensemble de spécifications, nous proposons dans le troisième chapitre une approche de pilotage hétérarchique pour l'amélioration continue des performances des SPBS. Cette approche est fondée sur la décentralisation totale des capacités décisionnelles du système de pilotage. Les différentes entités composant le SPBS sont dotées d'autonomie décisionnelle et de mécanismes d'apprentissage. La spécificité de notre travail réside dans l'utilisation d'un ensemble de techniques issues ou qui ont donné des résultats intéressants dans d'autres domaines. En ce sens, nous avons intégré l'autonomie, des mécanismes génétiques et d'apprentissage dans le système de pilotage hétérarchique proposé.

Le quatrième chapitre propose une mise en œuvre informatique de l'ensemble des concepts présentés dans ce mémoire. Pour présenter la structure logicielle orientée objets que nous avons développée, nous utilisons le langage UML afin de modéliser les différents éléments du système de pilotage proposé.

Dans le cinquième et dernier chapitre, nous définissons et exploitons un exemple de SPBS sur lequel nous avons évalué et validé le modèle de pilotage et les différents algorithmes décisionnels présentés dans ce mémoire. Enfin, nous présentons des conclusions à notre travail et nous indiquons les perspectives de recherche qu'offre notre contribution.

CHAPITRE I : AMELIORATION CONTINUE DES PERFORMANCES DES SYSTEMES DE PRODUCTION DE BIENS ET DE SERVICES : ETAT DE L'ART

Actuellement, les systèmes de production de biens et de services (SPBS) opèrent dans un environnement dynamique caractérisé par une forte concurrence et une grande fluctuation des marchés. Par conséquent, les SPBS doivent faire face simultanément à ces deux contraintes. En d'autres termes, un SPBS doit être agile pour pouvoir 'prospérer dans un environnement compétitif en constante évolution' [Gunasekaran, 99]. Pour assurer la prospérité d'un SPBS, une solution consiste à garantir une amélioration continue de ses performances. L'objectif de notre travail de recherche est de contribuer à cette amélioration. Le but de ce premier chapitre est :

- de présenter le contexte dans lequel opèrent les SPBS actuels. Cette présentation nous conduit à nous intéresser à un paradigme plus global : l'agilité. Nous caractérisons ce paradigme avec un cadre formalisé en précisant les relations existant entre agilité, performance et amélioration continue des performances,
- d'identifier les différentes problématiques rencontrées pour atteindre cet objectif. Ces problématiques sont réparties en catégories selon leurs interférences avec les activités d'un processus décisionnel (pilotage),
- de dresser un état de l'art des différentes approches qui traitent globalement ou partiellement l'amélioration continue des performances. Les différentes références sont analysées par rapport à l'objectif recherché ainsi qu'aux problématiques qu'elles traitent.

1. Cadre de notre étude

Le cadre général de notre travail est l'étude des Systèmes de Production de Biens et de Services (SPBS). Cette appellation recouvre la majorité des systèmes rencontrés dans les domaines industriels et des services. Dans le tableau Tab. 1, nous illustrons cette distinction entre 'produit' et 'service' :

	Domaine	
	production de 'biens'	production de 'services'
Exemples	fabrication (assemblage, usinage, …)	transport (terrestre, aérien, maritime), logistique (hospitalière, alimentaire,…)

Tab. 1 Exemples de système de production de 'biens' ou de 'services'

1.1. Contexte actuel des systèmes de production de biens et de services

Le contexte industriel actuel est caractérisé par des mutations socio-économiques diverses et variées. Désormais, les SPBS opèrent dans un environnement incertain, en constante évolution et de plus en plus complexe. Globalement, ce contexte peut être caractérisé par les trois points suivants :

- des exigences continues et évolutives exprimées par les différents 'acteurs' des SPBS (producteurs, clients, fournisseurs, …). Ces exigences sont le résultat de plusieurs facteurs tels que l'intensification de la concurrence, la segmentation et l'évolution de la demande ou encore la personnalisation des produits ou des services. Ce point représente la motivation principale des efforts d'amélioration entrepris par ces acteurs. L'**objectif** consiste alors à satisfaire l'ensemble de ces exigences,
- un ensemble de perturbations (ou aléas) qui affectent le fonctionnement des SPBS. Ces perturbations, d'origines internes ou externes, dégradent considérablement le fonctionnement de ces systèmes. La présence de ces perturbations constitue le **problème** le plus pénalisant qui affecte le rendement d'un SPBS,
- des progrès technologiques en perpétuelle évolution dans le domaine de la communication ou du traitement de l'information. Leur utilisation judicieuse constitue un **moyen** pour satisfaire les différentes exigences et faire face aux perturbations rencontrées.

De ce fait, les différents décideurs des SPBS doivent trouver impérativement une adéquation entre ces trois points (objectifs, problèmes, moyens).

1.2. Analyse des besoins

Les SPBS doivent disposer des conditions favorables pour que leurs fonctionnements s'adaptent à l'évolution continue de ce contexte. En d'autres termes, il s'agit de garantir la prospérité d'un SPBS en faisant face efficacement aux différentes perturbations et en exploitant avantageusement les progrès technologiques actuels. Sans être exhaustive, la liste ci-dessous donne les conditions essentielles pour atteindre l'adéquation (objectifs, problèmes, moyens) recherchée :

- exploitation judicieuse et optimale des ressources (moyens de production mis à disposition,

investissements,…) disponibles pour assurer le fonctionnement d'un SPBS, c'est-à-dire, éviter leur sous-exploitation,

- intégration accrue des technologies actuelles qui sont caractérisées par leurs coûts de plus abordables et par leur miniaturisation. Parmi celles-ci, citons les technologies relatives aux systèmes de télécommunications (dont Internet est la plus répandue), aux systèmes d'identification (Tags électroniques, RFID[1]), aux systèmes de manutention (robots manipulateurs) ou encore aux systèmes de transfert (tapis roulants, chariots AGV[2]),
- réactivité face à une demande de plus en plus variable en terme de volume et de nature (changement de gamme, introduction d'ordres de fabrication urgents, pannes machines, rupture des stocks, évolutivité des produits ou des normes…),
- maîtrise et réduction des coûts de fabrication, passant obligatoirement par une exploitation optimisée des différentes ressources,
- développement du 'retour d'expérience' consistant à exploiter l'expérience acquise au cours du fonctionnent d'un SPBS afin d'améliorer le savoir-faire et par la suite améliorer ses performances,
- qualité (du produit ou du service) qui représente actuellement un facteur déterminant de la compétitivité d'un SPBS. Elle demande une maîtrise totale des processus de fabrication et un savoir-faire optimal,
- respect des exigences environnementales de plus en plus strictes (limitations des déchets et rejets, processus de fabrication moins polluants, …).

Dans cette optique, l'agilité représente un paradigme global qui recouvre la majorité des besoins mentionnés ci-dessous. Il exprime une nécessité vitale pour un SPBS engendrée à la fois par le contexte actuel et par l'obligation de satisfaire les besoins énumérés précédemment. Le paragraphe suivant est consacré à sa présentation.

1.3. Agilité

L'agilité est un paradigme qui a été introduit au début des années 90 par un groupe de chercheurs de l'institut Iacocca à l'université de Lehigh, Pennsylvanie. Depuis, plusieurs travaux ont été publiés sur ce sujet, notamment Yusuf et al. [Yusuf et al., 99] qui ont tenté de recenser les définitions existantes de l'agilité. Ils constatent que la relative nouveauté de ce paradigme, conduit à une forte diversité dans sa définition, correspondant aux multiples points de vue des différents auteurs. En annexe 1, une présentation succincte de ce paradigme est donnée selon plusieurs points de vue.

Dans notre travail, nous reprenons la caractérisation donnée par Gunasekaran [Gunasekaran, 99] qui définit l'agilité des systèmes de production comme leur capacité à prospérer dans un environnement compétitif en constante évolution. Par ailleurs, l'auteur considère les systèmes de production agiles comme une évolution des systèmes de production 'lean[3]' (Agarwal et al. [Agarwal et al., 05] emploient le terme 'leagile' comme une continuité des deux paradigmes).

1.3.1. Analyses

[1] Radio Frequency IDentification
[2] Aided Guided Vehicles
[3] Dans ce type de systèmes, il s'agit d'éliminer tous les excès dans les processus de production (les opérations non utiles) y compris le niveau des stocks.

L'analyse des différents travaux qui s'intéressent au paradigme d'agilité (voir [Gunasekaran, 99], [Giachettia et al., 03], [Cagliano et al., 04], [Elkins et al., 04], [Agarwal et al., 05], …) fait émerger les constats suivants :

- ces références ont un socle de définitions commun établi autour de notions telles que la productivité, la qualité, le prix, la personnalisation des produits avec une forte valeur ajoutée et la réponse aux changements et perturbations (au sens large),
- la majorité des travaux s'intéresse à l'agilité d'un type spécifique de systèmes de production de biens ou de services. en effet, il s'agit dans la majorité des cas d'un groupement d'entreprises en réseau (entreprises virtuelles, chaîne logistique, …). La multiplication des dépendances, les alliances stratégiques ou les fusions entre plusieurs entreprises peut justifier cette tendance. Dans cette même logique, peu d'intérêt est accordé à l'agilité des éléments de base d'un SPBS (produits, unités de production) ce qui représente à notre avis un obstacle devant la généralisation du paradigme de l'agilité,
- dans ces travaux, l'objectif de l'agilité est souvent lié à la garantie des performances globales de ces systèmes. Cette garantie représente la finalité principale du paradigme de l'agilité,
- la plupart des références s'intéressent principalement à l'agilité sous un angle structurel et/ou organisationnel. Généralement, l'agilité ou non d'un SPBS est souvent liée à la présence ou non de moyens de production 'sophistiqués' et bien organisés.

Ces différents constats nous suggèrent de proposer un cadre conceptuel générique capable de caractériser le paradigme d'agilité pour les SPBS d'une façon globale. Ce cadre servira de support à notre proposition.

1.3.2. Proposition d'une caractérisation de l'agilité

Nous proposons, dans notre travail, une caractérisation de l'agilité des SPBS qui est faite suivant deux points complémentaires : la nature du SPBS étudié et le type d'agilité.

- la nature du SPBS étudié (voir Figure 1), nous distinguons trois classes d'agilité :
  - agilité de classe I : elle concerne les SPBS étendus ou en réseaux de type chaînes logistiques, entreprises virtuelles, alliance d'entreprises, … C'est le type d'agilité qui est le plus étudié actuellement dans la littérature correspondante (voir par exemple [Frayret et al., 01], [Cagliano et al., 04], [Ip et al., 04], [Agarwal et al., 05], [Yusuf et al., 99]). Le but de ce type d'agilité est souvent de synchroniser le fonctionnement des différents maillons du réseau ainsi que l'adéquation des compétences et des savoir-faire pour atteindre un objectif commun (par exemple synchroniser le fonctionnement d'une firme industrielle composée de plusieurs divisions),
  - agilité de classe II : elle concerne les SPBS mono-entreprise. Cette agilité consiste à optimiser l'utilisation et le fonctionnement des moyens de production (physiques, humains). Généralement, il s'agit d'utiliser judicieusement les capacités disponibles (flexibilité, compétences) afin de maximiser la productivité d'un SPBS. Parmi les travaux relatifs à ce type d'agilité citons [Rabelo et al., 99] relatif à un shopfloor, [He et al., 01] et [Yang et Li, 02] pour la production personnalisée, ou encore [Elkins et al., 04] pour l'industrie automobile,
  - agilité de classe III : (que Yusuf et al. [Yusuf et al., 99] qualifient d'élémentaire ou fondamentale) elle concerne les différentes entités qui composent un SPBS (unités de production, produits, …). Il s'agit de concevoir des entités agiles capables d'opérer collectivement d'une façon autonome afin d'atteindre un objectif commun. Comme nous

allons le montrer ultérieurement, ce type d'agilité est très peu étudié actuellement.

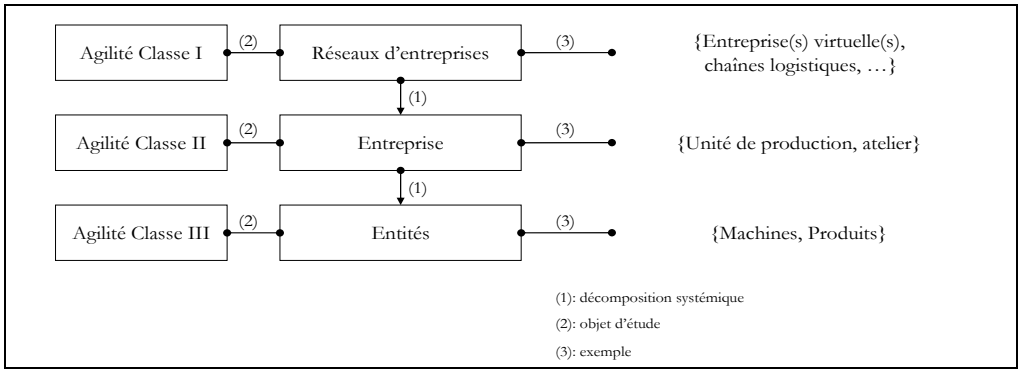


Figure 1- Classification de l'agilité selon la typologie du SPBS étudié

- le type d'agilité : nous définissons un cadre conceptuel du type d'agilité qui s'inspire largement de l'approche systémique (structure, activité, évolution) développée par Le Moigne [Le Moigne, 94][4]. Ainsi, le paradigme d'agilité peut être considéré comme la combinaison de trois types :
  - agilité structurelle : elle concerne le système physique (le SPBS). Cette agilité détermine si un SPBS possède l'infrastructure (les moyens de production) nécessaire et suffisante pour réaliser la ou les missions pour lesquelles il a été conçu. À titre d'exemple, un SPBS contenant des unités de production polyvalentes ou des outils de transfert modernes possède une agilité structurelle forte,
  - agilité opérationnelle : elle définit l'aptitude d'un SPBS d'une part à atteindre un objectif fixé (poursuite) et, d'autre part, à réagir en présence de perturbations (régulation). L'agilité opérationnelle exploite les moyens offerts par l'agilité structurelle pour assurer cette poursuite/régulation,
  - agilité évolutionniste : elle s'appuie sur les deux types d'agilité précédents et dépasse le cadre restreint d'une simple réaction face aux perturbations ou la poursuite d'un objectif. En effet, ce type d'agilité ambitionne d'assurer, d'une façon continue au cours du temps, une exploitation optimale du SPBS.

Ainsi, nous pouvons représenter ces trois types d'agilité avec un repère (x,y,z), voir Figure 2 :

Sur l'axe x, nous représentons à partir de quel moment du fonctionnement d'un SPBS apparaît chaque type d'agilité :

- l'agilité structurelle correspond à la phase de conception du SPBS (début de son cycle de vie) et constitue un pré-requis pour tout SPBS afin de réaliser les objectifs pour lesquels ce système a été conçu,
- l'agilité opérationnelle commence avec le début fonctionnement (ou exploitation) du SPBS,
- enfin, un SPBS acquiert une agilité évolutionniste uniquement après un intervalle de temps : cet intervalle exprime le temps nécessaire à l'acquisition d'un savoir faire (retour d'expérience).

Sur l'axe y, nous représentons la quantification ou le degré de chaque type d'agilité. Toutefois, deux précisions doivent être apportées :

[4] Micouin [Micouin, 06] note que la méthode SAGACE, développée par Jean-Marie Penalva [Penalva, 97], en référence au cadre épistémologique et méthodologique défini par Jean Louis Le Moigne, représente la seule méthode d'ingénierie des systèmes qui se situe explicitement dans le courant de pensée systémique.

- premièrement, les trois types d'agilité ne sont pas mesurables sur une même échelle (et par conséquent elles ne sont pas comparables),
- deuxièmement, le degré de chaque type d'agilité peut varier au cours du temps (augmenter ou diminuer). C'est pour des raisons de lisibilité de la Figure 2, que nous n'avons utilisé qu'un seul axe y (pour le degré) et représenté une amplitude constante de chaque type au cours du temps.

Enfin, sur l'axe z, sont représentés les trois types d'agilités.

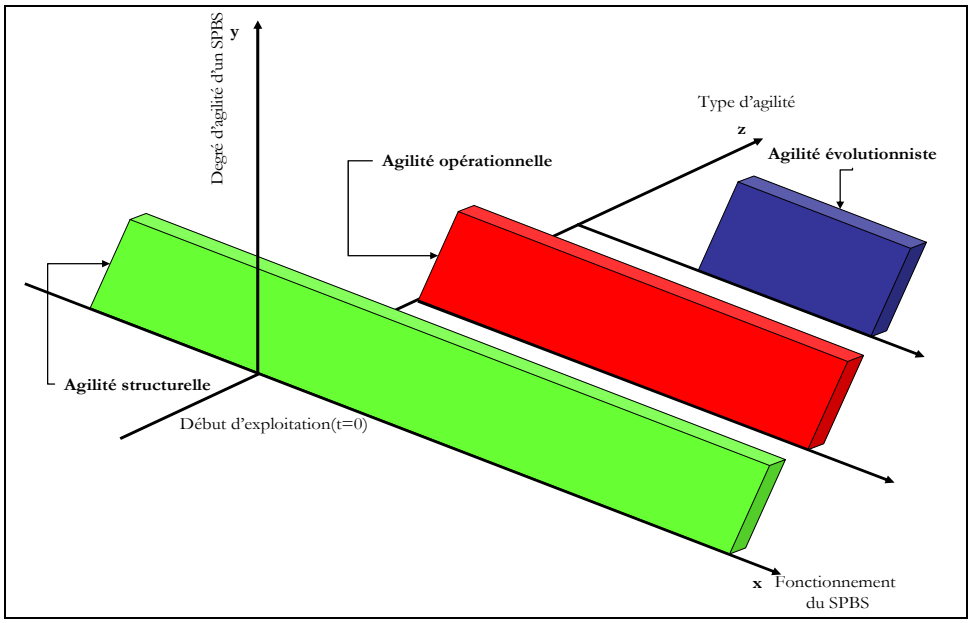


Figure 2- Représentation des trois types d'agilités (structurelle, opérationnelle, évolutionniste) d'un SPBS

Pour résumer, cette caractérisation fournit un cadre générique (<classe, type>) pour l'agilité d'un système de production de biens et de services. Le tableau Tab. 2 donne un positionnement d'un ensemble références bibliographiques grâce à cette caractérisation :

		Type de l'agilité		
		Structurelle	Opérationnelle	Évolutionniste
Classe de l'agilité	Classe I	[Giachettia et al., 03] (chaîne logistique)	[Agarwal et al., 05] (chaîne logistique) [Rabelo et al., 99], [Ip et al., 04] (entreprises virtuelles) [Cagliano et al., 04] (chaînes d'approvisionnement) [Frayret et al., 01] (réseaux d'entreprises)	-[5]
	Classe II	[Yang et Li, 02], [Sharifi et Zhang, 99]	[He et al., 01] (ordonnancement de machines parallèles identiques) [Elkins et al., 04] (industrie automobile)	-
	Classe III	-	[Sallez et al., 04]	-

Tab. 2 Classification des travaux relatifs à l'agilité selon la caractérisation <classe, type>

L'agilité est considérée actuellement comme une garantie nécessaire et suffisante pour avoir un SPBS performant. En effet, nous constatons qu'il existe une relation étroite entre l'agilité en tant que paradigme global et la garantie d'une performance en tant qu'objectif final (voir le troisième point du § 1.3.1). C'est sur cette notion de performance que nous nous focalisons dans ce travail. Plus précisément, nous nous intéressons à l'amélioration continue des performances des SPBS. De ce fait, il nous paraît primordial de présenter d'abord le concept de performance des

[5] À notre connaissance, il n'existe pas de références.

SPBS avant de discuter du concept d'amélioration continue des performances.

1.4. De la performance, …

Jacot [Jacot, 90] assimile la performance à un compromis entre pertinence, efficience, efficacité et effectivité. Avec un point de vue similaire, Sénéchal [Sénéchal, 04] identifie la performance globale à l'obtention conjointe de la pertinence, de l'efficience et de l'efficacité, appréciée en termes de coûts et de valeur, sur l'intégralité du cycle de vie d'un système. Bescos [Bescos et al., 95] identifie les concepts d'efficacité, d'efficience et de pertinence à partir du triplet (objectifs, résultats, moyens). Pour les trois auteurs, les concepts de pertinence, d'efficience et d'efficacité sont définis comme suit :

- l'efficacité caractérise l'écart entre les objectifs à atteindre et les résultats obtenus : est-on arrivé à ce que l'on avait l'intention de faire et à quel point l'objectif fixé est-il atteint ? Si l'efficacité du système n'est pas satisfaisante, les actions possibles peuvent porter sur l'organisation interne du système ou concerner le système de pilotage (la notion du pilotage sera présentée dans le paragraphe 1.6.). Par exemple si nous fixons, pour un système de conditionnement, un objectif de production de 5000 bouteilles/heure et que, réellement, ce système produit 6000 bouteilles/heure alors nous dirons que ce système est efficace[6],
- la pertinence concerne l'adéquation entre les moyens mis en œuvre et les objectifs. Son évaluation répond à la question suivante : les moyens mis en œuvre correspondent-ils aux objectifs ? La réponse à cette question est fondamentale en phase de conception du système de production. En effet, la pertinence permet d'éviter le surdimensionnement coûteux et la mise à disposition des moyens suffisants pour atteindre un niveau de satisfaction fixé. Par exemple, si ce système de conditionnement est composé de quatre machines d'embouteillage d'une capacité de 2000 bouteilles/heure chacune et que l'objectif de production est fixé à 10000 bouteilles/heure, alors ce système n'est pas pertinent,
- l'efficience mesure le 'rendement' du système par comparaison entre les moyens mis en œuvre et les résultats obtenus : est-ce que les résultats sont suffisants compte tenu des moyens mis en œuvre ? L'efficience est primordiale en phase d'exploitation du système de production. En effet, si manque d'efficience est observé ce sont alors les décisions de pilotage ou de management qui doivent être réajustées, l'efficience caractérise la différence entre les capacités d'un système et les résultats réellement obtenus. Par exemple, si avec quatre machines capables de traiter 10000 bouteilles/heure, la production est égale à 6000 bouteilles/heure alors nous pouvons conclure qu'il y a un problème d'efficience dans ce système de conditionnement.
- enfin, l'effectivité est l'adéquation des objectifs, des moyens et des résultats au regard de la finalité du système.

Pour formaliser la relation entre ces différents concepts, nous avons adopté une représentation 'automaticienne' de la performance, voir Figure 3. Cette figure montre les différentes relations qui existent d'une part entre le triplet (objectifs, résultats, moyens) et le triplet (efficacité, pertinence, efficience) et d'autre part entre la performance et l'efficacité, la pertinence et l'efficience. La partie suivante établit la relation entre la caractérisation proposée précédemment du paradigme de l'agilité et la performance.

[6] À condition que ce dépassement apporte un plus à l'entreprise, parfois il n'est pas forcément efficace de produire plus que souhaité.

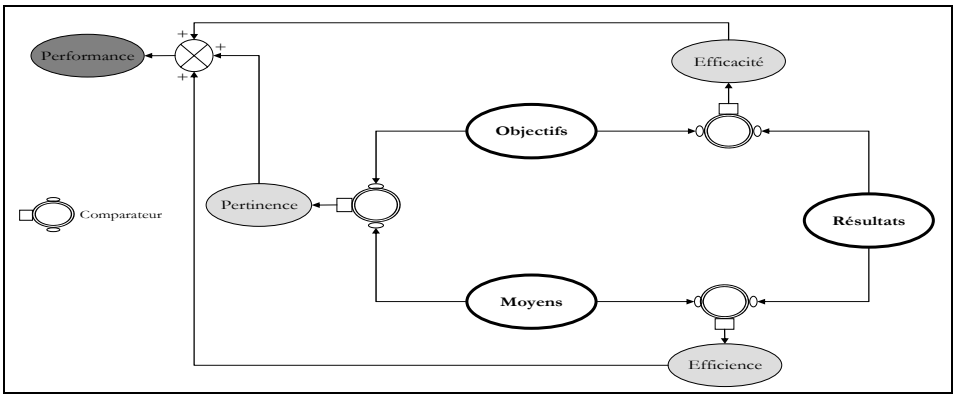


Figure 3- Représentation de la performance comme combinaison de (pertinence, efficacité, efficience)

Selon cette définition de la performance (combinaison d'efficacité, de pertinence et d'efficience), une relation peut être établie entre la caractérisation de l'agilité des SPBS que nous avons proposée et la performance, voir Figure 4 :

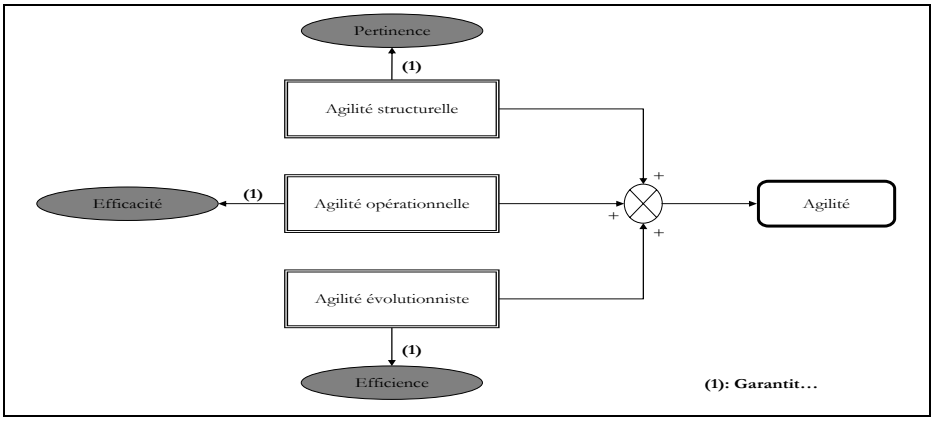


Figure 4- Relation entre les trois types d'agilité et le triplet (efficacité, pertinence, efficience)

Dans ce sens, l'agilité structurelle peut être vue comme une garantie pour obtenir un SPBS pertinent. L'agilité opérationnelle, à travers ses capacités d'action/réaction assure l'efficacité d'un SPBS. Enfin, l'efficience peut être considérée comme la finalité de l'agilité évolutionniste. Ainsi, sur la Figure 4, l'agilité est une combinaison (somme) d'agilités structurelle, opérationnelle et évolutionniste. Par ailleurs, nous avons montré dans le tableau Tab. 2, qu'actuellement, l'agilité est principalement structurelle et/ou opérationnelle et qu'elle est rarement évolutionniste. En conséquence, la garantie de la performance d'un SPBS consiste à assurer sa pertinence et son efficacité sans s'attacher à améliorer continuellement ses performances (efficience). La Figure 5 résume la relation actuelle qui existe entre l'agilité et la performance :

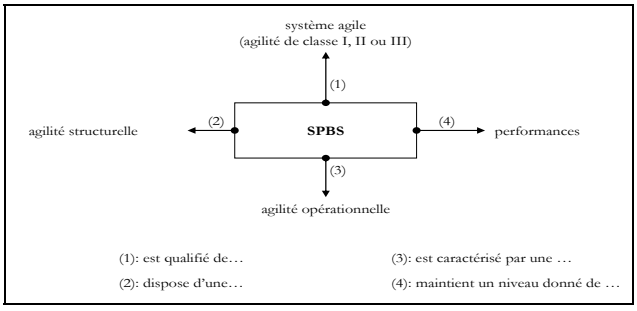


Figure 5- Relation agilité/performance (finalité actuelle d'un SPBS)

Cependant, les systèmes de production de biens et de services évoluent dans un contexte en constante évolution. Par conséquent, l'agilité en tant que concept global doit permettre aux SPBS de s'adapter constamment à ce contexte, c'est-à-dire garantir une amélioration continue des performances du SPBS (objectif). La partie suivante traite plus particulièrement de cet aspect, en présentant à la fois plusieurs points de vue sur l'amélioration continue des performances des SPBS ainsi que le nôtre.

1.5. … à l'amélioration continue des performances

1.5.1. Une double présentation académique et industrielle

Dans le domaine des systèmes de production de biens et de services, une analyse des approches abordant directement ou indirectement l'amélioration continue des performances permet d'identifier deux points de vues, distincts mais complémentaires, issus des domaines académique et industriel.

Sur le plan académique, Choi [Choi, 95] dresse un état de l'art sur le concept de l'amélioration continue (continuous improvement) suivant un point de vue orienté management. Cette étude critique l'intérêt marqué à des améliorations opérationnelles (en liaison avec le facteur humain) en soulignant le peu d'importance accordée aux aspects organisationnels.

L'étude présentée dans [Selen et Ashayeri, 01] et réalisée sur un système de production réel (industrie automobile), tente d'établir une relation entre l'amélioration des performances et les paramètres intrinsèques d'une chaîne d'assemblage (temps opératoires, taille des zones de stockage, temps de maintenance MTTR, MTBF[7]). Les résultats obtenus, en faisant varier ces paramètres, montrent l'impact de chaque paramètre sur l'amélioration des performances (exprimée en terme de nombre de voitures assemblées par jour). Cette référence a le mérite de mettre en avant l'apport de la simulation comme un outil incontournable pour quantifier l'apport des approches de pilotage (surtout les nouvelles), en terme d'amélioration continue des performances.

Dans un cadre plus théorique, [Vits et Gelders, 02] établissent les relations entre l'amélioration continue des performances et l'apprentissage. Les auteurs distinguent trois facteurs pouvant conduire à l'amélioration des performances : l'apprentissage des opérateurs et leur acquisition de nouvelles compétences, l'investissement dans de nouveaux moyens de production et enfin l'apprentissage lors du processus de pilotage.

Sur le plan industriel, un ensemble d'approches consacrées à l'amélioration continue des performances des SPBS existe. Dans cet ensemble, nous distinguons les deux approches d'amélioration continue les plus répandues : le Kanban [Ohno, 88] et le Kaizen [Yamamoto et al., 01]. Ces deux méthodes représentent les éléments précurseurs de la philosophie d'amélioration continue des performances des SPBS et sont désormais très courantes (industrie automobile, distribution, etc…). Nous présentons dans ce qui suit ce que les principales méthodes Kanban, Kaizen et Rex [Malvache et Prieur, 95][8] apportent pour l'amélioration continue des performances, sur quels facteurs de la performance elles ont un impact, ainsi que leurs limites.

L'avantage d'un système Kanban est d'être auto-améliorant. En effet, après une période de mise au point, le processus d'amélioration de l'écoulement du flux se fait par des actions telles que la réduction des temps de préparation, réduction des dysfonctionnements des postes ou encore l'accroissement de la flexibilité des opérateurs. Les quatre points suivants représentent ce que peut

[7] Mean Time To Repair et Mean Time Between Failure.

[8] Une brève description de ces différentes méthodes est donné en annexe 1.

apporter un système Kanban en terme d'amélioration continue des performances :

- une capacité de réaction et une flexibilité accrues (meilleur suivi du marché, capacité à traiter les commandes urgentes, planification de production sur un horizon court avec des commandes fermes),
- une réduction des coûts obtenue par l'organisation des activités (baisse des coûts de la non-qualité, transmission de l'information simplifiée…),
- une réduction des stocks et des en-cours,
- une réduction des besoins d'investissements et des charges d'entretien (simplification des systèmes informatisés de gestion de production, réduction des moyens de stockage,...),
- la méthode Kanban est bien adaptée à des productions de type 'masse' pour lesquelles le nombre de références n'est pas trop élevé et la demande est régulière ou à faibles variations.

Le principal problème relatif à la mise en place d'un système Kanban réside dans le dimensionnement, c'est-à-dire le nombre de kanbans qui circulent dans le système. Il s'agit de trouver un compromis entre le risque d'une rupture de stock et l'immobilisation financière due à un niveau de stockage trop élevé. En outre, étant donné qu'aucune décision n'est réellement prise au sein des différentes mailles (centres de pilotage locaux), le Kanban s'apparente plus à une commande décentralisée de production ou à un mécanismes auto-régulé et complètement déterminé, sans réels autonomie et degré de liberté [Trentesaux, 02].

La philosophie Kaizen consiste à améliorer continuellement la productivité et la qualité d'un SPBS. Cette amélioration est rendue possible grâce à la participation de tous les intervenants dans un SPBS, quel que soit leur rang, de faire connaître leurs observations et propositions (bonnes ou mauvaises). Ces propositions seront évaluées par un comité compétent et les suggestions retenues et mises en application se voient généralement récompensées.

La méthode Rex est une méthode de gestion des connaissances initialement conçue au Commissariat à l'Energie Atomique (CEA) pour capitaliser l'expérience accumulée dans cette entreprise. D'un point de vue pratique, la méthode Rex peut être perçue comme un ensemble de procédures qui dirigent et assistent l'explicitation, le recueil, l'organisation et la valorisation des connaissances et des expériences d'une entreprise. Une des originalités de la méthode Rex est d'encourager la structuration de la terminologie du domaine de connaissance capitalisé par la construction de modèles représentant le fonctionnement d'un SPBS. Ainsi, les modèles Rex contribuent à l'amélioration du double processus de recherche et d'interprétation des connaissances. En contraignant à une définition précise des termes utilisés, ils permettent la maîtrise de la qualité des échanges et du sens des informations capitalisées.

1.5.2. Constats

De ces deux points de vue industriel et académique, nous pouvons mettre en avant les constats suivants :

- pas d'unanimité autour d'une définition formelle et pragmatique : qu'est ce que l'amélioration continue des performances ? Certains points de vue considèrent que l'amélioration fait partie du domaine des études cognitives (comportement des ouvriers par exemple) tandis que d'autres auteurs s'accordent de lier cette finalité à l'aspect organisationnel d'un système,
- même si plusieurs références traitent d'une façon globale cette thématique de recherche (souvent il s'agit de réduire les délais et/ou les coûts), le qualificatif 'continue' qui doit

caractériser cette amélioration est généralement absent des différentes approches proposées (voir par exemple [Selen et Ashayeri, 01], [Dupas, 04]). Dans ces travaux, le but est plutôt de proposer un ensemble d'algorithmes de calcul (par exemple en proposant des méthodes d'ordonnancement) ou une structure de pilotage sans s'attacher réellement à assurer la continuité de cette amélioration,

- il existe deux écoles d'amélioration continue des performances : la première s'appuie sur un système de pilotage pour assurer cet objectif tandis que la deuxième représente plutôt une philosophie (ou un état d'esprit). La première approche est parfaitement représentée par les systèmes de production dits à Kanban et la deuxième est symbolisée par l'approche Kaizen,
- nous constatons que cette philosophie d'amélioration continue est beaucoup plus ancrée dans le monde industriel (le Kaizen) mais qu'elle s'intéresse majoritairement à des aspects reliés globalement à l'organisation d'un SPBS : agencement des unités de production, comportement des opérateurs humains, d'où une grande composante cognitive. Nous pensons que, tant pour le système Kanban que pour le Kaizen, il s'agit de deux philosophies de production (ou de management) qui s'appuient fortement sur l'implication déterminate des opérateurs humains et n'offrent pas un modèle[9] générique applicable sur les différents SPBS. En revanche, ces approches industrielles mettent l'accent sur l'importance de l'apprentissage des opérateurs humains. Néanmoins, ces efforts consomment un temps relativement long entre les démarches administratives et le retour d'informations. En effet, nous remarquons que le délai entre la proposition, la validation et l'application des nouvelles propositions est important,
- ces approches mettent l'accent sur l'importance d'un retour d'expériences (méthode Rex) acquises au cours du fonctionnement d'un SPBS. En effet, une fois récoltées et analysées, ces expériences passées peuvent être exploitées afin d'adapter le fonctionnement d'un SPBS au contexte dans lequel il opère.

1.6. Motivations et objectif de notre travail

En considérant les dimensions (agilité, amélioration continue des performances), nous pouvons maintenant formaliser notre objectif. Nous nous proposons, au-delà de la garantie et du maintien d'un niveau de performance, de nous focaliser sur l'objectif d'amélioration continue des performances d'un SPBS. Ainsi, la relation entre agilité et performance mentionnée sur la Figure 5 évoluera vers une amélioration continue des performances. Nous considérons qu'un SPBS agile est un système dont les performances sont améliorées continuellement, voir Figure 6. Notre motivation pour atteindre cet objectif s'appuie sur les deux arguments suivants :

- dans un contexte dominé par une forte concurrence, l'agilité est l'actuel objet de compétition entre les SPBS [Trentesaux, 02]. Cet objectif représente désormais une priorité pour laquelle tous les efforts (matériels et managériaux) doivent être consentis,
- les données relatives aux fonctionnements des SPBS sont devenues de plus en plus accessibles grâce aux développements des technologies de traitements de l'information (stockage, transmission, réception, analyse, …). Cet atout peut être utilisé judicieusement pour contribuer à l'agilité d'un SPBS et par conséquent assurer l'amélioration continue des performances de ses performances,

[9] D'ailleurs, les logiciels qui mettent en œuvre ces deux approches sont plutôt rares.

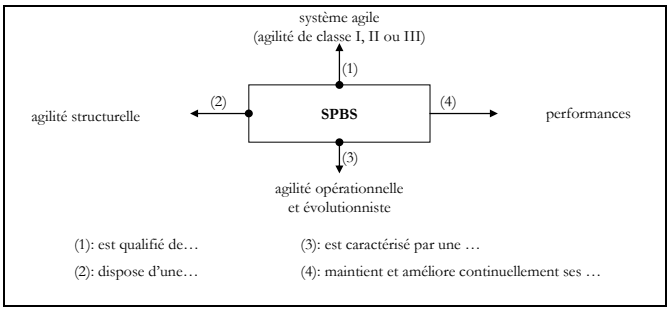


Figure 6- Relation entre agilité et amélioration continue des performances (objectif recherché)

La Figure 7 représente les relations qui existent entre les différents concepts présentés précédemment : l'agilité, grâce à ces trois composantes (agilité structurelle, opérationnelle et évolutive), constitue un moyen pour assurer l'amélioration continue des performances (objectif) d'un SPBS. Cette amélioration continue concerne les trois caractéristiques d'un système performant : pertinence, efficacité et efficience.

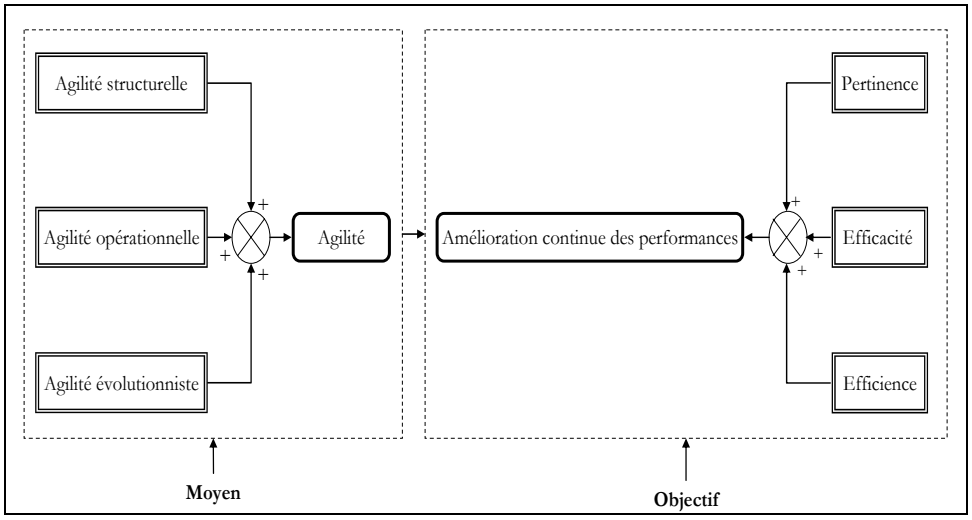


Figure 7- Relations entre les différents concepts de notre travail

En résumé :

- dans cette thèse, nous nous focalisons sur l'agilité de type <classe III, évolutionniste> (agilité des entités qui composent un SPBS). Nous avons choisi de traiter ce type d'agilité car premièrement nous nous intéressons au niveau opérationnel[10] d'un SPBS. Deuxièmement nous nous plaçons dans un cadre de SPBS complètement (ou fortement) automatisés. Dans ce cadre, l'opérateur humain tient essentiellement un rôle de surveillance : ce choix signifie que l'agilité résulte essentiellement des ressources matérielles et non de l'opérateur humain (qui ne dispose alors d'aucun ou de très peu de pouvoir décisionnel),
- notre objectif est de contribuer à l'amélioration continue des performances d'un SPBS : intuitivement, l'agilité évolutionniste constitue un moyen pour atteindre cet objectif d'amélioration continue,
- nous adoptons une approche par pilotage (moyen) pour garantir l'amélioration continue des performances d'un SPBS. Ce choix est justifié car la finalité d'un système de pilotage correspond à cet objectif. En effet, nous considérons la définition du pilotage donnée dans

[10] Les deux autres niveaux de la gestion d'un SPBS sont les niveaux stratégique et tactique [Tchako, 94].

> [Trentesaux, 02] : 'le pilotage consiste à décider dynamiquement des commandes pertinentes à donner à un système soumis à des perturbations pour atteindre un objectif donné décrit en termes de maîtrise de performances. La notion de maîtrise intègre non seulement celle de maintien d'un niveau de performance donné, mais également celle d'une évolution vers un niveau de performance souhaité ou amélioration continue'.

Cependant, la garantie de l'amélioration continue des performances d'un SPBS se heurte actuellement à un certain nombre de problématiques. Celles-ci ont pour origine, d'une part, la nature et la dynamique d'un SPBS (de plus en plus complexe) et d'autre part les différentes perturbations auxquelles un SPBS est sujet.

2. Problématiques liées à l'amélioration continue des performances

Le traitement et la résolution des problématiques qui peuvent représenter, d'une manière ou d'une autre, un frein à l'obtention de l'amélioration continue des performances sont l'objectif principal des systèmes de pilotage. C'est pourquoi nous identifions et analysons ces problématiques par rapport au processus de pilotage. Pour ce faire, nous nous sommes référés à la définition d'un processus de pilotage donnée par Simon ([Simon, 77]). Cette définition décrit globalement un processus de pilotage par la mise en relation de trois activités successives : information (ensemble de données nécessaires utilisées par les autres activités), constructions et prises de décisions et enfin évaluation a posteriori des performances générées par ces décisions. Nous reprenons cette caractérisation pour le pilotage des SPBS et nous identifions les problématiques liées à ces trois activités.

2.1. Problématiques liées aux informations

Les SPBS actuels sont caractérisés par un nombre très important de paramètres et une grande variété de données générées par le fonctionnement du système, notamment celles qui expriment ses performances. Dans un but d'amélioration continue des performances des SPBS et face à ces deux caractéristiques, les systèmes de pilotage actuels doivent résoudre les trois problèmes suivants :

2.1.1. La non diversification des données (P1)

La non diversification des données est illustrée par un comportement 'standard' du système du pilotage : c'est-à-dire l'application d'un ensemble limité de décisions et/ou de décisions identiques dans un contexte en constante évolution. Pour mieux comprendre cette problématique, nous faisons une analogie intuitive avec les techniques d'identification en automatique. L'identification est un ensemble de techniques visant à déterminer des modèles mathématiques capables de reproduire aussi fidèlement que possible le comportement d'un système physique sur la base des observations expérimentales entrées-sorties [Landau, 93]. L'identification d'un système comporte deux étapes : premièrement, sur la base d'une connaissance a priori du système (par exemple un système de 1^{er} ou $2^{ème}$ ordre) à identifier, on fixe une structure du modèle comportant des coefficients inconnus. Deuxièmement, ces coefficients du modèle sont déterminés à partir des réactions de celui-ci à des sollicitations données et connues (grâce à la variation des fréquences de ces excitations). Un problème se pose alors pour choisir les excitations qui doivent être appliquées à l'entrée du système afin d'obtenir une meilleure estimation des coefficients. En effet, des entrées peu nombreuses ou identiques ne permettent pas une bonne identification du système. Généralement, la solution consiste à exciter le processus par une Séquence Binaire Pseudo

Aléatoire (S.B.P.A qui est un signal proche d'un bruit blanc[11]) pour une bonne estimation des coefficients. Intuitivement, dans une logique d'amélioration continue des performances d'un SPBS, le même constat peut s'appliquer sur le système de pilotage : la fonction de pilotage est assimilée à l'excitation du système en entrée et la bonne estimation permanente des paramètres peut être identifiée à une amélioration continue des performances d'un SPBS.

Conséquences de P(1) sur l'amélioration continue des performances des SPBS : pour un système de pilotage, l'adoption des mêmes décisions d'une façon continue ne peut conduire qu'à un niveau identique de performances au cours du temps. Outre le fait que la non variation des situations décisionnelles ne permet pas de générer des connaissances sur le fonctionnement du SPBS, la non diversification des données peut avoir deux inconvénients majeurs :

- dans le cas où il y a évolution du contexte du SPBS, notamment le changement des caractéristiques des ressources de production (flexibilité) ou des produits (nombre ou ordre des tâches à réaliser), les commandes qui émanent du système de pilotage ne sont plus valides,
- dans le cas où il y a apparition/disparition des perturbations, une réaction (décision) identique face à des situations différentes provoquera une dégradation sensible des performances du SPBS.

2.1.2. La non pertinence des données (P2)

Les entrées d'un système de pilotage sont des données qui caractérisent et qui reflètent l'état du SPBS à piloter. Or, il est possible qu'une partie de ces données ne soient pas pertinentes. Il s'agit généralement d'un problème lié à l'exploitation/exploration d'une grande quantité d'informations (données intrinsèques, attributs liés à la dynamique du système, variables d'états, performances…). Considérons, de nouveau, l'exemple de l'identification automatique : il est fréquemment souhaitable d'attribuer des pondérations différentes aux mesures accumulées. En ce sens, des mesures de piètre qualité doivent intervenir plus légèrement dans les calculs (détermination des coefficients du modèle) que celles qui sont très précises [Longchamp, 95].

Conséquences de P(2) sur l'amélioration continue des performances des SPBS : la prise en compte, dans l'élaboration des mécanismes de pilotage, de données non pertinentes conduit nécessairement à la dégradation des performances. En effet, le décalage crée entre l'état reproduit par des données non pertinentes et l'état réel du SPBS donne lieu à une inadéquation entre les consignes fournies par le système de pilotage et celles qui doivent retournées, donc une dégradation des performances du SPBS.

Par ailleurs, un problème d'ordre 'technique' (choix des modèles de données, des architectures et moyens informatiques) peut apparaître sur le plan pratique. Ce problème a pour origine la diversité et à la multitude des données dans un processus de pilotage. En effet, dans une optique de réutilisation ultérieure, le problème de mémorisation des différentes informations disponibles se pose. Il s'agit de déterminer comment mémoriser une grande quantité de données pour être 'facilement' exploitables.

2.2. Problématiques liées aux décisions

La prise de décision représente la deuxième activité dans le processus de pilotage défini par Simon. Sur le plan pratique, la mise en œuvre de cette activité est loin d'être facile dans le cas des SPBS. Cette difficulté résulte essentiellement des trois problèmes suivants : la cohérence des

[11] Le bruit blanc est un signal dont les valeurs mesurées à des instants différents sont des variables aléatoires indépendantes les unes des autres. La densité spectrale d'un bruit blanc est constante.

décisions prises, l'impossibilité, dans certains cas, d'élaborer (en un temps raisonnable[12]) des décisions optimales et l'éventuelle évolution du système (entre l'instant de l'application d'une décision et son effet).

2.2.1. Cohérence des décisions prises (P3)

Dans des SPBS de plus en plus complexes et étendus, la question de cohérence des décisions se pose fréquemment. En effet, des conflits peuvent apparaître tels que le partage des unités de production, l'ordre de priorité pendant le traitement des différentes tâches ou encore dans le cas d'objectifs contradictoires. Dans une optique d'amélioration continue des performances, l'existence et la résolution des situations conflictuelles conduisent nécessairement au ralentissement de la dynamique du SPBS et éventuellement à l'apparition de situations de blocage. En définitif, il faudra trouver un compromis entre le besoin d'assurer en permanence l'amélioration continue des performances et la garantie d'une cohérence globale des décisions.

Conséquences de P(3) sur l'amélioration continue des performances des SPBS : l'élaboration de décisions non cohérentes (même si, considérées séparément, elles sont optimales) peut désynchroniser le fonctionnement d'un SPBS, et par conséquent entraîner la diminution de sa pertinence.

2.2.2. Élaboration en un temps raisonnable de décisions satisfaisantes ou optimales (P4)

Ce problème[13], connu sous le nom d'explosion combinatoire, est lié principalement à la grande quantité de données à prendre en compte dans un processus décisionnel (par exemple les contraintes liées à la disponibilité des ressources, l'ordre des tâches à exécuter, …). Cette caractéristique engendre sur le plan pratique des temps de calcul (temps nécessaire à l'élaboration des décisions) très longs. Dans ce cas, on est amené souvent à élaborer des décisions (solutions) satisfaisantes (dans le sens où elles respectent le maximum de contraintes) en un temps raisonnable. Des outils en recherche opérationnelle existent pour contourner ces problèmes même partiellement [Smith, 04]. Or l'obtention de solutions satisfaisantes présente aussi un inconvénient : le risque de 'minima (ou maxima) locaux', c'est-à-dire, l'obtention de solutions qui ne sont pas optimales.

Conséquences de P(4) sur l'amélioration continue des performances des SPBS : force est de constater que, dans une logique d'amélioration continue des performances et dans un contexte continuellement changeant, ce problème génère soit l'impossibilité de déterminer la bonne solution (explosion combinatoire) au bon moment, soit une déviation sensible par rapport à l'objectif recherché (minima ou maxima locaux).

2.2.3. Décalage entre l'application d'une décision et son effet (P5)

Ce problème provient du retard qui peut exister entre l'émission de la commande (l'instant où elle est appliquée) et l'observation de son effet. Par exemple, en productique, ce problème apparaît dans les systèmes de conditionnement à haute cadence [Trentesaux, 02] : il existe un retard entre le changement de la vitesse des convoyeurs (décision ou consigne) et la disparition du bourrage ou famine (effet) de la chaîne d'embouteillage.

Conséquences de P(5) sur l'amélioration continue des performances des SPBS : dans la plupart des cas, l'état d'un SPBS change pendant l'intervalle quantifié par ce retard. Par conséquent, les décisions prises ne sont plus pertinence car elles ne sont plus valables d'où une déviation par rapport à

[12] Raisonnable par rapport à la dynamique du système.
[13] Ce type de problématique n'existe que si les contraintes de temps sont fortes, notamment au niveau opérationnel, pour certains types de décisions non programmables.

l'objectif recherché au cours du temps.

2.3. Problématiques liées aux performances

La troisième classe de problématiques est relative aux performances obtenues par un SPBS. Premièrement, il y a une dégradation sensible de ces performances dans le cas où des perturbations surgissent (problème connu sous l'appellation de tolérance aux pannes). Deuxièmement, des mécanismes d'évaluation doivent être définis pour quantifier et/ou qualifier les performances obtenues. Ces deux problématiques sont présentées dans la partie suivante.

2.3.1. Stabilité et tolérance aux pannes (P6)

En automatique, un système est dit stable si toute application d'un signal d'entrée borné à ce système fournit une sortie bornée [Longchamp, 95]. Sénéchal [Sénéchal, 04] qualifie la stabilité des systèmes de production par leur aptitude à recouvrer rapidement la trajectoire économique qui leur a été assignée suite à l'occurrence des perturbations[14] ou à des variations de consignes ponctuelles. En effet, lors de son fonctionnement, un SPBS est sujet à plusieurs perturbations qui peuvent être internes (par exemple pannes des unités de production, qualité non conforme des produits, …) ou externes (rupture des stocks, commandes urgentes, …). Duffie et Prabhu [Duffie et Prabhu, 96] remarquent que la tolérance aux pannes fait partie des objectifs qui sont difficiles à atteindre lors de la conception d'un système de pilotage. En outre, ils précisent qu'aucun algorithme n'existe ni pour prévoir tous les modes de dysfonctionnement possibles d'un système très complexe (l'imprévisibilité est un facteur accroissant de l'effet des perturbations [Miyagi et Riascos, 05]) ni pour mettre en place des stratégies pour toutes les situations. La rapidité à retrouver un fonctionnement normal constitue une condition sine qua none, d'une part, pour revenir à un niveau acceptable de performances, et d'autre part, pour pouvoir les améliorer continuellement.

Conséquences de P(6) sur l'amélioration continue des performances des SPBS : en présence de perturbations, une dégradation sensible du rendement d'une ou plusieurs unités de production ou de leur arrêt total sont observés. Cette dégradation va à l'encontre d'une amélioration continue des performances car la pertinence et l'efficacité du SPBS sont considérablement réduites.

2.3.2. Évaluation des performances obtenues (P7)

Les performances d'un SPBS, présentées sous forme de données brutes et en grande quantité, posent des problèmes relatifs à leur processus d'évaluation. Ce processus d'évaluation des performances d'un SPBS est caractérisé par des difficultés dues essentiellement à la définition et à l'évolution d'une multitude d'indicateurs de performance. Une première tâche consiste à déterminer, de façon rationnelle, un ensemble d'indicateurs non nécessairement compatibles.

Conséquences de P(7) sur l'amélioration continue des performances des SPBS : en pratique, et dans une logique d'amélioration continue des performances, il est difficile de disposer de bons indicateurs (le(s)quel(s) allons-nous améliorer ?) car souvent les indicateurs de performances sont incohérents.

En résumé, nous considérons que la problématique qui existe autour de l'amélioration continue des performances d'un SPBS est une combinaison des problèmes que nous venons de présenter. Dans notre travail (troisième chapitre) nous proposons des solutions (dans le cadre d'une approche de pilotage) pour la résolution des problèmes présentés précédemment. La partie suivante est consacrée à un état de l'art des différentes contributions qui s'intéressent partiellement ou totalement à l'amélioration continue des performances des SPBS et qui traitent un ou plusieurs

[14] Nous proposons, dans le cinquième chapitre, une typologie des perturbations qui peuvent apparaître pendant le fonctionnement d'un SPBS.

de ces problèmes.

3. État de l'art dans le domaine de l'amélioration continue des performances

3.1. Cadre comparatif

Rappelons d'abord le cadre général de notre travail : celui-ci concerne un cadre d'agilité <classe III, évolutionniste>. En conséquence, seules les références qui étudient le pilotage au niveau des entités éléments de base d'un SPBS (produit ou ressource[15]) ont été considérées. Nous parlons alors de pilotage par le produit (PPP, voir [Gouyon et al., 04]) ou pilotage par la ressource (PPR). L'étude des principales contributions dans le domaine des SPBS dont l'objectif[16] est l'amélioration (continue) des performances fait émerger une double distinction : les approches à inspiration naturelle (biologiques, phéromones) et les approches dites 'techniques' (systèmes à base d'agents, hétérarchique[17] ou holoniques). Par ailleurs, des approches issues d'autres domaines seront présentées. Ces différentes approches sont détaillées dans la partie suivante.

3.2. Approches de pilotage inspirées des systèmes vivants

3.2.1. Les approches bioniques

Les systèmes de production bioniques (SPB) ont été proposés pour la première fois par Okino [Okino, 93]. Cette approche exploite les propriétés des systèmes naturels, c'est-à-dire qu'elle transpose les caractéristiques biologiques (auto-organisation, évolution, apprentissage, ou adaptation) pour concevoir un système de production [Ueda, 01]. Cette imitation concerne aussi le pilotage de ces systèmes. En effet, les entités qui composent un système de production bionique sont autonomes et auto-organisées, par analogie avec les cellules qui composent un système vivant. Généralement, c'est une architecture de pilotage totalement décentralisée qui est adoptées pour ce genre de systèmes de production.

Fondée sur une analogie entre les systèmes de production et les systèmes vivants, Vaario et al. [Vaario et al., 97] combinent les techniques de la réalité virtuelle avec des principes d'auto-organisation pour concevoir un système de pilotage intelligent dans lequel l'utilisateur peut participer à l'élaboration des décisions grâce à une interface de simulation (voir Figure 8). L'idée de base de ce système de pilotage auto-organisé est la définition d'un champ de gradient (attraction ou répulsion) pour chaque entité (transporteur ou machine). Ces champs expriment la force qui existe entre les différentes entités autonomes en fonction de la distance qui les sépare. Dans le cas d'un ordonnancement, l'allocation d'une tâche est assurée par une concordance entre les champs d'attraction de la machine et celui de la tâche. La validation est fournie à travers l'exemple d'une ligne d'assemblage de bicyclettes. Cette approche auto-organisée est intéressante car elle dote les différentes entités d'une autonomie importante qui permet une recherche continue de la machine ou du produit la (le) plus adapté(e) pour traiter une tâche (pour être traité). Néanmoins, les auteurs considèrent que toutes les entités sont capables de se déplacer (même les machines) ce qui n'est pas vrai pour la plupart des SPBS.

[15] L'intérêt porté, uniquement, à ces deux entités sera justifié dans le deuxième chapitre.
[16] Objectif déclaré clairement ou implicitement.
[17] La notion d'hétérarchie sera présentée dans le deuxième chapitre (hétérarchie = non hiérarchie).

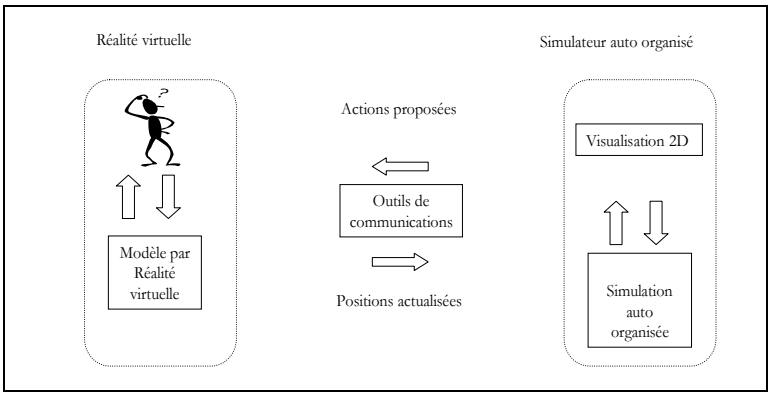


Figure 8- Approche biologique par auto-organisation [Vaario et al., 97]

Dans ce même cadre (inspiration des systèmes naturels), Brezocnik et Balic [Brezocnik et Balic, 01] proposent un modèle d'une unité d'assemblage de pièces mécaniques auto-organisée. L'auto-organisation de cette unité consiste à doter l'ensemble des pièces de caractéristiques de l'évolution génétique des systèmes vivants (brassage, croisement, mutation) afin d'obtenir des pièces assemblées (produit fini). En outre, les perturbations sont prises en compte en introduisant des pièces présentant des défauts dimensionnels. Cette approche, grâce à l'heuristique qu'elle propose (algorithmes génétiques), peut être utile pour l'ordonnancement d'un SPBS : une gamme de tâches peut être assimilée aux différentes étapes proposées par cette approche, voir Figure 9. En outre, ce travail est intéressant dans le sens où il met en œuvre les notions d'exploitation et d'exploration continues de l'espace des solutions assurées grâce à l'évolution des populations génétiques et s'intègre ainsi parfaitement dans une logique d'amélioration continue des performances. Cette approche n'inclut pas de mécanismes d'apprentissage (les mauvaises combinaisons peuvent se reproduire) et les perturbations liées au fonctionnement des machines ne sont pas prises en compte.

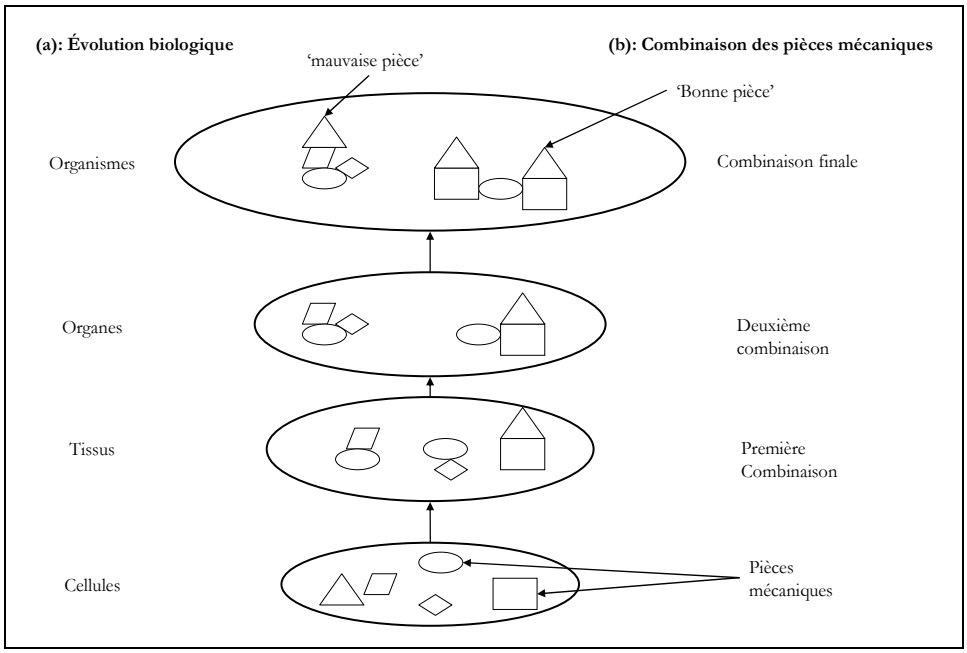


Figure 9- Imitation de l'évolution biologique pour une unité d'assemblage [Brezocnik et Balic, 01]

3.2.2. Les approches par phéromones

Les algorithmes à base de colonies de fourmis ont été introduits par Dorigo et al. ([Dorigo et

al., 91]). Cette approche s'inspire du comportement de fourmis cherchant à se nourrir. Chaque fois qu'une fourmi se déplace, elle sécrète une substance (phéromone) sur son chemin. Les autres fourmis suivront les chemins établis par leurs prédécesseurs. Le tableau Tab. 3 fournit une analogie entre l'approche par phéromones et le fonctionnement des systèmes de production :

Approche par phéromones	Analogie	Mise en œuvre
Fourmi	Observation, création et diffusion des informations	Agent ressource, agent ordre de fabrication
Phéromone	Informations, feedback, évaporation, …	Objets communicants
Environnement	Tableaux noirs distribués (embarqués)	Système de transfert (convoyeurs)

Tab. 3 Analogie entre l'approche par phéromones/système physique

L'approche proposée dans [Sallez et al., 04] utilise cette technique et se base sur deux modèles : approche orientée ressource et approche orientée produit. Dans les deux cas, une autonomie de prise de décision est donnée à chaque type d'entités afin d'aboutir à un comportement global caractérisé par un haut niveau de performances. Cette approche est intéressante dans la mesure où elle traite des situations de pannes (une simulation sur un cas réel est donnée) et met en avant l'importance d'un retour d'expérience (les trajets suivis par chaque produit). Cette propriété constitue à notre avis une approche pertinente qui a montré l'intérêt de l'apprentissage par renforcement (bon ou mauvais trajet ?) pour l'amélioration continue des performances (amélioration de la qualité des trajectoires), voir Figure 10 :

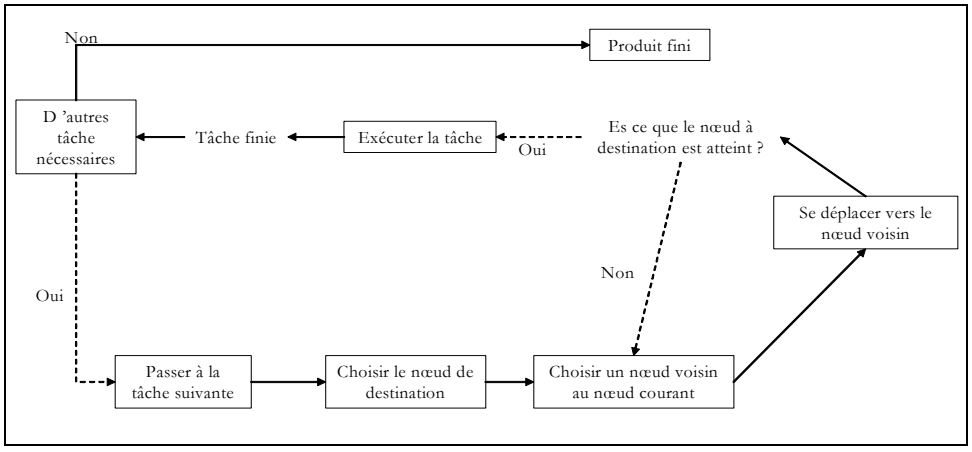


Figure 10- Modèle orienté produit par phéromone [Sallez et al., 04]

Peeters et al. [Peeters et al., 01] présentent une approche pour la reconfiguration des systèmes flexibles de production. Dans cette approche, chaque entité (identifiée à une fourmi) est totalement autonome et n'interagit pas avec les autres entités. Elle s'informe uniquement de l'état de son environnement (grâce aux phéromones) et adapte son comportement en fonction de ce qu'elle observe dans cet environnement. En effet, au fur et à mesure, un phénomène de renforcement émerge : les trajets contenant de la nourriture sont renforcés par la déposition d'autres phéromones et par conséquent ces trajets sont les plus suivis. Au contraire, pour les trajets les moins suivis, les phéromones s'évaporent ce qui rend ces trajectoires moins attirantes. Cette référence traite essentiellement le problème de la non diversification des données (variation des trajectoires) et représente les avantages suivants : simplicité de la mise en œuvre, d'implémentation et de modification du système de pilotage ou encore des mécanismes de coordination simples. Néanmoins, elle présente des inconvénients : le temps nécessaire pour trouver les bonnes trajectoires est important et la nécessité d'un ajustement des différents paramètres (nombre d'entités ou fourmis nécessaires, la vitesse d'évaporation, la pertinence des informations diffusées par les phéromones).

3.3. Approches 'techniques'

3.3.1. Les approches de pilotage à base d'agents

Plusieurs travaux ont montré l'intérêt de l'approche à base d'agents pour modéliser un système de pilotage réactif. En effet cette approche permet de réaliser un pilotage réparti et reconfigurable. Ces deux propriétés permettent de prendre en compte l'hétérogénéité des équipements (unités de production), leurs évolutions et également de mieux intégrer les décisions des opérateurs humains aux décisions des agents. La modélisation multi-agents appréhende en outre le système opérant non plus par une approche globale descendante, mais par assemblage de comportements locaux de sous-systèmes de pilotage. Dans ce sens, une population d'agents réactifs et cognitifs modélisent des comportements locaux des entités composant un système de production. Ces entités s'organisent grâce à l'interaction des comportements des agents afin de procurer un comportement global cohérent par rapport aux objectifs fixés. Les caractéristiques des agents englobent l'auto-organisation, la tolérance aux pannes, un comportement émergent (globalisation de la performance par combinaison des performances locales), autonomie, coopération, réactivité, adaptation et décentralisation.

Mařík et Lažanský [Mařík et Lažanský, 06] dressent un état de l'art des applications industrielles des systèmes multi agents : ces applications englobent l'industrie automobile ou chimique, les systèmes de manutentions et de transport (de type AGV), l'ordonnancement de la production ou encore les entreprises virtuelles.

[Aydin et Öztemel, 00] proposent un système d'ordonnancement d'atelier de production composé de deux parties : un environnement simulé du système de production et un agent intelligent. Le fonctionnement de cet agent consiste à percevoir l'état de l'environnement et de lui envoyer des règles de décision (dispatching rules). Cet agent apprend au cours du temps à l'aide d'un algorithme d'apprentissage par renforcement (grâce à un retour d'expérience de l'environnement). Outre la structure centralisée de ce système de pilotage (un seul agent, non adéquat pour des systèmes étendus), des critiques peuvent être faites à cette référence. En effet, les auteurs utilisent un nombre limité de règles de décision (3 règles : SPT, COVER et CR[18]), ce qui limite considérablement l'exploitation et l'exploration de la totalité des capacités offertes par le système. Un autre inconvénient concerne la non prise en compte des perturbations (pannes machines), ce qui représente une hypothèse non réaliste. Par contre, l'avantage principal de ce travail est de proposer des mécanismes d'apprentissage pour la détermination permanente des meilleures décisions, contribuant ainsi à l'amélioration continue des performances.

Maione et Naso [Maione et Naso, 01] proposent une approche de pilotage originale qui consiste à doter les produits et les machines de capacités décisionnelles. Tout en affirmant qu'il existe peu d'applications qui proposent une adaptation dans un environnement dynamique, les auteurs proposent un modèle de pilotage intégrant une adaptation en ligne utilisant les algorithmes évolutionnistes. Ces algorithmes sont couplés à des règles d'inférence floues pour déterminer, pour chaque action d'ordonnancement, la règle de décision la plus adéquate. Les auteurs utilisent une approche multi-agents pour mettre en œuvre leur proposition. Chaque produit ('part') et chaque machine ('workstation') se voient attribuer deux types d'agent respectivement PA (part agent) et WA (workstation agent). Nous avons relevé dans cette contribution, sur le cas d'étude traité, la prise en compte d'un ensemble de perturbations qui surviennent sur les machines. Cette adaptation vis-à-vis de ces perturbations consiste à choisir les règles (pondérations) les plus appropriées pour ordonnancer le système de production. Les règles sont évaluées constamment en fonction des performances qu'elles réalisent. Cette évaluation continue s'intègre dans le cadre d'une recherche

[18] Shortest Processing Time, C over T et Critical Ratio.

permanente des meilleures décisions qui permettent de réaliser les meilleures performances au cours du temps.

3.3.2. Les approches hétérarchiques

Saad et al. [Saad et al., 97] traitent le problème de l'ordonnancement hétérarchique des systèmes de production flexibles en utilisant une approche fondée sur le concept du Contract Net [Parunak, 85]. Les auteurs proposent une approche de modélisation basée sur des entités distribuées et localement autonomes. Le modèle proposé (appelé Production Reservation) intègre des mécanismes de négociations entre les différentes entités. La comparaison des résultats obtenus avec ceux donnés en appliquant les règles d'ordonnancement classiques (FIFO, EDD, SPT[19], …) montre l'intérêt d'une approche de pilotage hétérarchique. Par rapport à la problématique de l'amélioration continue des performances, cette approche contribue à la résolution des problèmes liés à l'explosion combinatoire (ordonnancement décentralisé) et à la complexité des SPBS (entités autonomes). Néanmoins, les hypothèses adoptées sont réductrices : pas de temps de transfert, temps opératoire identique pour la totalité des machines, la négociation entre les entités peut être pénalisante en temps surtout dans le cas de systèmes de grande taille et enfin les perturbations ne sont pas traitées.

Tchako [Tchako, 94] propose une structure de pilotage totalement distribuée et générique s'appuyant sur la notion de SIP (Station Intégrée de Pilotage). Une SIP a pour fonction d'assurer la gestion locale d'une ou plusieurs ressources ainsi que des moyens de transfert, de stockage, des produits et des outils de manutention, voir Figure 11. Chaque SIP est composée d'un système de décision responsable de la sélection de l'action à entreprendre pour piloter le système de production. En outre, un système de gestion des informations, un système d'interface (gestion des dialogues avec des opérateurs humains ou avec d'autres SIP) et un système de contrôle (exécutions des ordres émanant du système de décision) sont intégrés au niveau de chaque SIP.

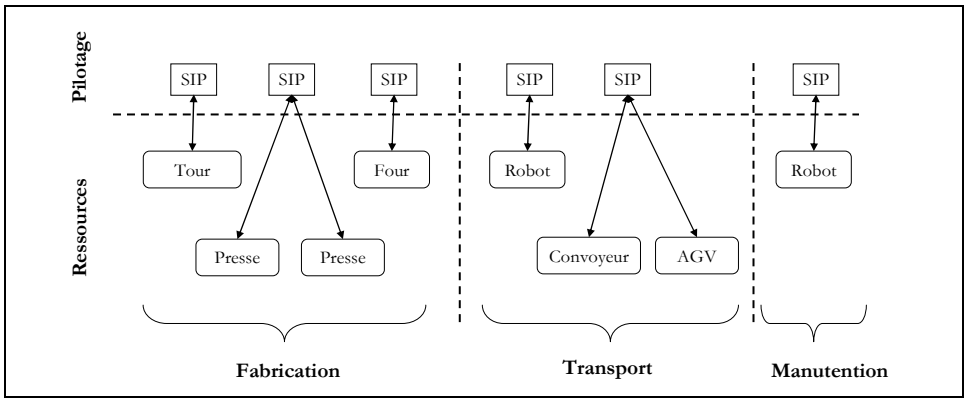


Figure 11- Pilotage hétérarchique basé sur la notion de SIP [Tchako, 94]

Certes, l'aspect amélioration continue des performances n'apparaît pas clairement dans ce modèle de pilotage (chaque SIP a pour objectif le maintien d'un niveau acceptable de performances) mais ce modèle hétérarchique présente un ensemble d'avantages. En effet, cette structure de pilotage a été appliquée sur une chaîne d'embouteillage et a donné des résultats intéressants concernant l'allocation dynamique des tâches (ordonnancement) et la tolérance aux pannes. Cependant, ce modèle peut montrer des limites : dans le cas d'un SPBS étendu (combien de SIP doit-on mettre en place ?) et dans le cas où il y a un partage de ressources (apparition de conflits et problèmes de cohérence des décisions prises).

[19] First In First Out et Earliest Due Date.

Le modèle de pilotage proposé dans [Trentesaux, 96] présente la spécificité d'intégrer l'opérateur humain en utilisant conjointement la notion de SIP et l'intelligence artificielle distribuée. Cette approche hybride est caractérisée par (voir Figure 12) :

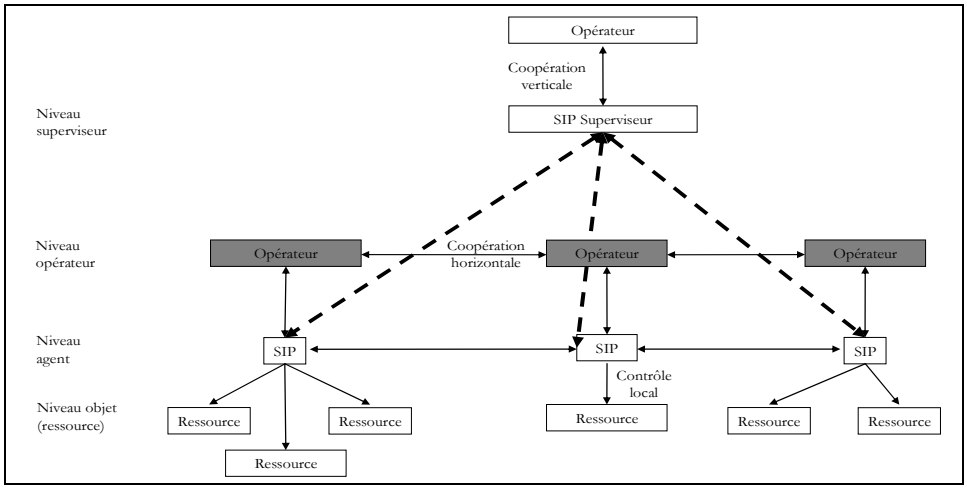


Figure 12- Intégration de l'opérateur humain dans une structure de pilotage hétérarchique [Trentesaux, 96]

- une structure de pilotage distribuée, supervisée et multicritère,
- une détection dynamique et l'ordonnancement d'une ressource goulet,
- un système interactif multicritère d'aide à la décision (SIAD) qui consiste à intégrer l'opérateur humain dans le processus décisionnel,
- une coopération verticale (entre un opérateur et une SIP),
- une coopération horizontale entre les différents opérateurs entre eux et aussi entre les SIP.

L'intégration de l'opérateur humain et des mécanismes décisionnels multicritères peut contribuer à l'amélioration continue des performances du système piloté. D'une part, l'opérateur humain permet d'assurer une meilleure prise en compte de l'adaptabilité du processus de pilotage au cours du temps (une pertinence et une cohérence accrues des décisions prises). D'autre part, cette approche montre (sur un cas industriel) que la variation des critères lors de la prise des décisions (diversification des données) apporte un compromis efficace en terme de flexibilité et de réactivité temps-réel.

3.3.3. L'approche holonique

L'application du paradigme holonique est relativement nouvelle pour le pilotage des systèmes de production. Cette approche associe les meilleures propriétés des systèmes de pilotage hiérarchiques et hétérarchiques[20] (voir par exemple PROSA[21] [Valckenaers et al., 98], MetaMorph [Maturana et al., 99] ou [Simão et al., 06]). Cette association offre une garantie des performances (grâce aux mécanismes de coordination dans les systèmes de pilotage hiérarchiques) et une grande robustesse contre les perturbations (grâce à une structure de pilotage hétérarchique) [Bongaerts et al., 00]. Morel et al. [Morel et al., 03] ont montré que cette approche constitue un moyen adéquat pour faire le lien entre la productique (entreprise intégrée) et l'automatique. En outre, l'approche holonique permet la représentation et l'intégration de l'opérateur humain dans le système de

[20] L'appellation 'holarchique' est attribuée pour exprimer un compromis entre organisations hiérarchiques et organisations distribuées [Charpentier et al., 01].
[21] Product-Resource-Order-Staff Architecture.

pilotage. Mařík et Lažanský [Mařík et Lažanský, 00] dressent les similarités et les différences entre les systèmes multi agents et les systèmes holoniques.

Dans le cadre des systèmes de production de produits personnalisés (une grande variabilité de produits) Gouyon [Gouyon, 04], en se focalisant sur les problèmes de flexibilité, de reconfiguration et de traçabilité de chaque produit, utilise une approche originale qui consiste à intégrer le produit dans la boucle de contrôle, en le dotant de fonctions décisionnelles, voir Figure 13. La contribution présentée dans cette référence porte sur la structuration et la synthèse du contrôle par le produit de la production. Les techniques de synthèse formelle de la commande, issues de la théorie du contrôle par supervision (Supervisory Control Theory, [Ramadge et Wonham, 87]) permettent dans cette approche d'obtenir automatiquement une structure de contrôle à partir de modèles de procédé et d'objectifs à atteindre. Cette contribution propose d'élaborer :

- 'hors ligne' les règles de commande des différentes ressources de production. Chaque ressource (modélisée par un holon) est capable de collaborer avec d'autres ressources dans un but global qui est la fabrication d'un produit,
- 'en ligne' des règles de contrôle personnalisé qui sont embarquées dans les produits (un produit est un holon composé d'une partie physique et d'une partie informationnelle).

Les mécanismes décisionnels proposés dans cette approche sont composés de deux processus :

- un processus de contrôle de la fabrication par le produit composé de quatre sous processus :
  - un sous processus de contrôle qui assure le routage sur les différentes ressources de production,
  - un sous processus d'activation réalisé par les ressources coordonnées par le produit,
  - un sous processus opératif correspondant au procédé de fabrication du produit,
  - un sous processus d'observation des transformations du produit réalisé par les ressources du système.

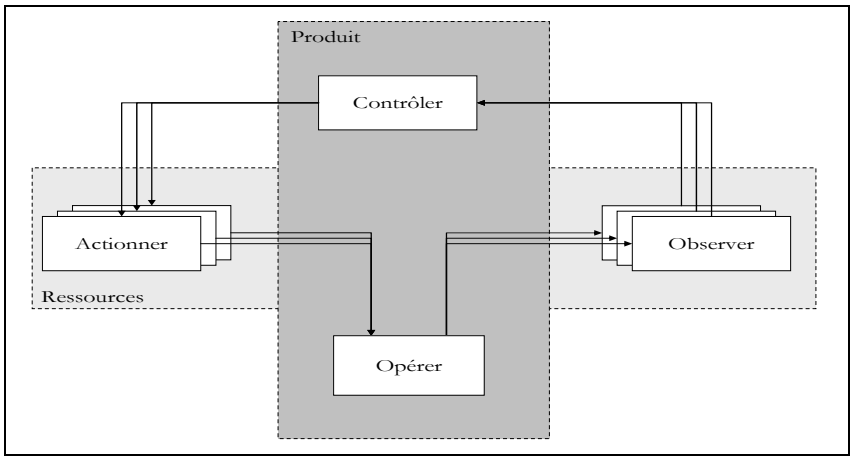


Figure 13- Processus de contrôle d'une ressource par le produit [Gouyon, 04]

- un processus de contrôle des ressources par le produit composé également de quatre sous processus :
  - un sous processus de contrôle qui reçoit et traite les requêtes émises par le produit,

- un sous processus d'activation qui transforme l'information transmise par le processus de contrôle en une action physique sur le produit,
- un sous processus d'opération qui assure une transformation morphologique, spatiale ou temporelle du produit,
- un sous processus d'observation qui interprète la transformation physique et qui la transmet au processus du contrôle du produit.

Ces travaux confirment l'orientation actuelle vers des fonctions de pilotage décentralisées au niveau des entités qui composent un SPBS. Cette orientation paraît appropriée pour mettre en œuvre une autonomie décisionnelle et intégrer des mécanismes d'apprentissage. En effet, la fonction de pilotage est distribuée dans les produits et dans les ressources. De plus, la prise de décision globale concernant l'optimisation de la productivité est assurée par l'interrogation d'un système de pilotage type MES (Manufacturing Execution System). Une telle architecture de pilotage donne la possibilité aux entités d'évoluer et d'adapter leur fonctionnement et par conséquent élaborer les décisions adéquates. L'adaptation continue des décisions en fonction du contexte constitue un moyen pour améliorer continuellement les performances d'un SPBS.

Maturana et al. [Maturana et al., 99] proposent une architecture de pilotage appelée MetaMorph. C'est une structure holonique basée sur un ensemble de médiateurs. Ces médiateurs sont des supports de prise de décision qui coordonnent les activités du système holonique. Cette coordination implique trois phases principales : l'identification des tâches à exécuter (subtasking), la création de communautés virtuelles d'agents et l'exécution des tâches. Les mécanismes d'adaptation dans cette architecture sont facilités par des changements structurels et organisationnels.

En considérant la même architecture, Shen et al. [Shen et al., 98] proposent une approche originale qui résulte de la dotation de ces médiateurs avec des capacités d'apprentissage (voir Figure 14) par une analyse de l'expérience passée (learning from history). La seconde originalité de cette architecture, la plus importante à notre avis, réside dans la projection temporelle des comportements des différents médiateurs (prédiction) à l'aide de la simulation (learning from future).

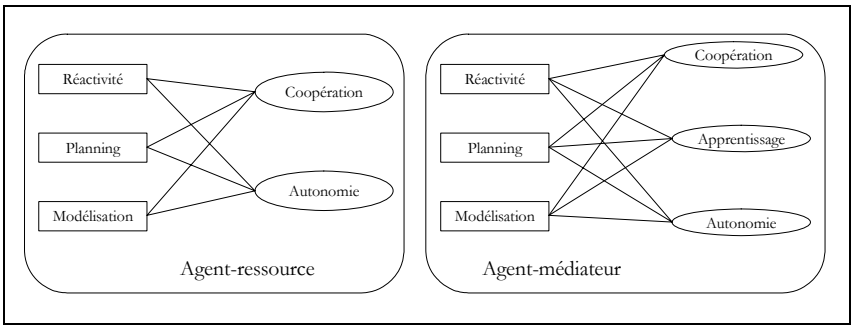


Figure 14- Les caractéristiques de base de l'architecture MetaMorph [Maturana et al., 99]

Ainsi :

- le premier type d'apprentissage est de type apprentissage à base de cas (Case Based Learning), il est réservé principalement à l'affectation des différentes tâches sur les différentes ressources,
- le deuxième type d'apprentissage est assuré grâce à une simulation du comportement du système sur un futur proche, ce type d'apprentissage est réservé au traitement des fonctionnements perturbés. Cette simulation utilise un modèle virtuel du système pour prévoir

le comportement des différents médiateurs.

Les critiques qui peuvent être faites à cette approche concernent d'une part, l'approche centralisée (médiateurs) qui peut présenter des limites avec l'augmentation des tâches et/ou des ressources. D'autre part, la projection dans le futur est une approche peu réaliste compte tenu des hypothèses émises et de l'évolution du système pendant la période de simulation. Néanmoins, cette référence met en avant le rôle important de l'apprentissage dans le processus décisionnel à travers les possibilités qu'il offre par exemple pour traiter le problème de la tolérance aux pannes. En effet, des capacités d'apprentissage permettent au système de pilotage d'acquérir des nouvelles connaissances, ce qui contribue à l'élaboration des décisions plus pertinentes. Cette propriété d'évolutivité du système de pilotage s'inscrit dans une logique d'amélioration continue des performances.

3.4. Approches issues d'autres domaines

3.4.1. L'identification automatique

McFarlane et al. ([McFarlane et al., 02]) combinent l'identification automatique avec l'approche multi agents pour concevoir des produits intelligents (le concept de l'IMS, Intelligent Manufacturing Systems est présenté dans [Zaremba et Morel 03]) et par conséquent mettre en œuvre un système de pilotage décentralisé et intelligent. L'identification automatique est une extension logique des codes à barres et cette nouvelle technique (Electronic Product Code (EPC)) consiste à associer à chaque produit une étiquette scannée par un lecteur (RFID).

Selon les auteurs, l'avantage de l'emploi de cette technique est l'amélioration du suivi des produits et des stocks. L'identification automatique est la liaison entre le produit et son agent correspondant. Dans ce sens, un produit intelligent :

- possède une identité unique,
- est capable de communiquer avec son environnement,
- peut stocker ses propres données,
- utilise un langage pour exprimer ses besoins ou ses caractéristiques,
- est capable de participer au processus de pilotage (voir Figure 15-b).

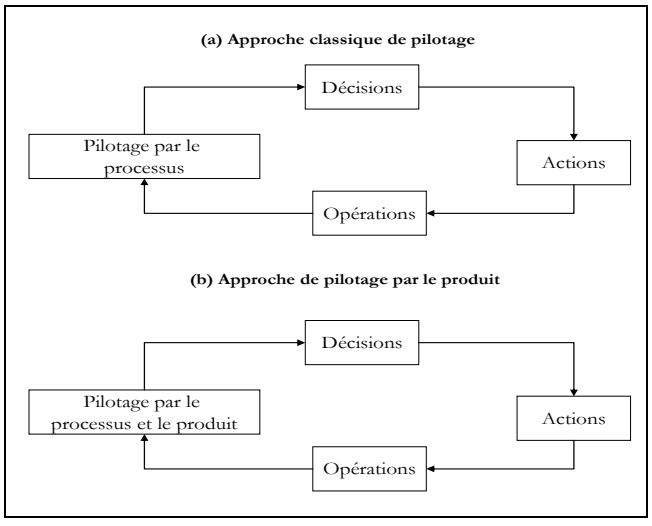


Figure 15- Identification et pilotage par le produit [McFarlane et al., 02]

L'utilisation des technologies récentes (RFID, voir [Wong et McFarlane, 06]) est proposée pour élaborer un système de pilotage par le produit. En outre, sa mise en œuvre permet l'auto-organisation (autonomie) et des capacités d'apprentissage (évolutivité) dans le processus décisionnel de chaque produit. Or ces deux aspects ne sont pas traités dans la référence citée bien qu'ils représentent à notre avis une excellente alternative pour contribuer à l'amélioration continue des performances d'un SPBS.

Nous avons présenté, à ce stade du chapitre, un état de l'art s'appuyant sur des références issues de la productique. Nous avons jugé essentiel d'étendre cette étude sur d'autres domaines. Dans ce sens, la robotique mobile représente un domaine dans lequel l'aspect 'amélioration continue des performances' est omniprésent (amélioration continue des connaissances d'un robot). Dans la partie suivante nous présentons les apports qu'offre la robotique mobile transposables et/ou source d'inspiration pour le cas des systèmes de production de biens et de services.

3.4.2. La robotique mobile

Dans le domaine de la robotique, citons les travaux de chercheurs pionniers tels que Brooks ou Kaelbling (voir [Brooks, 91] ou [Smart et Kaelbling, 02]) dans le domaine de la robotique mobile qui traitent l'apport de l'apprentissage sur les performances d'un robot. D'une part, un robot est doté de mécanismes d'apprentissage (généralement par renforcement, c'est-à-dire à l'aide d'un système de récompense/pénalité) pour exécuter ses tâches. D'autre part, un robot peut être vu comme une entité autonome (dans le sens où il n'y a pas de commande centrale) et cherchant à améliorer ses performances en apprenant de nouveaux chemins ou de nouvelles connaissances (éviter les obstacles). Ces travaux peuvent constituer une base intéressante extensible aux SPBS en donnant aux systèmes de pilotage plus d'autonomie (prise de décision locale) et d'évolution (apprentissage).

3.4.3. Le projet Swarm-bots

Il s'agit d'un projet financé par le programme Technologies Futures et Emergentes de la communauté européenne ([Swarm, 05]) qui vise à mettre en œuvre des nouvelles approches pour la conception et l'implémentation d'artéfacts (objets) auto-organisés. Un 'swarm-bot' est une agrégation (ou collection dont le nombre peut atteindre 100) de petits robots mobiles capables de s'auto-assembler en se connectant/déconnectant l'un de l'autre. Cette auto-organisation leur permet d'explorer, de naviguer et de transporter des objets sur des terrains difficiles. En outre, leurs mouvements donne lieu à un comportement global et collectif leur permettant par exemple de passer une fosse (améliorer une performance globale) ou d'éviter des obstacles (perturbations). Pour ce type d'applications, le système de pilotage est axé autour de deux idées complémentaires :

- la conception d'un système de contrôle qui imite les systèmes biologiques (les insectes sociaux),
- la dotation de ces robots de capacités d'apprentissage. En effet, chaque swarm-bot apprend de ses actions et des actions des autres robots.

Les applications possibles d'une telle approche peuvent être à titre d'exemple l'exploration de l'espace (planète Mars) ou les fonds marins (installation/réparation de câblage de télécommunication, pipeline, …). Dans ce cas, les swarm-bots opèrent dans un 'milieu hostile' et possèdent peu de connaissances au début de leurs fonctionnements. McLurkin [McLurkin, 04] affirme que le 'swarm-bot' représente la tendance future des recherches en robotique et devra aider l'humanité à accomplir des tâches délicates (sauvetage pendant les périodes des catastrophes naturelles).

McLurkin et Smith [McLurkin et Smith, 04] proposent une application utilisant 56 robots qui

explorent un environnement fermé (zone de dimension 6 m x 6 m comportant plusieurs couloirs et pièces). Le but de cette application est de disperser efficacement ces robots dans cet environnement ainsi que de leur permettre de charger périodiquement leur batterie en se connectant à des bornes d'alimentation. Les robots communiquent entre eux par transmission infra rouge, des signaux lumineux et des messages sonores. La trajectoire et les mouvements de chaque robot ont pour origine la détection de la présence des autres robots (système de voisinage). Ainsi, cette intelligence collective permet de disperser tous les robots et l'évitement de collisions. Une autre application (dispersion de 108 robots afin de trouver un objet au sein d'une base militaire) est également présentée.

Cette approche montre l'intérêt d'une décentralisation totale des capacités décisionnelles au niveau des différentes entités. En effet, cette décentralisation permet à la fois des capacités d'auto-organisation (autonomie et prise de décision locale) et d'apprentissage (évolution des connaissances des swarm-bots). L'apport essentiel de ce travail est la confirmation (grâce à des simulations) que même en ayant un très grand nombre d'entités autonomes, un comportement cohérent et performant peut émerger. Nous sommes persuadés que cette architecture de pilotage peut être transposable dans le cas d'un SPBS pour assurer l'amélioration continue de ses performances.

Le tableau Tab. 4 récapitule et synthétise l'apport des différentes approches qui contribuent (totalement ou partiellement) ou qui peuvent représenter une source d'inspiration pour assurer l'amélioration continue des performances des SPBS. Sur ce même tableau, figure(nt) la ou les problématique(s) qu'elles traite(nt) :

Approches de pilotage	Nature de l'approche	Référence(s)	Type de pilotage	Concept(s) pouvant présenter un apport pour l'amélioration continue des performances	Outils/méthodes de mise en œuvre	Problématique(s) traitée(s)
à inspiration biologique	Les systèmes bioniques	[Brezocnik et Balic, 01]	PPP	Autonomie	Algorithmes génétiques/simulation	Perturbations, la non diversification des données
		[Vaario et al., 97]	PPP/PPR	Autonomie	Simulation (réalité virtuelle)	Cohérence des décisions
		[Beslon, 95]	PPR	Apprentissage	Réseaux de neurones	La non diversification des données
	Les systèmes par phéromones	[Sallez et al., 04]	PPP/PPR	Autonomie + apprentissage	Système par phéromones/Simulation	Perturbations
		[Parunak et Brueckner, 01]	PPP	Autonomie	Système par phéromones	Cohérence des décisions
'techniques'	Les systèmes multi agents	[Aydin et Öztemel, 00]	PPR	Apprentissage	SMA/Simulation	Évaluation des performances
	Les systèmes holoniques	[Shen et al., 98], [Maturana et al., 99]	PPR	Apprentissage	Système holonique/Simulation	Perturbations
		[Gouyon, 04], [Simão et al., 06], [Baïna et Morel, 06]	PPP	Autonomie	Système holonique/Simulation	Cohérence des décisions
	Les approches hétérarchiques	[Saad et al., 97]	PPR	Autonomie	SMA	Perturbations
		[Tchako, 94], [Trentesaux, 96]	PPR	Autonomie	Système multi agents	Perturbations, cohérence des décisions
Autres domaines	Identification automatique	[McFarlane et al., 02]	PPP	Autonomie	Système holonique/multi agents	La non diversification des données, cohérence des décisions
	Swarm-bots	[McLurkin et Smith, 04]	PPP	Autonomie + apprentissage	Swarm-bots	
	Robotique	[Brooks, 91], [Smart et Kaelbling, 02]	PPP	Autonomie + apprentissage	Robots	

Tab. 4 Tableau récapitulatif des différentes approches pour l'amélioration continue des performances

3.5. Analyses

L'analyse du Tab. 4 ci-dessus nous permet de constater que la plupart des contributions présentées dans l'état de l'art et qui peuvent contribuer à l'amélioration continue des performances proposent une structure hétérarchique de pilotage. Cette orientation est doublement justifiée par les possibilités offertes par les progrès technologiques actuels et l'échec des approches de pilotage classiques à s'adapter à l'évolution du contexte industriel. Cependant, nous remarquons que :

- la généricité des approches proposées est limitée. En effet, il s'agit souvent de proposer une contribution pour un type donné de système de production avec généralement une mise en œuvre grâce aux systèmes multi agents ou holoniques,
- les travaux utilisant simultanément l'autonomie et l'apprentissage dans la prise de décision restent limités. Ces deux propriétés peuvent être complémentaires dans le but d'une amélioration continue des performances,
- les problèmes de stabilité et de retard, en raison de leurs difficultés théoriques et expérimentales, ne sont pas beaucoup étudiés,
- l'aspect évolutif du système de pilotage est peu étudié. Nous nous proposons dans ce travail d'étudier cette caractéristique nécessaire pour l'amélioration continue des performances,
- il existe peu de travaux qui proposent conjointement le pilotage par le produit et le pilotage par les ressources.

Les observations précédentes nous ont conduit à mener notre travail de recherche sur la conception d'un système de pilotage de SPBS, hétérarchique, générique et fondé sur l'autonomie et l'apprentissage. Ces propriétés *doivent* procurer au système de pilotage les capacités d'évolution nécessaires à l'amélioration continue des performances.

4. Conclusion

Dans ce premier chapitre nous avons tout d'abord présenté le contexte actuel dans lequel sont plongés les SPBS, puis nous avons exprimé les nouveaux besoins résultant des caractéristiques de l'évolution permanente de ce contexte (besoins exprimés en terme d'amélioration continue des performances). L'agilité des SPBS constitue un élément central de la réponse à cette évolution. Dans ce cadre, nous avons proposé :

- une classification de l'agilité selon la nature du SPBS étudié (agilité de classe I, II et III),
- une typologie de l'agilité en relation avec le fonctionnement du SPBS (agilité structurelle, opérationnelle ou évolutionniste).

Nous avons ensuite dressé l'état de l'art des contributions (systèmes de pilotage) relatives à l'amélioration continue des performances. Cet état de l'art, a montré les insuffisances des approches de pilotage actuelles où des problèmes restent peu traités (la non diversification des données, la tolérance aux pannes,…).

Le chapitre suivant présente les spécifications d'un système de pilotage hétérarchique répondant aux exigences induites par la recherche de l'amélioration permanente de la performance d'un SPBS.

CHAPITRE II : SPECIFICATIONS D'UN SYSTEME DE PILOTAGE HETERARCHIQUE POUR L'AMELIORATION CONTINUE DES PERFORMANCES

Le but de ce chapitre est de spécifier un système de pilotage hétérarchique évolutif pour garantir l'amélioration continue des performances des SPBS.

Deux parties principales seront mises en avant dans ce chapitre : la première partie donnera les spécifications par rapport au système opérationnel (SPBS) tandis que la deuxième partie s'intéressera à celles relatives au système de pilotage. Dans ce sens, nous proposons tout d'abord un modèle systémique générique de SPBS correspondant à l'agilité de classe III. Ce cadre de modélisation est fondé sur le concept d'entité. La seconde partie du chapitre est consacrée à la spécification d'un système de pilotage hétérarchique capable d'assurer l'amélioration continue des performances d'un SPBS. Ce système de pilotage résulte de l'intégration d'une autonomie décisionnelle et des capacités d'apprentissage au niveau des entités.

1. Notations

Les différentes notations utilisées dans le second chapitre sont données dans les tableaux ci-dessous :

- Notations relatives aux entités fixes de traitements :

EFT :	Ensemble des entités fixes de traitements disponibles dans un SPBS.
eft(k) :	Entité fixe de traitement, identifiée par son indice k.
efs(k) :	Entité fixe de stockage, identifiée par son indice k[22]. Chaque entité fixe de stockage est composée de plusieurs cases, et le nombre de ces cases déterminera si l'efs possède une capacité infinie ou finie.
NbR :	Nombre total d'entités fixes de traitements qui composent un SPBS.
to(k,m) :	Temps opératoire nécessaire pour exécuter la tâche m sur eft(k).
tt(k,k') :	Temps nécessaire pour transférer une emp entre deux entités fixes de traitement k et k'.
ta(i,m,k) :	Temps d'attente que passe une emp(i) sur efs(k) avant de passer sur eft(k) pour réaliser sa tâche m.
CEft(k,h) :	Cycle de décisions d'indice h réalisé par une entité fixe de traitement eft(k).
NbD :	Nombre total de décisions prises dans chaque cycle CEft(k,h).
Cf(x) :	Critère (d'indice x) utilisé pendant le processus décisionnel d'une eft.
NCf :	Nombre total de critères dont dispose une eft pour élaborer son processus décisionnel.
ECF :	Ensemble des critères disponibles pour les eft, ECF= {Cf(x), x∈ [1..NCf]}.
NSf :	Nombre total de stratégies dont dispose une eft pour élaborer ses décisions.
ESF(k) :	Ensemble des stratégies relatives à une eft(k).
SEft(d) :	Stratégie d'indice d appartenant à ESF(k), d∈ [1..NSf].
SEft(d,n) :	Stratégie d'indice d adoptée pour prendre la décision d'indice n, d∈ [1..NSf].
EEft(n) :	Une étape d'indice n.
EEft(i,n) :	L'emp(i) sélectionnée pendant l'étape n.
EMP(k) :	Ensemble des entités mobiles produits présentes dans une efs(k).
EFT(m) :	Ensemble des eft candidates, c'est-à-dire les eft capables d'exécuter la tâche m, EFT(m)={eft(k)∈EFT/ to(k,m)≠0}.
t_f :	Instant de prise de la première décision pour réaliser un nouveau cycle de décisions relatif à une eft(k).
NbC(k,t_f) :	Nombre total de cycles de décisions réalisés par une eft(k) à l'instant t_f.

[22] Nous gardons le même indice k car nous supposons que chaque entité fixe de traitement est couplée à une entité fixe de stockage.

λ_f : Facteur d'oubli adopté pendant le processus décisionnel d'une eft(k).

- Notations relatives aux entités mobiles produits :

emp(i) : Entité mobile produit, identifiée par son indice i.

emt(j) : Entité mobile de transfert, identifiée par son indice j.

NbP : Nombre total d'entités mobiles produits qui sont traitées dans un SPBS.

NbT : Nombre total de tâches que doit réaliser chaque emp.

t(m) : Tâche à réaliser d'indice m, m∈ [1..NbT].

Cm(y) : Critère (d'indice y) utilisé pendant le processus décisionnel d'une emp.

NCm : Nombre total de critères dont dispose une emp pour élaborer son processus décisionnel.

ECM : Ensemble des critères disponibles pour les emp, ECM= {Cm(y), y∈ [1..NCm]}.

NSm : Nombre total de stratégies dont dispose une emp pour élaborer ses décisions.

ESM : Ensemble des stratégies disponibles pour l'ensemble des emp.

SEmp(p) : Stratégie d'indice p appartenant à ESM, p∈ [1..NSm].

SEmp(p,m) : Stratégie d'indice p adoptée pour exécuter la tâche m, p∈ [1..NSm] et m∈ [1..NbT].

EEmp(k) : Une étape d'indice k (c'est par définition eft(k)).

EEmp(k,m) : L'étape sur laquelle emp(i) a exécuté la tâche m.

PEmp(i,m) : Pour une emp(i), c'est la performance intermédiaire obtenue suite à l'exécution de la tâche m.

PEmp(i) : Pour une emp(i), c'est la performance globale obtenue suite à l'exécution de la totalité des tâches($PEmp(i) = \sum_{m=1}^{NbT} PEmp(i,m)$).

t_m : Instant de prise de la première décision d'une emp, c'est par définition la date d'entrée d'une emp dans le SPBS.

$NbM(i,t_m)$: Pour une emp d'indice i, c'est le nombre des emp ayant quitté le SPBS à l'instant t_m.

A(EA) : Processus d'apprentissage d'une entité active : A(EA)=<<source,état>,objet,moment>.

λ_m : Facteur d'oubli adopté pendant le processus décisionnel d'une emp(i).

2. Pilotage hétérarchique des systèmes de production de biens et de services

2.1. Modélisation systémique d'un SPBS

La conception d'une structure de pilotage nécessite une description complète de l'organisation du système à piloter. Il faut disposer d'un modèle représentatif du système en question. Dans le domaine des systèmes de production de biens et de services, nous constatons que l'approche systémique [Le Moigne, 94] représente une base de modélisation adoptée par plusieurs auteurs, voir par exemple [Charpentier et al., 01], [Trentesaux et Tahon, 02], [Le Quéré, 04].

Premièrement, la modélisation systémique décompose l'environnement d'un système en général selon le modèle canonique suivant :

- le système d'informations **SI**,
- le système opérant **SO**,
- le système de pilotage **SP**.

Deuxièmement, l'approche systémique caractérise un système donné par le triplet (structure, activité, évolution). La structure d'un système définit comment il est organisé. Son activité exprime ce qu'il fait et enfin l'évolution décrit les changements observés (structurels, organisationnels ou comportementaux) du système ainsi que sa réponse face à ces changements.

Dans ce type de modélisation, le système de pilotage a pour but de contrôler un flux matière qui circule dans le système opérant en se basant sur le système d'information. Or, plusieurs travaux de recherche ont traité de la possibilité d'intégrer de l'information dans les entités qui composent un SPBS (voir [Brun-Picard et al., 93][23], [Brun-Picard et Sousa, 99], [McFarlane et al., 02], [Gouyon, 04]). Ce type de systèmes de pilotage localisés au niveau des entités s'intègre dans le cadre général du pilotage décentralisé. Par ailleurs, plusieurs auteurs ([Dilts et al., 91], [Duffie et Prabhu, 96], [Cantamessa, 97], [Maione et Naso, 01]) ont montré l'intérêt d'une structure de pilotage décentralisée pour accroître :

- la réactivité en réduisant les temps de circulation des informations (échange verticaux) et en prenant les décisions au plus près des endroits où le besoin s'en fait sentir [Pujo et al., 99],
- l'évolutivité et la réduction des coûts de modélisation/conception sur le long terme grâce à la modularité d'un système de pilotage décentralisé,
- la fiabilité et la tolérance aux pannes grâce à une meilleure maintenabilité [Zwingelstein, 95],
- l'adéquation avec les nouvelles structures de production de type entreprise réseau, entreprise virtuelle ou entreprise étendue.

Ainsi, les trois premiers points cités précédemment montrent l'apport potentiel d'une structure de pilotage décentralisée dans une optique d'amélioration continue des performances. C'est sur ce type de structure de pilotage que nous nous orientons notre travail. Une présentation

[23] Nous remarquons qu'il y a eu une sorte de 'creux' à partir de cette date jusqu'au début des années 2000. Une raison possible est, à notre avis, l'insuffisance du développement des moyens technologiques (communication, réseaux) pour mettre en œuvre de telles approches. Ces dernières années, cette approche de pilotage a eu un regain d'intérêt de la part des scientifiques.

des systèmes de pilotage décentralisés est donnée dans la partie suivante.

2.2. Systèmes de pilotage décentralisés

Une synthèse structurée et pertinente de la notion de décentralisation de la fonction pilotage qui fait référence à l'agencement des entités a été donnée dans [Trentesaux, 02]. En effet, l'auteur distingue deux mécanismes d'agencement complémentaires en se basant sur les théories des systèmes et de leurs organisations ([Meinadier, 98], [Mesarovic et al., 80], [Mintzberg, 82]) :

- un mécanisme d'agencement vertical qui fait référence à la notion de hiérarchie. Une hiérarchie correspond à un agencement vertical d'entités composant un système global. Dans un agencement hiérarchique d'entités la priorité est accordée, à un niveau i de la hiérarchie, à l'action ou au droit d'intervention sur une entité de niveau i+1 et où la performance du niveau i+1 conditionne celle du niveau i [Mesarovic et al., 80],
- un mécanisme d'agencement horizontal qui fait référence à l'absence totale de toute hiérarchie entre entités : il s'agit de structure 'hétérarchique' [McCulloch, 45], [Duffie et Prabhu, 96].

Selon le degré de hiérarchisation de l'agencement considéré par rapport au degré d'hétérarchisation, trois type d'agencements peuvent être distingués : les agencements purement verticaux, les agencements mixtes (horizontaux et verticaux) et les agencements purement horizontaux. Le point le plus important dans cette synthèse, à notre avis, c'est que l'auteur considère que le type d'agencement des entités permet de différencier pilotage hiérarchique et pilotage hétérarchique en distinguant trois classes de systèmes de pilotage décentralisés :

- classe I : elle comporte les systèmes de pilotage décentralisés dont les agencements sont purement verticaux. Ils sont qualifiés de purement hiérarchiques (ou hiérarchiques au sens strict). Ce type de système de pilotage présentes des avantages tels que la maîtrise des informations, l'absence de conflits entre acteurs hiérarchiquement dépendants, la faible diversité des interactions possibles, la garantie de performance, …,
- classe II : elle comporte les systèmes de pilotage décentralisés présentant, à la fois, des agencements hiérarchiques et hétérarchiques. Ils sont qualifiés de hiérarchiques ou hétérarchiques au sens large (multi-niveaux),
- classe III : elle comporte les systèmes de pilotage dont les agencements sont purement horizontaux. Ils sont qualifiés de purement hétérarchiques ou hétérarchiques au sens strict.

Les constats donnés à la fin du premier chapitre nous ont conduit à nous focaliser sur une structure de pilotage de classe III. Nous estimons que cette structure est plus à même pour intégrer des capacités d'autonomie et d'apprentissage. Par ailleurs, la décentralisation de la fonction pilotage au niveau des entités (physiques ou abstraites) a fait l'objet de plusieurs travaux de recherches :

Chi et Turban [Chi et Turban, 95] identifient deux types d'entité, l'un lié au processus de pilotage (surveillance des variables d'états, déclenchement des besoins décisionnels, prise de décision, gestion des variables d'action) et l'autre lié à la connaissance (génération de connaissances, gestion des connaissances générées et utilisation des connaissances).

Cantamessa [Cantamessa, 97] distingue d'une part des entités 'physiques' auxquelles on peut associer une fonction similaire à celle d'un objet passif ou actif du système opérant (ressource, produit, acteur) et d'autre part des entités 'virtuelles' dont la fonction ne peut être rattachée à aucun de ces objets (par exemple, gestion des connaissances, suivi des informations, etc.).

Tranvouez et al [Tranvouez et al., 99] proposent également deux types d'entité : l'un basé sur les fonctions de gestion de données, variables et contraintes associées au problème de pilotage, et le second basé sur la nature des objets composant le système de production : produits, ressources, ordres de fabrication, acteurs, etc.

Dans le cadre de l'ordonnancement décentralisé réactif, Shen et Norrie [Shen et Norrie, 99] identifient deux types, le premier lié aux savoir-faire en ordonnancement (méthode de placement, technique de recherche incrémentale, de backtracking, etc.) et le second lié aux savoir-faire ayant trait au pilotage local des ressources du système opérant (machines, cellules, etc.).

Trentesaux [Trentesaux, 02] propose deux types d'entités (relative à la complétude du processus de pilotage) :

- des entités qui mettent en œuvre un ou plusieurs processus complet de pilotage ou 'entités de pilotage' (EDP). Pour une telle entité, il est possible d'identifier un système opérant (ses variables d'entrées-commande et sorties-observation), une finalité de pilotage et par conséquent, une mesure de performance. Le système opérant est typiquement une ressource de production, mais il peut également être un service d'une entreprise, une entreprise voire un système mettant en œuvre une fonction de production. Dans la majorité des cas, la structure du système conçu reproduit celle des ressources physiques composant le système opérant. Elle aboutit à une localisation spatiale fixe des entités,
- des entités qui mettent en œuvre des processus qui ne sont pas des processus complets de pilotage ou 'entités pour le pilotage' (EPP). Un ensemble non réduit à un singleton de ces entités est requis pour mettre en œuvre un processus de pilotage complet. Chacune des entités contribue à une ou plusieurs activités d'un processus de pilotage; ces entités mettent ainsi en œuvre des fonctions nécessairement différentes. Il n'est pas possible d'identifier, pour ces entités, un système opérant. Par exemple, une entité pour le pilotage peut être dédiée à la résolution de conflits, à la gestion de connaissance ou à la surveillance d'un ensemble de données issues de capteurs, au suivi d'un ordre de fabrication ou au suivi de la réalisation d'un produit. Baker qualifie les systèmes obtenus par combinaison d'EPP d'architecture fonctionnelle [Baker, 98].

Le dénominateur commun des différentes références citées précédemment est l'association des capacités de prise de décisions aux entités qui composent un SPBS. Cette association (partielle ou totale) s'intègre dans un cadre de pilotage décentralisé.

La double distinction, proposée dans [Trentesaux, 02], de l'agencement et de la typologie des entités (EPP et EPP) permet de positionner les différentes contributions relatives à l'intégration de la fonction pilotage au niveau des entités. Néanmoins, cette typologie ne considère pas le cas d'une entité mixte (EDP-EPP). En effet, une entité peut, en même temps, mettre en œuvre un processus de pilotage (prise de décision) et intégrer ses propres connaissances (informations, mécanismes de résolution de conflits, suivi,…). Nous considérons que, dans une logique d'amélioration continue des performances d'un SPBS, une entité 'globale' de pilotage doit être définie. La globalité signifie qu'une entité englobe à la fois un processus de pilotage et les données nécessaires à la mise en œuvre d'une fonction de pilotage locale. La définition d'une entité globale de pilotage s'intègre dans une structure de pilotage hétérarchique (classe III) : c'est notre choix relatif au moyen capable d'assurer l'amélioration continue des performances d'un SPBS.

Afin de concevoir ce type unifié d'entités, nous proposons de modifier le modèle systémique classique d'un SPBS (voir Figure 16-a) vers une forme intégrée de l'environnement d'un SPBS (voir Figure 16-b) dans laquelle :

- le système de pilotage est inclus dans le système opérant,
- il existe une intersection, d'une part, entre le système d'information et le système de pilotage, et d'autre part entre le système d'information et le système opérant.

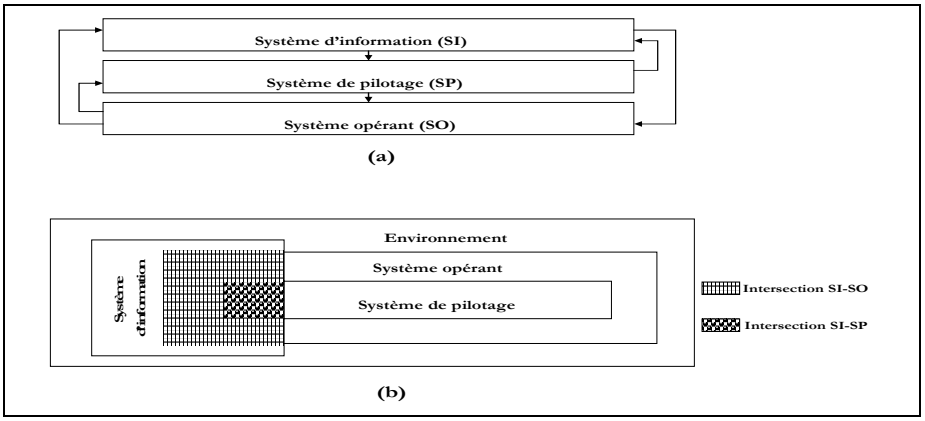


Figure 16- Modélisation systémique (classique et proposée) d'un SPBS

Cette modélisation d'un SPBS, fondée sur la notion d'entités, permet de répondre à la question suivante : de quoi est composé un SPBS ? Notre objectif est alors de proposer un modèle générique capable de modéliser des SPBS appartenant à des secteurs industriels divers.

2.3. Modèle entité d'un SPBS

Notons tout d'abord que l'activité principale d'un SPBS est d'assurer, moyennant un ensemble de traitements, la transformation d'une entité (physique ou abstraite), visant à lui apporter une valeur ajoutée. Chaque SPBS est caractérisé par un type qui détermine les fonctionnalités des entités qui le composent. Par exemple la productique (ateliers flexibles, chaînes logistiques), l'informatique (réseaux de télécommunication, systèmes embarqués, …) le transport (terrestre, aérien, …) peuvent être considérés comme des types de SPBS. Plus précisément, nous spécifions l'ensemble des entités composant un SPBS par la double distinction suivante : une distinction structurelle relative à l'emplacement physique des différentes entités dans le SPBS et une distinction fonctionnelle qui définit le rôle de chaque entité dans le SPBS.

- distinction structurelle : nous différencions les entités fixes des entités mobiles. Une entité mobile change d'emplacement physique *au moins une fois* au cours son cycle de vie; une entité fixe conserve *indéfiniment* sa position spatiale initiale et ceci durant le fonctionnement du SPBS,
- distinction fonctionnelle, nous distinguons deux familles : entités qui contribuent aux traitements et celles qui en font l'objet. Dans la première famille, nous trouvons :
  - les entités fixes de traitement d'indice k (notées eft(k), par exemple des machines),
  - les entités fixes de stockage[24] d'indice k (notées efs(k), par exemples des zones de stockage),
  - les entités mobiles de transfert d'indice j (notées emt(j), par exemple des chariots AGV).

La deuxième famille contient les entités mobiles à traiter (d'indice i et notées emp(i)[25], par

[24] Nous considérons qu'une entité fixe de stockage peut apporter une valeur ajoutée (par exemple pour la synchronisation de flux).
[25] Nous utilisons la notation emp pour la distinguer des entités mobiles de transfert (emt).

exemple des pièces à usiner). Dans la suite nous utiliserons le terme 'entité mobile produit'. La Figure 17 illustre les quatre types d'entités proposés dans cette modélisation ainsi que leurs interactions.

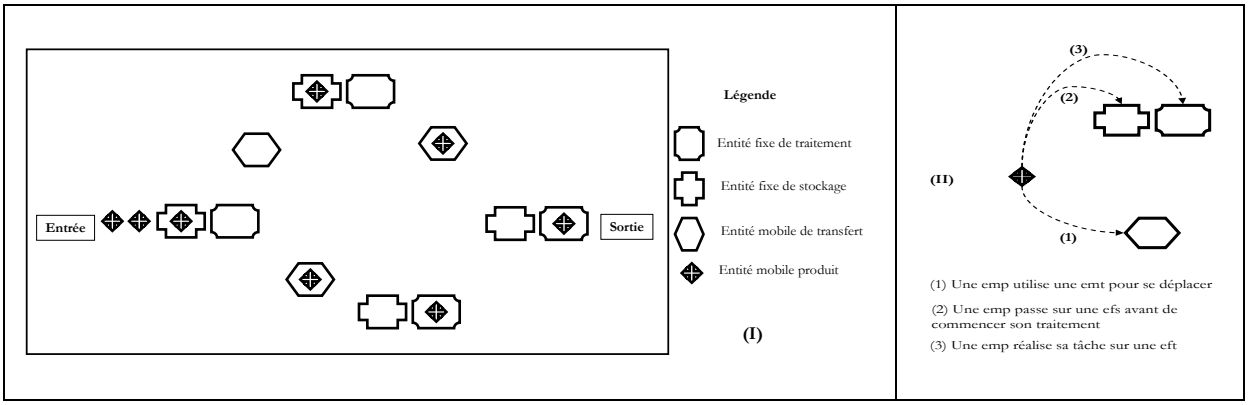


Figure 17- Modèle général des entités et leurs interactions[26]

Après avoir précisé la composition d'un SPBS, nous décrivons dans la partie suivante le fonctionnement des différentes entités au sein d'un SPBS.

2.4. Fonctionnement général d'un SPBS

Dans la modélisation proposée, un SPBS possède deux points de passage principaux : un point d'entrée et un point de sortie (voir Figure 17-I). Par le point d'entrée transitent toutes les entités mobiles produits. Lorsque les différents traitements à réaliser sur une emp sont terminés, elle est transférée au point de sortie. Des traitements sont réalisés par l'ensemble des entités fixes de traitement sur les entités mobiles produits. Chaque emp doit suivre une gamme[27] opératoire qui représente une succession de tâches selon un ordre bien défini (une tâche d'indice *m* est notée t(m)). La durée d'un traitement d'une tâche donnée t(m) sur une eft(k) représente le temps opératoire noté to(k,m). Ce temps d'exécution peut varier d'une eft à l'autre.

Pour changer d'emplacement, chaque emp utilise une entité mobile de transfert[28] pour se déplacer d'une eft à la suivante (c'est le temps nécessaire pour se déplacer entre eft(k) et eft(k')). La durée de ce déplacement est quantifiée par un temps de transfert noté tt(k,k'). La présence simultanée des différentes entités mobiles dans le système et la capacité de traitement finie de chaque eft induisent naturellement des temps d'attente (notés ta(i,m,k)). C'est par définition le temps que met chaque entité mobile produit sur une entité fixe de stockage avant de passer sur une eft, voir Figure 17-II. Pour mieux assimiler cette modélisation, nous l'illustrons avec l'exemple suivant (cet exemple sera exploité pour illustrer les premiers concepts et notions de notre travail)[29] :

Soit un produit sur laquelle une gamme de quatre tâches doit être réalisée (dans l'ordre suivant {t(1)→t(2)→t(3)→t(4)}). Pour cela, nous disposons de quatre machines, voir Figure 18. Dans cet exemple, chaque machine est capable de réaliser deux tâches (par exemple, R(3) est capable d'exécuter les tâches t(1) et t(4)). Ce produit passe successivement sur différentes machines pour réaliser sa gamme. Dans cet exemple, le produit suit la trajectoire suivante : Entrée→R(1)→R(2)→R(2)→R(3)→Sortie.

[26] Ce formalisme représentant les différentes entités sera adopté dans la suite de notre travail.
[27] Dans ce travail, toutes les emp réalisent la même gamme opératoire comportant NbT tâches. Nous verrons dans la suite du document que cette hypothèse n'est pas réductrice.
[28] Nous supposons qu'il n'y a pas de contrainte sur le nombre d'emt, elles sont suffisamment nombreuses et disponibles.
[29] Une mise en forme appropriée (encadrée) sera employée à chaque fois que nous présentons un exemple.

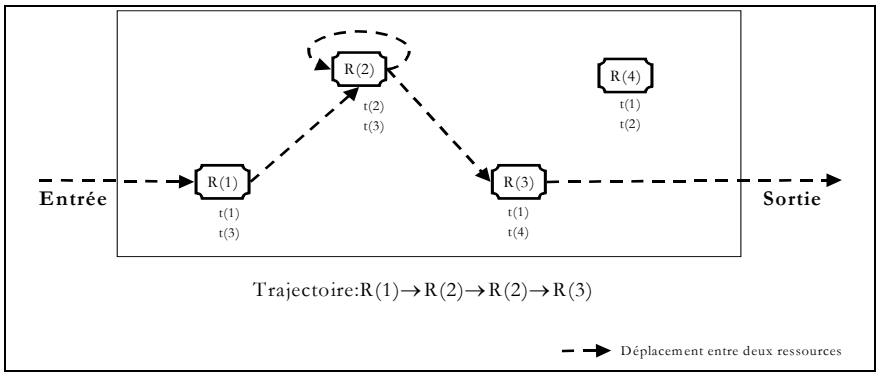


Figure 18- Exemple de trajectoire d'une emp

Cet exemple identifie un point important pour la suite de notre travail, c'est la notion de trajectoire. Celle-ci définit l'ensemble des déplacements d'une entité mobile produit sur les différentes entités fixes de traitement. Nous définissons une trajectoire par la succession de plusieurs étapes (une étape est notée EEmp(k,m)). Selon cette définition, une étape est identifiée à une entité fixe de traitement (EEmp(k,m)=eft(k)). Cette notion de trajectoire s'intègre dans une logique d'amélioration continue des performances. En effet, l'amélioration continue de la qualité des trajectoires des différentes emp est un aspect de l'amélioration continue des performances du système global. De ce fait, nous définissons une performance[30] globale notée PEmp(i) qui quantifie la trajectoire de l'entité mobile produit. Cette performance est une agrégation d'un ensemble de performances intermédiaires (notées PEmp(i,m)) obtenues lors de la réalisation de chaque tâche m (avec $\mathrm{PEmp(i)} = \sum_{m=1}^{NbT} \mathrm{PEmp(i,m)}$).

L'objectif, pour ce type d'exemple, est la recherche continue à optimiser la trajectoire du produit : le système dans lequel évolue le produit doit être régulé *en permanence*, c'est-à-dire qu'il faut optimiser constamment la durée relative à la trajectoire d'un produit. Le tableau Tab. 5 présente un cas de figure qui sera pris en compte dans la suite :

machine	capacité de traitement	temps opératoires
R(1)	t(1); t(3)	3; 5
R(2)	t(2); t(3)	4; 2
R(3)	t(1), t(4)	1; 7
R(4)	t(1); t(2)	4; 8

Tab. 5 Scénario explicatif de l'exemple choisi

Nous établissons tout d'abord la liaison entre le cadre de modélisation des SPBS proposé dans ce travail et cet exemple en identifiant les différents types d'entités présentes dans ce système. Dans cet exemple, l'entité mobile produit (emp) est le produit sur lequel doit être exécuté un ensemble de tâches. Pour ce faire, chaque produit passe successivement sur des machines qui représentent des entités fixes de traitements (eft). Les files d'attente qui existent à proximité des machines constituent des entités fixes de stockage (efs). Enfin, pour transporter ces produits, des chariots filo-guidés ou des tapis roulants sont identifiés à des entités mobiles de transfert (emt). Dans le tableau Tab. 6, nous donnons la correspondance entre les différentes notions de l'exemple pris en compte et celles présentées dans notre modélisation (traitement, transfert, …) :

différentes notions du modèle d'entités proposé	exemple
SPBS	système de fabrication

[30] Nous parlons de performance dans l'absolu.

entité fixe de traitement	machine
entité fixe de stockage	file d'attente
entité mobile produit	produit
entité mobile de transfert	AGV, tapis roulants…
tâche	opération d'usinage
gamme de tâches	exécuter successivement les tâches t(1)→t(2)→t(3)
temps de transfert entre deux eft (tt)	temps de déplacement entre deux machines
temps de traitement sur une eft (to)	temps opératoire sur une machines
temps d'attente sur une efs (ta)	temps d'attente passé avant d'exécuter une tâche
une étape (EEmp)	une machine
trajectoire d'une emp	Entrée→R(1)→R(2)→R(3)→Sortie
performance intermédiaire (PEmp(i,m))	temps intermédiaire passé sur une file d'attente
performance globale (PEmp(i))	temps total passé sur les files d'attente

Tab. 6 Identification des différentes entités et notions de l'exemple choisi

Après avoir défini le cadre de modélisation d'un système de production de biens et de services, la partie suivante est consacrée à la présentation des différentes spécifications nécessaires à la conception d'un système de pilotage hétérarchique pour l'amélioration continue des performances des SPBS. Ce type de structure de pilotage s'intègre parfaitement dans la modélisation à base d'entités que nous proposons. En ce sens, la fonction de pilotage globale résulte de l'émergence des fonctions locales de pilotage 'virtuellement' liées aux entités. Pour cela, nous avons indiqué dans les conclusions du premier chapitre que le système de pilotage proposé *doit* intégrer des capacités d'autonomie et d'apprentissage (une présentation de la typologie et des différentes techniques d'apprentissage est donnée en annexe 2).

3. Spécifications d'un système de pilotage hétérarchique pour l'amélioration continue des performances

Un système de pilotage hétérarchique fait référence à la notion d'entités autonomes, c'est-à-dire des entités toutes capables d'assurer complètement la fonction de pilotage [Pujo et Ounnar, 01].

3.1. Autonomie

Pour présenter le concept d'autonomie nous donnons deux définitions relatives à deux domaines :

- en productique, nous reprenons la définition donnée dans [Van Brussel et al., 98] : 'l'autonomie correspond à la capacité d'une entité de créer et de contrôler l'exécution de ses propres plans et/ou stratégies',
- dans les systèmes multi-agents, Ferber [Ferber, 95] donne la définition suivante : 'un agent est autonome s'il possède une connaissance suffisante pour avoir une certaine liberté de manœuvre et pour pouvoir s'adapter à son environnement. L'autonomie se traduit également par l'ensemble des actions que peut entreprendre cet agent, sans faire intervenir les autres agents du système.

Ces deux définitions mettent en avant la propriété d'une entité autonome à s'adapter à son environnement. Cette adaptation est une caractéristique essentielle de l'amélioration continue des performances. En effet, nous avons indiqué que pour un SPBS la décentralisation des fonctions de pilotage accroît l'évolutivité. Or, celle-ci mesure non seulement la capacité d'un système à être 'toujours réactif', mais aussi l'adéquation des moyens par rapport aux résultats et aux objectifs. Dans ce cadre, Tharumarajah et al. [Tharumarajah et al., 96] confirment qu'une approche

favorisant l'autonomie des centres de décision est bien adaptée pour assurer cette évolutivité. Enfin, l'autonomie permet à une entité de maîtriser de ses liens avec l'environnement afin de pouvoir disposer de moyens de 'ripostes face aux perturbations' [Trentesaux, 02].

Le concept d'autonomie est étroitement lié à la capacité de prise de décision. Il nous paraît alors nécessaire de présenter la notion de prise de décision. En ce sens, nous commençons par présenter les principaux types des processus décisionnels dans les systèmes de pilotage.

3.1.1. Typologie des processus décisionnels dans les systèmes de pilotage

La typologie présentée dans [Trentesaux, 02] permet de recouvrir la plupart des processus de pilotage en milieu industriel. L'auteur distingue trois catégories de processus de pilotage suivant les types de décisions possibles :

- processus de pilotage selon le type de décision : à base de choix (sous contrainte, sur contrainte et portant sur la première ou la seconde alternative [Erschler et al., 97]), de tri ou de rangement [Roy et Bouyssou, 93]. Le premier type de processus (choix) est prépondérant en gestion de production [Thiel, 93],
- processus de pilotage selon la présence d'un opérateur (aide au pilotage) ou non (pilotage automatisé) [Trentesaux et al., 97]. Dans le premier cas, il est possible d'intégrer la complexité de la tâche de l'opérateur humain dans le soutien à la réalisation des activités de décision [Millot, 99],
- processus de pilotage structuré ou peu structuré [Simon, 77] ou de manière relativement proche, des processus de pilotage à base de compétence, règle ou connaissance [Rasmussen, 83].

Nous pensons que, dans une optique d'amélioration continue des performances, le premier type (processus de pilotage selon le type de décision) est adéquat. Outre sa prépondérance en milieu industriel, ce type de processus de pilotage offre la possibilité de faire adapter le fonctionnement d'un SPBS à son contexte en optant *continuellement* pour le meilleur *choix*.

3.1.2. Processus décisionnel à base de choix

La décision est souvent présentée comme le fait d'un individu isolé (le 'décideur') exerçant librement un choix entre plusieurs possibilités d'actions à un moment donné dans le temps. Néanmoins, même si, en dernier ressort, la responsabilité d'une décision incombe à un individu clairement identifié, celle-ci est souvent la résultante d'interactions entre de multiples individus au cours d'un processus de décision [Roy et Bouyssou, 93]. Selon cette définition, un processus décisionnel s'articule autour de deux points essentiels :

3.1.2.1 Application d'une action…

Une action 'a' est la représentation d'une éventuelle contribution à la décision globale susceptible, eu égard à l'état d'avancement du processus de décision, d'être envisagée de façon autonome et de servir de point d'application à l'aide à la décision (ce point d'application pouvant suffire à caractériser 'a') [Roy et Bouyssou, 93].

3.1.2.2 … selon un ou plusieurs critère(s)

Aider à décider, c'est en tout premier lieu aider à clarifier la formation, la transformation et l'argumentation des préférences. À ce niveau, le concept-clé est celui du critère. Formellement, un

critère C est une fonction à valeurs réelles définie sur l'ensemble A des actions potentielles a(j), j=1..NbD de telle sorte qu'il soit possible de raisonner ou de décrire le résultat de la comparaison de deux actions a et b de l'ensemble A à partir des deux nombres C(a) et C(b). Il est commode, dans le cas de plusieurs critères, de résumer le résultat de l'analyse des conséquences dans un tableau de performances. Soit un ensemble A de NbD actions et C(1), …, C(NbC) avec NbC le nombre de critères formant une famille C. On appelle tableau de performances de A sur C le tableau contenant la valeur c(i,j) de l'action a(j) pour le critère C(i), voir tableau Tab. 7 :

actions/critère	C(1)	C(2)	…	C(i)	…	C(NbC)
a(1)	…	…	…	…	…	…
a(2)	…	…	…	…	…	…
a(j)	…	…	…	c(i,j)	…	…
a(NbD)	…	…	…	…	…	…

Tab. 7 Tableau de performances ([Roy et Bouyssou, 93])

Dans notre travail, nous spécifions l'élaboration d'un choix entre plusieurs possibilités d'actions par l'adoption d'un *critère* afin de déterminer la bonne décision.

3.1.3. Spécifications du processus décisionnel adopté

Le cadre de définition générique défini dans [Roy et Bouyssou, 93] basé sur la notion d'action et de critère, est approprié à la spécification des mécanismes décisionnels capables d'assurer l'amélioration continue des performances. Cette définition met en évidence la possibilité d'évaluer la ou les conséquence(s) d'une décision prise selon un critère. Cette appréciation est adéquate dans une optique d'évaluation continue des performances. Néanmoins, la notion de critère nous paraît assez générale pouvant être exploitée plus finement. En effet, la spécification d'un critère C, selon un point de vue donné, peut donner lieu à un ensemble de *stratégies*. Une stratégie découle de l'application d'un critère selon un objectif donné (la notion de stratégie sera explicitée dans la partie suivante au travers de l'exemple présenté précédemment).

3.1.3.1 Adoption des stratégies dans le processus décisionnel

Dans notre approche, nous distinguons entre une vision *binaire* ou *n-aire* d'un critère :

- une vision binaire d'un critère consiste à en déduire uniquement deux stratégies (Figure 19-a),
- une vision n-aire d'un critère consiste à en déduire n stratégies (n>2, voir Figure 19-b).

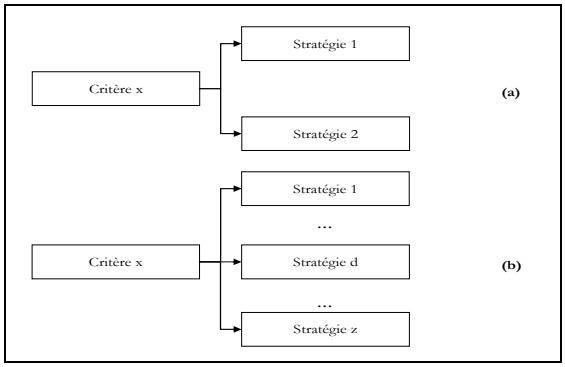


Figure 19- Vision binaire (a) ou n-aire (b) des critères

L'illustration de cette notion de vision binaire/n-aire est donnée à l'aide de l'exemple du système de production. Nous présentons tout d'abord un scénario dans lequel un produit (P(i))

doit prendre une décision pour réaliser sa prochaine tâche en tenant compte des données suivantes :

données relatives à P(i)	valeurs
position actuelle	le produit est sur la machines R(2)
prochaine tâche	réaliser la tâche t(2)
distances entre les machines	R(2)→ R(1) : 10 m
	R(2)→ R(3) : 20 m
	R(2)→ R(4) : 30 m
taux de remplissage des différentes files d'attentes relatives à …	R(1) : 100%
	R(3) : 50%
	R(4) : 10%

Tab. 8 Scénario pour la distinction binaire ou n-aire des critères

Avec les données mentionnées dans le tableau Tab. 8, cette notion relative à la vision binaire ou n-aire d'un critère présentée précédemment est expliquée dans le tableau Tab. 9 :

(a) vision binaire d'un critère		
critère		
proximité d'une machine		
stratégie 1=Aller vers la machine la plus proche	stratégie 2=Aller vers la machine la plus lointaine	
décision : aller vers la machine R(1)	décision : aller vers la machine R(4)	
(b) vision n-aire d'un critère (n=3)		
critère		
occupation d'une file d'attente		
stratégie 1=Aller vers la file d'attente la moins remplie	stratégie 2=Aller vers la file d'attente dont le taux de remplissage est inférieur à 30%	stratégie 3=Aller vers la file d'attente la plus remplie
décision : aller vers la machine R(4)	décision : aller vers machine R(3)	décision : aller vers la machine R(1)

Tab. 9 Exemple d'une distinction binaire ou n-aire selon des critères

Dans ce travail, nous avons opté pour une vision binaire d'un critère pour les raisons suivantes :

- généralement, dans une logique d'amélioration continue des performances, le but consiste à maximiser ou minimiser une fonction objectif. À notre avis, une vision binaire d'un critère convient mieux dans ce cas,
- il est nécessaire d'éviter la multiplication des stratégies afin d'empêcher l'apparition d'une explosion combinatoire dans le processus décisionnel,
- nous pensons qu'une vision n-aire d'un critère s'intègre mieux dans un cadre de logique floue qui dépasse notre champ d'investigation et pourra être traitée séparément en terme de perspective à ce travail.

Nous en déduisons la première spécification (S1) : Une entité appartenant à un SPBS doit prendre, d'une façon autonome, un ensemble de décisions lors de son fonctionnement. Une décision est prise en appliquant une stratégie et l'agrégation des différentes décisions forme le processus de pilotage d'un SPBS qui doit assurer l'amélioration continue de ses performances.

Cependant, la distribution des fonctions de pilotage localement sur les différentes entités ne doit pas omettre d'assurer la cohérence globale des décisions. Ce point est détaillé dans la partie

suivante.

3.1.3.2 Cohérence des décisions prises

Lorsque les fonctions locales de pilotage émanent des entités dont la typologie structurelle ou opérationnelle est assez différente, des problèmes liés à la cohérence des décisions prises peuvent apparaître. En ce sens, nous estimons que doter les quatre types d'entités {eft, efs, emp et emt} de des fonctions décisionnelles engendrera inévitablement des problèmes de cohérence et éventuellement des conflits lors de la prise de décision. Une illustration de ce problème est celle du cas où une eft et une efs gèrent en même temps, étant donné leur proximité, les mêmes emp : dans ce cas quelle décision prévaudra ?

Dans notre approche, nous choisissons de distinguer deux classes d'entités : les entités *actives* et les entités *passives*. En effet, une distinction sera faite entre les entités capables de *prendre* une décision et de *l'exécuter* (appelées entités actives) et celles qui sont capables *seulement* d'*exécuter* les décisions prises par les autres entités (appelées entités passives). Dans cette logique, nous considérons d'une part que les entités fixes de traitement et les entités mobiles produits comme entités actives et d'autre part que les entités mobiles de transfert et les entités fixes de stockage appartiennent à la classe des entités passives. Cette classification se base sur l'aspect fonctionnel de chaque entité. Le tableau Tab. 10 fait la correspondance entre les deux types d'entités pour le modèle d'entités proposé et pour l'exemple considéré :

	Modèle d'entités proposé	Exemple pris en compte
entités actives	eft, emp	machine, produit
entités passives	efs, emt	file d'attente, AGV

Tab. 10 Distinction entre les entités actives et les entités passives

D'une part, les décisions prises par une entité mobile produit détermineront les différentes étapes de sa trajectoire, et d'autre part les actions appliquées par une entité fixe de traitement se traduiront par un ensemble de décisions prises. Dans le paragraphe 3.1.2, nous avons explicité, au travers de l'exemple pris en compte, une décision que peut prendre un produit selon une stratégie donnée. Considérons le scénario représenté dans le tableau Tab. 11 :

exemple prise en compte	machine R(1)	
produits présents dans cette machine	P(1),P(2),P(3) et P(4)	
date d'arrivée de chaque produit	P(1)	3
	P(2)	4
	P(3)	2
	P(4)	1

Tab. 11 Scénario pour la prise d'une décision d'une machine dans l'exemple choisi

Nous illustrons la notion de décision prise par une entité fixe de traitement dans le tableau Tab. 12 : en tenant compte de la date d'arrivée des différents produits, la machine R(1) peut prendre une décision bien déterminée :

prise de décision de la machine R(1)	
critère= date d'arrivée	
stratégie 1=traiter en priorité le produit arrivé le premier (FIFO)	stratégie 2= traiter en priorité le produit arrivé le dernier (LIFO)
décision 1 : traiter le produit P(4)	décision 2 : traiter le produit P(2)

Tab. 12 Prise de décision dans l'exemple choisi

Nous en déduisons la deuxième spécification (S2) : Parmi les quatre types d'entités qui composent un SPBS,

les entités actives (entités fixes de traitement et les entités mobiles produits) sont capables de prendre et d'exécuter des décision. Les entités passives (entités fixes de stockage et entités fixes de transfert) ont pour mission d'exécuter les décisions prises par les entités actives.

La spécification des entités actives nous permet maintenant de définir, pour chaque type d'entité, un ensemble de critères (et par conséquent un ensemble de stratégies) à partir desquels les décisions seront élaborées. Le tableau Tab. 13 identifie pour les entités mobiles produits et pour les entités fixes de traitement leurs stratégies correspondantes :

emp	critère	…	Cm(y)		…	Cm(NCm)	
	stratégie	…	SEmp(2y-1)[31]	SEmp(2y)	…	SEmp(2NCm-1)	SEmp(2NCm)
eft	critère	…	Cf(x)		…	Cf(NCf)	
	stratégie	…	SEft(2x-1)[32]	SEft(2x)	…	SEft(2NCf-1)	SEft(2NCf)

Tab. 13 Critères/stratégies relatives aux entités actives

L'illustration des décisions prises par un produit ou une machine constitue une introduction à la généralisation du processus décisionnel des entités actives dans un SPBS :

- une entité fixe de traitement (eft(k)) choisit de traiter en priorité une entité mobile produit parmi un ensemble d'emp (noté EMP(k)) présentes dans la file d'attente correspondante (efs(k)). Ce choix est réalisé en adoptant une stratégie appartenant à l'ensemble des stratégies (noté ESF(k)) disponibles pour cette eft(k) (dans le troisième chapitre, nous détaillons comment ce choix est fait), voir Figure 20,
- deuxièmement, une entité mobile produit (emp(i)) choisit d'aller sur une eft donnée pour réaliser une tâche m. Cette destination est sélectionnée parmi un ensemble d'eft candidates (noté EFT(m)). Ce choix est réalisé en adoptant une stratégie appartenant à l'ensemble des stratégies (noté ESM[33]) relatif à chaque emp(i), voir Figure 21.

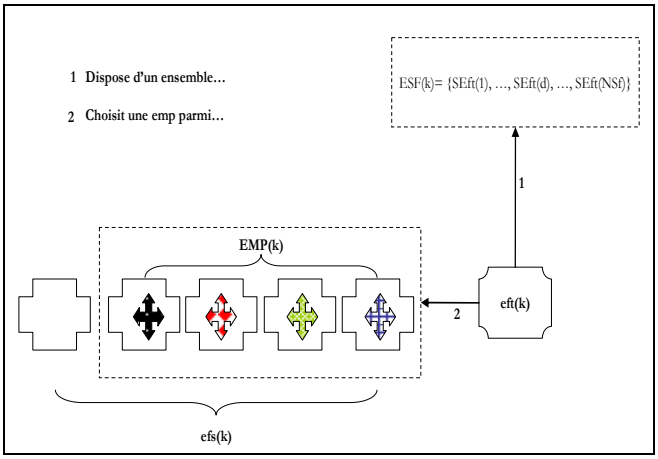


Figure 20- Les principaux éléments du processus décisionnel d'une eft

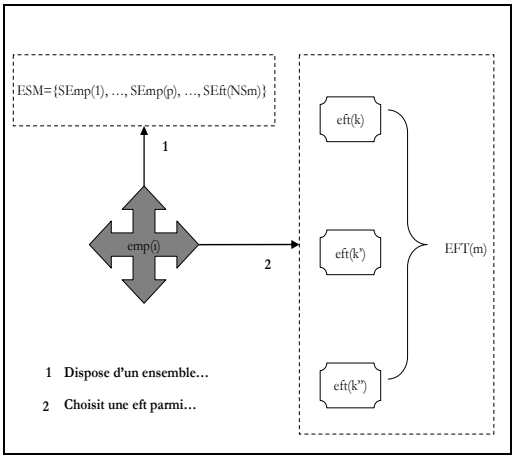


Figure 21- Les principaux éléments du processus décisionnel d'une emp

Précisons qu'en cas d'égalité lors de l'adoption d'une stratégie par une entité active (par exemple si une entité mobile produit adopte un stratégie à l'instant t et qu'il y a deux ou plusieurs

[31] Le coefficient '2' (2x) résulte de la vision binaire d'un critère prise en compte.
[32] Dans ce qui suit, nous utilisons la notation SEmp(p) et SEft(d) (voir les différentes notations adoptées dans ce document).
[33] Cet ensemble est commun pour toutes les emp. Par contre, chaque entité fixe de traitement a son propre ESF (k).

entités fixes de traitements qui présentent des caractéristiques identiques à cet instant) alors un choix aléatoire est opéré afin d'éviter les conflits.

3.1.3.3 Fréquence et types des décisions prises

Par ailleurs, l'adoption de ce processus décisionnel des entités actives pose une question triviale : combien de décisions une entité active prendra t-elle au cours de son fonctionnement (cycle de vie) ? Autant la réponse est évidente pour une entité mobile produit (le nombre de décisions est égal au nombre de tâches à réaliser (NbT)), autant il est nécessaire de spécifier ce nombre de décisions dans le cas d'une entité fixe de traitement. Intuitivement, il existe deux possibilités : un nombre très grand (fini ou infini) de décisions ou une prise de décision *cyclique*. Nous pouvons comprendre plus aisément cette différence à travers notre exemple :

Une machine fonctionne généralement d'une façon discontinue, c'est-à-dire qu'il y a des arrêts (fin de journée, week-ends, vacances, …). Autrement dit, les décisions qu'elle prend se renouvellent d'une façon cyclique et continue.

Dans notre travail, nous optons pour une prise de décision cyclique de l'entité fixe de traitement. La justification de ce choix réside dans la présence de la notion de cycle dans les systèmes industriels actuels (initialisation, mise à zéro ou encore les arrêts pour maintenance des unités de production). En outre, une évaluation continue et cyclique des décisions prises s'intègre parfaitement dans une logique d'amélioration continue des performances. De ce fait, nous spécifions, pour chaque entité fixe de traitement, un cycle des décisions. Une fois un cycle (noté CEft(k,h)) est achevé, un nouveau est commencé. Chaque cycle est composé de NbD décisions.

Pour définir plus finement le processus décisionnel d'une entité active, nous estimons que la détermination d'une décision peut être de deux types : *dynamique* ou *statique*. Dans le premier cas, la décision adoptée est obtenue en appliquant une stratégie, donc il s'agit d'une décision choisie parmi un ensemble de décisions possibles. Dans le deuxième cas, la décision est déterminée a priori (adoption d'une étape connue). L'illustration, grâce à notre exemple, de cette notion de décision dynamique/statique est donnée dans le tableau Tab. 14 :

exemple pris en compte	type de décision		exemple de décision
produit	décision dynamique	application d'une stratégie	'aller sur la machine la plus proche' (critère=proximité)
	décision statique	adoption d'une étape	'aller sur la machine R(1)'
machine	décision dynamique	application d'une stratégie	'traiter le produit qui est arrivé le premier sur la file d'attente (FIFO)' (critère=ordre d'arrivée)
	décision statique	adoption d'une étape	'traiter le produit P(10)'

Tab. 14 Application d'une stratégie/adoption d'une étape dans l'exemple pris en compte

Dans la modélisation que nous proposons, cette notion de processus décisionnel dynamique/statique d'une entité mobile produit est spécifiée comme suit :

- lorsqu'une emp(i) adopte une stratégie SEmp durant son processus décisionnel pour réaliser une tâche, l'eft sur laquelle sera exécutée cette tâche est sélectionnée parmi un ensemble d'eft candidates : c'est un décision dynamique,
- lorsqu'une emp(i) opte pour une étape EEmp durant son processus décisionnel pour réaliser une tâche, l'eft sur laquelle sera exécutée cette tâche est connue a priori : c'est un décision statique. Le tableau Tab. 15 présente les NbT stratégies (respectivement étapes) adoptées (respectivement sélectionnées) pendant le processus décisionnel d'une emp :

décision	1	…	m	…	NbT
stratégie	SEmp(*,1)[34]	…	SEmp(*,m)	…	SEmp(*,NbT)
étape	EEmp(*,1)	…	EEmp(*,m)	…	EEmp(*,NbT)

Tab. 15 Ensemble des stratégies et étapes relatives au processus décisionnel d'une emp

Par ailleurs, notons t_m l'instant de prise de la première décision d'une emp(i) (c'est par définition la date d'entrée d'une emp dans le SPBS). À cet instant t_m, nous supposons que $NbM(i,t_m)$ entités mobiles produits ont déjà quitté le SPBS.

Nous adoptons la même logique pour les entités fixes de traitement. Chaque eft renouvelle ses décisions cycliquement : un cycle, d'indice h, contient NbD décisions qui sont élaborées :

- dynamiquement en adoptant un ensemble de stratégies SEft pour exécuter NbD décisions pour chaque cycle,
- statiquement en sélectionnant des étapes EEft, c'est-à-dire en traitant des emp (en priorité) non pas selon un critère donné, mais plutôt en se basant sur des données statiques (par exemple le rang d'une emp dans l'entité de stockage). Le tableau Tab. 16 présente les NbD stratégies (respectivement étapes) adoptées (respectivement sélectionnées) pendant le processus décisionnel d'une eft :

décision	1	…	n	…	NbD
stratégie	SEft(*,1)	…	SEft(*,n)	…	SEft(*,NbD)
étape	EEft(*,1)	…	EEft(*,n)	…	EEft(*,NbD)

Tab. 16 Ensemble des stratégies et étapes relatives au processus décisionnel d'une eft

Notons aussi t_f l'instant de prise de la première décision pour réaliser un nouveau cycle relatif à une eft(k). À cet instant t_f, nous supposons que cette eft(k) a déjà effectué $NbC(k,t_f)$ cycles.

Nous en déduisons la troisième spécification (S3) : dans une optique d'amélioration continue des performances, les décisions prises par les entités actives doivent être évaluées en permanence. La réalisation de cette évaluation diffère selon le type de l'entité active : elle est réalisée à la fin de chaque cycle de décisions pour une entité fixe de traitement et suite à l'exécution des NbT tâches pour une entité mobile produit.

La Figure 22 résume les différentes notions relatives aux entités qui composent un SPBS. Dans la suite de ce travail, nous nous intéressons uniquement aux entités actives (eft et emp).

[34] Le signe (*) indique que l'indice d'une stratégie est inconnu et varie entre 1 et NSf. Cette convention sera adoptée dans le reste du document.

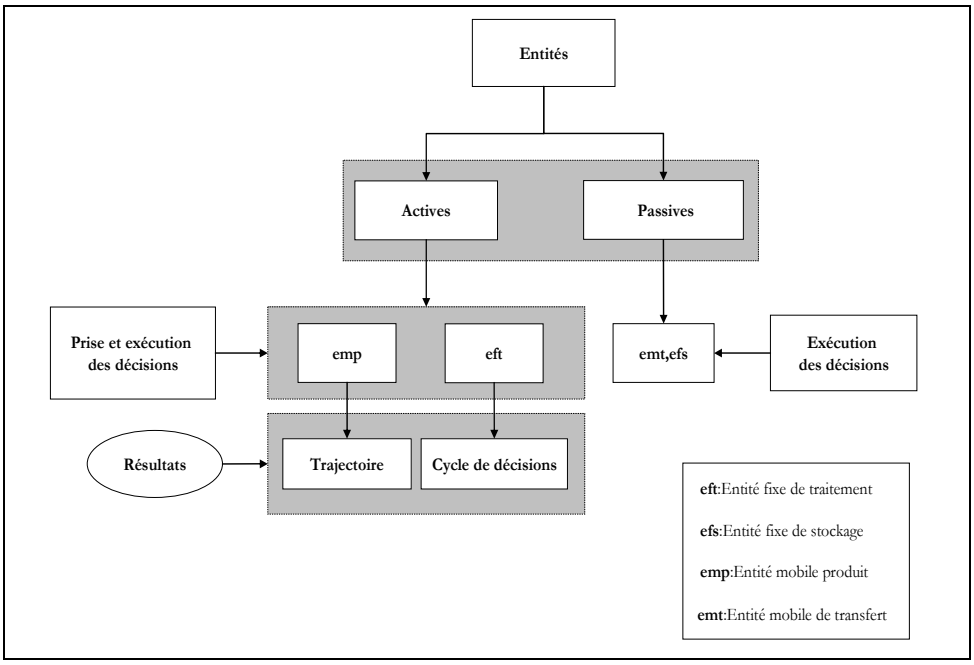


Figure 22- Typologie opérationnelle/processus décisionnel des entités d'un SPBS

Après avoir présenté les spécifications relatives à l'autonomie qui doit caractériser un système de pilotage hétérarchique conçu pour garantir l'amélioration continue des performances, la partie suivante donne les spécifications relatives aux mécanismes d'apprentissage.

3.2. Apprentissage

3.2.1. Définition

L'apprentissage automatique (machine learning) est un champ d'étude de l'intelligence artificielle. Il fait référence au développement, l'analyse et l'implémentation de méthodes qui permettent à une machine (au sens large, par exemple agent intelligent, robot, machine-outil) d'évoluer et de remplir des tâches associées à une intelligence artificielle grâce à un processus d'apprentissage. Cet apprentissage conduit à un système qui s'optimise en fonction de l'environnement, les expériences et les résultats observés [Monostori, 99]. Cette optimisation nécessite l'acquisition de nouveaux savoirs ou savoir-faire, c'est-à-dire la mise en relation entre un événement provoqué par l'extérieur (stimulus) et une réaction adéquate du système. L'apprentissage peut être un phénomène individuel ou collectif (une entité ou une population d'entités qui apprend). La distinction entre les deux dépend de l'échelle utilisée : pour un neurobiologiste, l'apprentissage chez l'homme (population de cellules) est généralement un apprentissage collectif au niveau de sa population de neurones (cellules). Appliqué aux systèmes de production de biens et de services, l'apprentissage d'une entité s'oppose intuitivement à un comportement prédéfini. En effet, une entité est capable d'apprendre si elle possède des mécanismes décisionnels qui lui permettent d'élaborer ses décisions en fonction de l'état de son environnement. En productique, l'apprentissage est une technique dont l'utilisation a vu un essor considérable et ce depuis la généralisation des contextes instables ou stochastiques dans le milieu industriel.

En se référant à la définition de l'apprentissage que nous avons donnée, nous pouvons affirmer que l'intégration de cette notion dans un système de pilotage hétérarchique peut contribuer à l'amélioration continue des performances d'un SPBS. Cette affirmation s'appuie sur le fait que les mécanismes d'apprentissage :

- peuvent être un phénomène individuel qui permet aux entités actives, comme nous les avons spécifiées, d'évoluer dans leur environnement,
- permettent à une entité active d'optimiser continuellement son fonctionnement en tenant compte de l'état de l'environnement, de ses expériences et des résultats observés,
- s'opposent intuitivement à un comportement prédéfini, c'est-à-dire que ces mécanismes génèrent, de façon continue, des nouvelles connaissances pour le système de pilotage. La génération de ces connaissances est utile pour faire face à des situations imprévisibles.

Dans la partie suivante, nous formalisons et spécifions des mécanismes d'apprentissage capables de résoudre les problématiques présentées dans le premier chapitre. Tout d'abord, nous remarquons que dans la littérature, il n'existe pas une formalisation générique des mécanismes d'apprentissage dédiés à l'amélioration continue des performances des SPBS. Souvent, les auteurs se focalisent sur la proposition d'un ensemble de mécanismes d'apprentissage sans pour autant les formaliser. En d'autres termes, pour des mécanismes d'apprentissage, nous avons rencontré des difficultés à trouver la ou les réponse(s) aux questions suivantes :

- quelle est la source de l'apprentissage ?
- dans le cas où une entité a apprend d'une entité b (apprentissage collectif), est ce que cette entité b est encore en fonctionnement (dans le SPBS) ou non ?
- quel est l'objet de l'apprentissage ?
- à quel moment du cycle de vie d'une entité les mécanismes d'apprentissage sont activés ?

Nous proposons dans la section suivante de répondre à ces quatre questions dans un cadre formalisé et présentons un cadre de définition d'un processus d'apprentissage d'une entité active (noté A(EA)).

3.2.2. Formalisation des mécanismes d'apprentissage

3.2.2.1 Source de l'apprentissage

La source d'apprentissage, notée S, détermine la nature des interactions que peut avoir une entité active pendant le processus d'apprentissage. Autrement dit, nous distinguons entre les trois cas suivants :

- une entité active apprend des actions des autres entités : nous parlons alors d'apprentissage *externe*, voir Figure 23-a,
- une entité active apprend de ses propres actions : nous parlons alors d'apprentissage *interne*, voir Figure 23-b,
- une entité apprend de ses propres actions *et* celles des autres entités : nous parlons alors d'apprentissage *mixte*, voir Figure 23-c.

Un produit, lors de la réalisation de sa gamme, apprend à partir de son propre comportement (interne), des comportements des autres produits ou, à la fois, de son comportement et celui des autres produits (mixte).

D'où S∈ {interne, externe, mixte}, voir Figure 23 qui illustre ces trois cas pour une entité mobile produit.

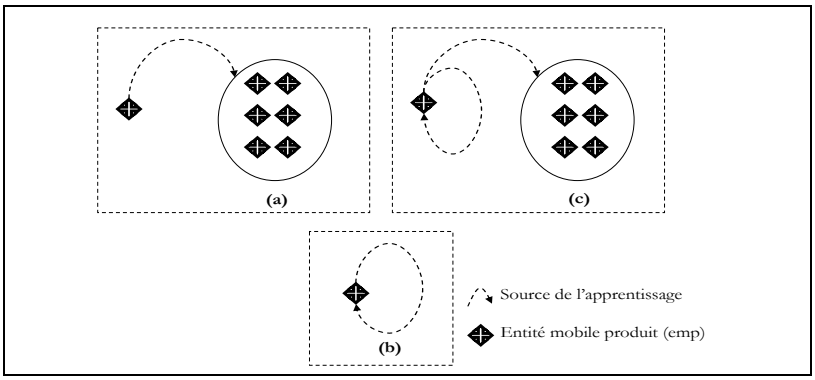


Figure 23- Source de l'apprentissage pour une emp ((a) externe, (b) interne, (c) mixte)

Par ailleurs, dans le cas où la source de l'apprentissage est externe ou mixte, nous spécifions l'état des entités à partir desquelles les mécanismes d'apprentissage sont élaborés.

3.2.2.2 État des entités sources de l'apprentissage

Pour un apprentissage de source externe ou mixte, trois cas de figure se présentent :

- l'apprentissage se fait en se basant sur les actions d'un ensemble d'entités dont les traitements sont terminés[35]. Nous parlons alors d'apprentissage *décalé* (Figure 24-a),
- l'apprentissage se fait à partir des actions d'un ensemble d'entités qui sont encore sous traitement. Nous parlons alors d'apprentissage *parallèle* (Figure 24-b),
- l'apprentissage se fait en se fondant, conjointement, sur les actions d'un ensemble d'entités dont les traitements sont terminés et d'autres qui sont encore sous traitement. Nous parlons alors d'apprentissage *mixte* (Figure 24-c).

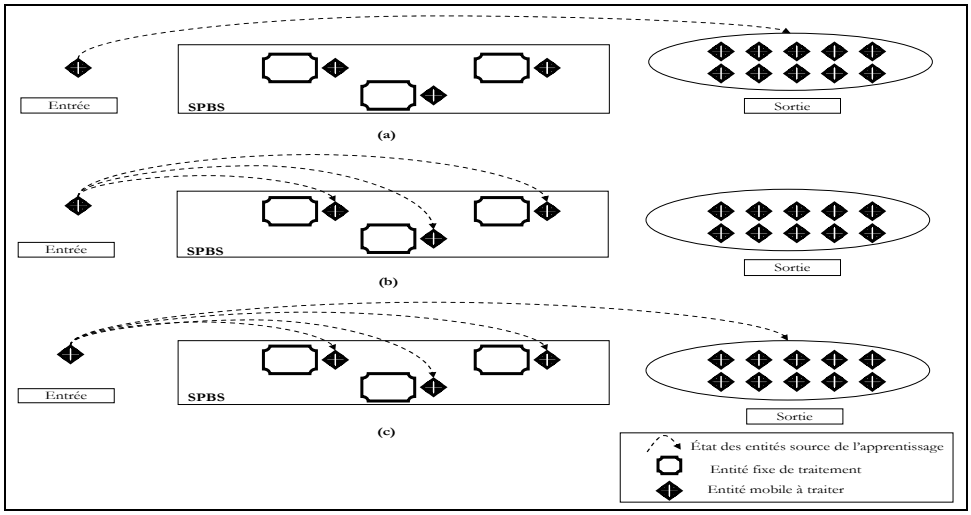


Figure 24- État des entités source de l'apprentissage ((a) décalé, (b) parallèle, (c) mixte)

Pour un produit, qui apprend à partir du comportement des autres produits, il peut effectuer cet apprentissage en observant le comportement des produits encore présents dans le système (parallèle), ou en observant le comportement des produits qui ont fini leur gamme (décalé) ou encore en observant, conjointement, le comportement des deux (mixte).

[35] Cette spécification est également applicable pour les entités fixes de traitements.

D'où selon cette spécification de l'apprentissage de source externe ou mixte, l'état, noté E, prend les trois valeurs suivantes {parallèle, décalé, mixte}.

3.2.2.3 Objet de l'apprentissage

L'objet de l'apprentissage, noté O, exprime sur quoi seront appliqués les mécanismes d'apprentissage. Nous distinguons en ce sens deux objets différents d'apprentissage :

- une stratégie : l'objet de l'apprentissage est une stratégie appartenant à l'ensemble des stratégies disponibles pour chaque entité active. Nous dirons qu'une entité *adopte* une stratégie. Dans ce cas, la décision sera prise en fonction de cette stratégie (Figure 25-a),
- une étape : l'objet de l'apprentissage est une étape faisant partie des différentes étapes disponibles pour chaque entité active. Nous avons alors une entité qui *opte pour* une étape. Dans ce cas, la décision est prédéfinie (Figure 25-b).

L'objet d'apprentissage d'un produit peut être soit une étape (aller directement vers une ville bien précise) ou l'application d'une stratégie donnée (choix d'une ville parmi un ensemble de villes 'candidates').

D'où O∈ {étape, stratégie}.

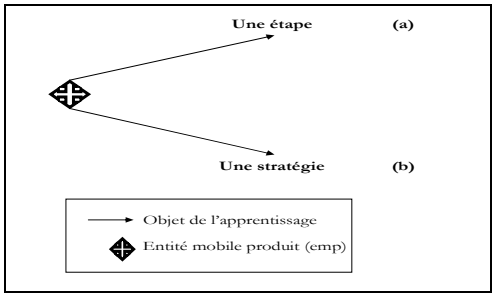


Figure 25- Objet de l'apprentissage d'une emp (stratégie, étape)

3.2.2.4 Moment de l'apprentissage

Le moment d'apprentissage, noté M, traduit la période pendant laquelle une entité active entreprend ses mécanismes d'apprentissage. En effet, une entité active peut déterminer l'ensemble de ses décisions soit :

- avant de commencer ses traitements : le moment de l'apprentissage est noté *anticipé* (Figure 26-a).
- pendant l'exécution de ses traitements : le moment de l'apprentissage est noté *simultané* (Figure 26-b).
- avant et pendant l'exécution de ses traitement : le moment de l'apprentissage est noté *mixte*.

D'où M∈ {anticipé, simultané, mixte}.

Un produit apprend son comportement future soit avant d'entrer dans le système routier (moment anticipé), soit pendant qu'il est dans le système (moment simultané) ou encore en apprenant une partie avant et l'autre partie pendant sa présence dans le système (moment mixte).

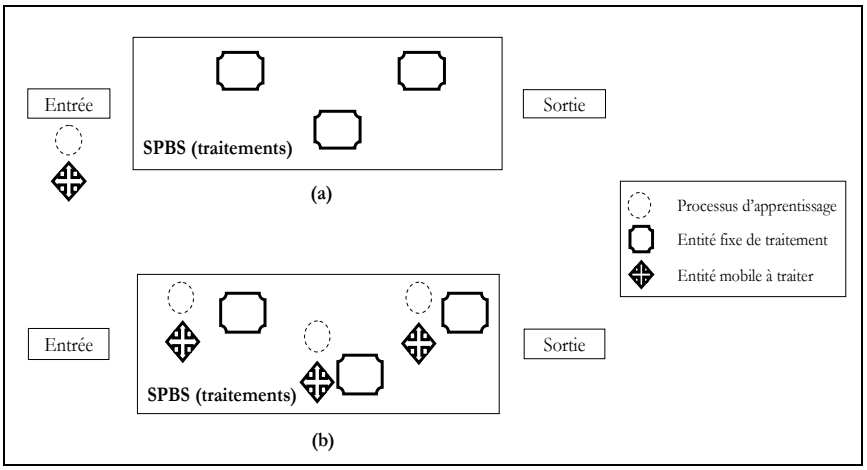


Figure 26- Moment de l'apprentissage d'une emp (anticipé, simultané, mixte)

Cependant, une distinction doit être faite pour la notion du moment d'apprentissage. En effet, il y a une différence entre 'entrer dans le système' (t=0) et 'débuter un fonctionnement' (à n'importe quel instant du cycle du vie d'une entité active). Dans notre travail, nous choisissons le terme 'commencer un traitement'. Cette nuance garantira l'applicabilité de la notion du moment d'apprentissage pour les entités fixes de traitements et les entités mobiles produits.

Pour résumer, nous formalisons le processus d'apprentissage d'une entité active, noté A(EA), par le quadruplet : A(EA)=<<S,E>,O,M>. Nous donnons ci-dessous un exemple pour une entité fixe de traitement et une entité mobile produit :

- A(eft)= <<interne,∅[36]>,stratégie,simultané>,
- A(emp)= <<externe,décalé>,étape,anticipé>.

Le tableau Tab. 17 récapitule les différentes possibilités qu'offre cette spécification des mécanismes d'apprentissage (42 combinaisons possibles) :

Source de l'apprentissage (S)	Objet de l'apprentissage (O)		Moment de l'apprentissage (M)
interne	stratégie		anticipé
			simultané
			mixte
	étape		anticipé
			simultané
			mixte
externe	État des entités source de l'apprentissage (E)		anticipé
	parallèle (à partir des entités encore sous traitement)	stratégie	
			simultané
			mixte
		étape	anticipé
			simultané
			mixte
	décalé (à partir des entités qui ont fini leurs traitements)	stratégie	anticipé
			simultané
			mixte
		étape	anticipé

[36] Ce symbole est utilisé dans le cas où la source de l'apprentissage est interne.

			simultané
			mixte
	mixte	stratégie	anticipé
			simultané
			mixte
		étape	anticipé
			simultané
			mixte
mixte	parallèle (à partir des entités encore sous traitement)	stratégie	anticipé
			simultané
			mixte
		étape	anticipé
			simultané
			mixte
	décalé (à partir des entités qui ont fini leurs traitements)	stratégie	anticipé
			simultané
			mixte
		étape	anticipé
			simultané
			mixte
	mixte	stratégie	anticipé
			simultané
			mixte
		étape	anticipé
			simultané
			mixte

Tab. 17 Spécification de l'apprentissage d'une entité active selon le quadruplet <<S,E>,O,M>

Après la présentation de la formalisation des mécanismes d'apprentissage, nous donnons dans la section suivante les spécifications nécessaires à ces mécanismes pour qu'ils puissent assurer l'amélioration continue des performances. En ce sens, les mécanismes d'apprentissage doivent garantir la pertinence des décisions prises, la diversification et l'évaluation continues de ces décisions et enfin le rejet des perturbations (tolérance aux pannes).

3.2.3. Apprendre à prendre des décisions pertinentes

Quelle que soit la source interne ou externe, les mécanismes d'apprentissage doivent s'appuyer sur des données pertinentes donnant une image la plus fidèle possible de l'état du SPBS. Pour résoudre la problématique relative à la non pertinence des données, nous avons recensé trois approches issues de trois domaines différents : le facteur d'oubli (automatique) le facteur d'actualisation (finance) et le data mining (intelligence artificielle) :

- dans l'identification d'un processus dynamique, dont les caractéristiques s'altèrent lentement au cours du temps, les échantillons récents doivent bénéficier d'un poids plus important que les anciens, lesquels sont périmés. En quelque sorte, il ne faut pas que le lointain passé sature le présent [Longchamp, 95]. Une solution à ce problème consiste à considérer un scalaire (appelé facteur d'oubli et noté λ) appartenant à]0,1[qui permet de donner plus d'importance aux mesures récentes que les anciennes [Landau, 93]. En pratique, le facteur λ est pris dans l'intervalle [0.95,0.99],
- en finance, le coefficient d'actualisation ou facteur d'actualisation (en anglais discount factor) est la quantité par laquelle, à une date donnée (la date d'actualisation), il faut multiplier le montant d'un flux financier passé ou à venir pour obtenir sa valeur actualisée à cette date. Cela permet d'estimer la valeur actuelle (la valeur aujourd'hui) d'un flux financier futur [Sødal, 06],

- en intelligence artificielle, le data mining a pour objet d'extraire un savoir implicite à partir de grandes quantités de données, par des méthodes automatiques ou semi-automatiques. Dans cette approche, des algorithmes détecteurs de corrélations et de classifieurs automatiques sont utilisés pour trouver des schémas 'intéressants' selon des critères fixés au départ. Le data mining est une approche très différente de la méthode statistique. En effet, le data mining fait émerger à partir de données brutes des hypothèses que souvent l'expérimentateur ne soupçonne même pas. Tandis que la méthode statistique exige qu'on se fixe une hypothèse, que les données vont confirmer ou non. Des applications ont été réalisées en productique (voir par exemple [Öztürk et al., 05] pour l'estimation du délai de production, la découverte d'une tendance parmi un ensemble de données [Last et Kandel, 04] ou l'apprentissage dans les systèmes manufacturiers [Koonce et al., 97]).

Nous remarquons que, pour les deux points que nous venons de citer (relatifs à l'automatique et à la finance), l'idée est identique : donner plus d'importance aux données récentes. Or, en tenant compte de la dynamique d'un SPBS, la prise en compte des dernières décisions prises (et par conséquent des dernières performances obtenues) semble approprié dans une logique d'amélioration continue des performances. Donc nous choisissons, dans notre approche, d'adopter un facteur d'oubli dans les mécanismes d'apprentissage des entités actives. La facilité de mise en œuvre d'un facteur d'oubli (voir troisième chapitre) représente une autre justification de ce choix. Cette notion de facteur d'oubli est expliquée au travers notre exemple, voir tableau Tab. 18 :

	entité active	facteur d'oubli
exemple pris en compte	machine	analyser les 5 dernières décisions prises (source de l'apprentissage =interne)
	produit	analyser les décisions prises par les 10 derniers produits finis ayant quitté le système (source de l'apprentissage =externe)
modèle de SPBS proposé	eft	λ_f
	emp	λ_m

Tab. 18 Facteurs d'oubli dans l'exemple et dans le modèle d'entités proposé

Nous en déduisons la quatrième spécification (S4) : dans une logique d'amélioration continue des performances, un facteur d'oubli doit être pris en compte dans les mécanismes d'apprentissage afin de garantir la pertinence des décisions prises.

3.2.4. Apprendre à diversifier les décisions prises

Pour comprendre cet aspect de 'diversification', considérons en premier lieu l'exemple du système de production considéré dans ce chapitre :

Dans notre exemple, un produit peut suivre plusieurs trajectoires différentes pour réaliser sa gamme de tâches. Or, un produit est orienté 'généralement' vers la machine la moins chargée ou celle qui exécute la tâche en moins de temps (application des règles classiques en ordonnancement de type SPT). Donc le choix 'naturel' et immédiat de chaque produit consiste à opter pour le plus court chemin (avec un temps de passage égal à t_{au}). Par conséquent, la totalité des produits auront ce même 'réflexe' et le choix des trajectoires suivies sera globalement identique. Or, l'adoption d'une stratégie unique (le chemin le plus court) est optimale en conditions normales (fonctionnement normal des machines) mais s'avère pénalisant dès l'apparition d'une ou plusieurs perturbations. L'apparition d'une perturbation, en l'occurrence l'arrêt temporaire ou définitif d'une machine, provoque l'augmentation des temps de passage de chaque produit dans le système de production. La Figure 27-a illustre ce cas pendant lequel le temps passé par chaque produit augmente et dépasse largement le temps nominal (t_{na}) d'une trajectoire quelconque (non optimale) d'où une sous exploitation des capacités effectives du système. La disparition de la perturbation (réparation de la machine) permet de retrouver progressivement un fonctionnement normal et par conséquent optimiser le temps de passage de chaque produit. Nous pouvons conclure que l'apparition d'une ou plusieurs perturbations dégrade les performances globales de ce système (augmentation du temps de passage) en l'absence de toute adaptation aux conditions réelles.

Une approche plus 'intelligente' consiste à injecter aléatoirement (ou périodiquement) des entrées (certes qui vont donner des sorties moins optimales) dans le but de varier, alterner et surtout exploiter pleinement les possibilités offertes par le système. Il s'agit en effet dans cet exemple d'éviter l'adoption par tous les produits des mêmes trajectoires pour la réalisation de leur gamme. La Figure 27-b montre cette démarche qui aboutira à une amélioration des performances même en cas de perturbations.

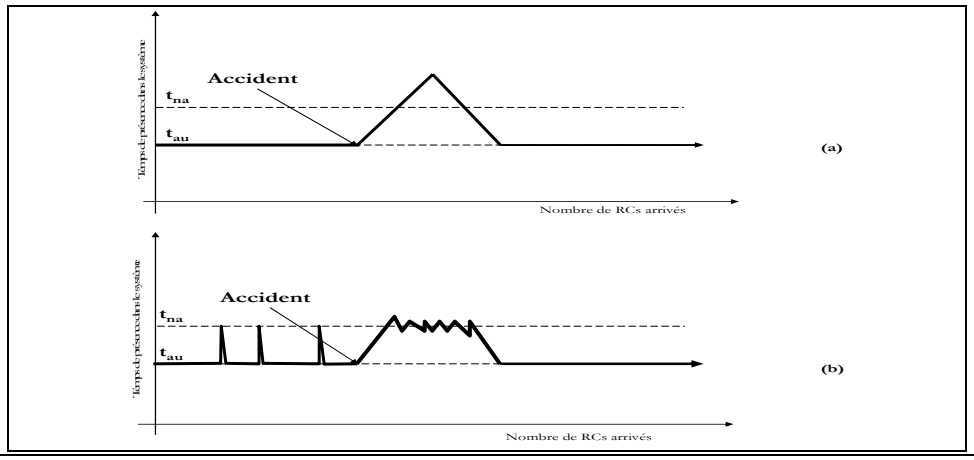


Figure 27- Fonctionnement du système : (a) sans apprentissage, (b) avec apprentissage

Une investigation des techniques dédiées ou qui peuvent résoudre le problème de la non diversification des données fait ressortir deux techniques principales : les mécanismes issus de l'évolution naturelle (sciences de la vie) et la notion de séquences binaires pseudo aléatoires (automatique). Dans les deux cas, il existe une importante composante aléatoire :

- l'évolution des systèmes vivants est caractérisée par des mécanismes de reproduction basés sur des opérations de sélection, de mutation ou de croisement. Ces mécanismes ont été imités pour donner lieu à des algorithmes dits évolutionnistes (algorithmes génétiques [Goldberg, 89], programmes évolutionnistes ([Michalewicz, 99]), programmation génétique [Koza, 92]),
- en automatique, pour bien identifier un processus, il faut appliquer une entrée 'riche' en fréquence [Landau, 93]. En pratique, la solution standard est fournie par l'utilisation des séquences binaires pseudo aléatoires (SBPA). Ces séquences représentent des successions d'impulsions rectangulaires modulées en largeur qui approximent un bruit blanc discret et donc qui ont un contenu 'riche' en fréquences. Une SBPA est caractérisée par une longueur de séquence à l'intérieur de laquelle les variations de la largeur des impulsions varient aléatoirement.

Dans notre travail nous avons opté pour les algorithmes génétiques comme moyen de diversification des données. Les algorithmes génétiques présentent plusieurs avantages (voir [Bousbia et Trentesaux, 04]) et notre choix pour cette technique est motivé par les arguments suivants :

- selon Goldberg [Goldberg, 89], les opérateurs génétiques (sélection, mutation et croisement) sont appropriés à l'exploration de l'espace de recherche et permettent l'exploitation des zones prometteuses découvertes,
- comme nous allons le montrer, l'application des opérateurs génétiques (croisement ou mutation) est possible grâce au codage et à la mémorisation des données que nous proposons (sous forme de matrices, voir le troisième chapitre),
- la montée en puissance des calculateurs (processeurs) rend la mise en œuvre de cette technique

plus abordable en terme de temps CPU[37].

Nous en déduisons la cinquième spécification (S5) : Les mécanismes d'apprentissage doivent permettre aux entités actives de diversifier leurs décisions afin d'une part améliorer les performances du SPBS et d'autre part réagir efficacement en cas de perturbations.

3.2.5. Apprendre à évaluer continuellement les décisions prises

D'abord, nous illustrons la notion d'évaluation continue des décisions prises grâce à l'exemple considéré :

Dans cet exemple, nous pouvons définir des temps de passage pour chaque trajectoire : nous distinguons entre le temps réalisé réellement et le temps nominal ou moyen (réalisé dans des conditions normales). Ces définitions permettent en conséquence d'évaluer les performances réalisées par l'ensemble des produits. En effet, une quantification simple consiste à définir un seuil qui sera fonction du temps nominal de chaque trajectoire et comparer ensuite les temps réalisés par rapport à ce seuil. Cette comparaison permet soit de conseiller une telle trajectoire soit de la déconseiller aux nouveaux produits entrants dans le système. En ce sens, les produits ayant fini la réalisation de la totalité de leur gamme peuvent afficher les temps qu'ils ont réalisés et les comparer par rapport aux temps nominaux. Prenons l'exemple d'un produit qui a opté pour une trajectoire donnée et qui a passé, dans le système, un temps t_n inférieur à t_{nn} (par exemple $t_n \leq 90\%$ (t_{nn})), la stratégie adoptée (par exemple adopter le chemin le plus long) sera conseillée et verra son évaluation augmentée. Au contraire, si un produit qui a suivi une trajectoire et qui a passé un temps supérieur à t_{na} (par exemple 150% (t_{na}) $\leq t_a$), la stratégie adoptée (par exemple adopter le chemin le plus court) sera déconseillée et verra son évaluation diminuée. Ces différentes évaluations permettront aux nouveaux produits d'adopter la bonne stratégie toujours en relation avec l'état réel du système. Une notion qui nous paraît importante est celle du facteur d'oubli. Par ailleurs, illustrons l'importance de la prise en compte d'un facteur d'oubli : pour avoir une 'image fidèle' de l'état du système, il est judicieux de ne considérer que les dernières produits finis ayant quitté le système de production afin de déterminer la bonne stratégie. La définition et la correspondance entre les indicateurs de cet exemple et les paramètres de notre modèle générique sont données dans le tableau Tab. 19 :

Modèle générique	Exemple
performance moyenne	t_{nn}, t_{an} : le temps nominal passé en suivant, respectivement, deux trajectoires différentes.
performance intermédiaire	t_n, t_a : le temps effectif passé en suivant, respectivement, chaque trajectoire.
seuil de performance (s_m)	s_{mn}, s_{ma} : seuil d'évaluation « tolérable » pour chaque trajectoire (par exemple si $t_n \in [t_{nn}-s_{mn}, t_{nn}+s_{mn}]$ ou si $t_a \in [t_{an}-s_{ma}, t_{an}+s_{ma}]$, alors nous parlons de performances acceptables (exemple s_{ma}=5 minutes et s_{mn}=10 minutes).

Tab. 19 Tableau des différents indicateurs de l'exemple choisi

Parmi les trois principales techniques d'apprentissage présentées en annexe 2, nous avons opté, dans notre approche, pour un apprentissage de type 'renforcement'[38]. Dans ce type d'apprentissage, une entité analyse en permanence les conséquences de ses décisions, en ayant une tendance à reproduire préférentiellement celles qui, dans les mêmes circonstances, ont conduit à des succès. Glorennec [Glorennec, 03] distingue deux approches principales pour la résolution d'un tel type de problème :

- l'utilisation de méthodes issues de la programmation dynamique (théorème de Bellman) en formalisant ce type de d'apprentissage par un problème de décision markovien,
- une recherche dans l'espace des comportements pour déterminer ceux qui permettent de

[37] Central Processing Unit.

[38] Cette technique d'apprentissage trouve ses origines dans les processus de décision markoviens. Or, si ces techniques d'apprentissage par renforcement s'avèrent appropriées dans certains domaines (robotique, théorie des jeux, …), nous pensons qu'elles ne sont pas ou peu adaptées dans le cas des systèmes de production de biens et de services. Autant, un robot peut effectuer un mauavais mouvement (déplacement dans une mauvais sens) autant un produit ne peut pas aller sur une machine qui n'est pas capable de le traiter. Néanmoins, nous avons exploité cette notion de retour d'expérience évaluée (récompense, pénalité) dans une logique d'amélioration continue des performances.

réaliser la tâche assignée. Cette recherche peut se faire avec des algorithmes génétiques,

Selon Whitley et al. [Whitley et al., 93] cités dans [Glorennec, 03], ces deux approches donnent en pratique des résultats comparables. Or, la première approche se base essentiellement sur des processus stochastiques. Or, nous nous plaçons dans un contexte déterministe et nous choisissons donc la deuxième approche de résolution. Nous estimons que ce choix est approprié à une logique d'amélioration continue des performances d'un SPBS. En effet, ce type d'apprentissage offre une évaluation continue des différentes décisions prises. Cette évaluation permet de 'détecter' les bonnes et les mauvaises décisions et par conséquent favoriser constamment les bonnes décisions permettant une amélioration continue des performances. En outre, l'utilisation des algorithmes génétiques est justifiée car ils permettent, grâce à la notion de 'fitness' de mettre en œuvre l'amélioration continue des performances. En effet, la théorie darwinienne stipule que les individus les plus adaptés survivent [Sebag et Schoenauer, 96] et ils auront plus de chances de devenir les parents de la génération suivante. Ainsi, dans une population d'individus qui évoluent et se reproduisent, la loi de la sélection naturelle fait émerger continuellement les individus les plus adaptés (meilleur fitness), donc les meilleures solutions,

Nous en déduisons la sixième spécification (S6) : Grâce à un processus d'évaluation continue des décisions prises, les mécanismes d'apprentissage doivent permettre au SPBS, en cas de perturbations, de retrouver rapidement ses performances.

3.2.6. Apprendre à réagir face aux perturbations

Cette spécification est liée à celles que nous venons de présenter. En effet, dans l'objectif d'une amélioration continue des performances et en présence d'un ensemble de perturbations, le système de pilotage doit, grâce à des mécanismes d'apprentissage, permettre au système de retrouver ses performances. Nous illustrons cette tolérance aux pannes par la Figure 28 sur laquelle nous remarquons que :

- lorsque une perturbation survient, les performances du système se dégradent (par exemple le temps de présence des entités dans le système augmente) en absence de mécanismes d'apprentissage adéquats,
- en présence de mécanismes d'apprentissage, le système réagit et retrouve un fonctionnement normal à chaque fois que le système est en fonctionnement perturbé.

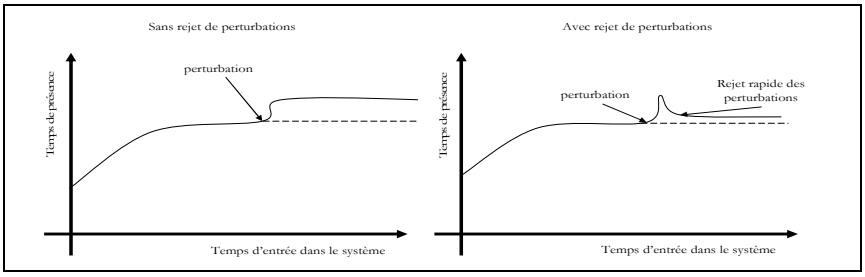


Figure 28- Rejet de perturbations

Pour récapituler, le tableau Tab. 20 fait la correspondance entre les problématiques présentées dans le chapitre I et les différentes spécifications définies dans ce chapitre :

	Spécification					
	S1	S2	S3	S4	S5	S6
Problématique(s)	P1, P2, P4, P5, P7	P3	P2, P6, P7	P2	P1, P6	P6, P7

Tab. 20 Correspondance entre les problématiques et les spécifications présentées

4. Conclusion

Dans ce chapitre nous avons présenté :

- une modélisation systémique à base d'entités d'un SPBS appropriée à la conception d'un système de pilotage hétérarchique pour l'amélioration continue des performances,
- les spécifications nécessaires d'un système de pilotage pour atteindre l'amélioration continue des performances qui sont l'autonomie et l'apprentissage :
  - nous avons spécifié le processus décisionnel de chaque entité active intégrant l'autonomie,
  - nous avons spécifié le processus d'apprentissage de chaque entité active en donnant un cadre formalisé des mécanismes d'apprentissage,

Dans le chapitre suivant, nous présentons notre contribution pour la conception d'un système de pilotage hétérarchique évolutif par apprentissage mettant en œuvre l'ensemble de ces spécifications.

CHAPITRE III : PROPOSITION D'UN SYSTEME DE PILOTAGE HETERARCHIQUE EVOLUTIF PAR APPRENTISSAGE

Le chapitre précédent nous a permis de définir un cadre générique pour la modélisation systémique des systèmes de production de biens et de services (SPBS). Nous avons également établi les spécifications d'un système de pilotage hétérarchique pour assurer l'amélioration continue des performances globales d'un SPBS et montré la nécessité de distribuer des capacités de prise de décision sur les différentes entités actives composant le système et de les doter de capacités d'évolution (apprentissage).

Notre contribution constitue une modélisation possible en réponse aux spécifications proposées.

Dans ce chapitre, nous présentons nos propositions pour concevoir un système de pilotage hétérarchique évolutif par apprentissage. Notre contribution se fonde sur l'ensemble des spécifications que nous avons présentées dans le chapitre deux. Notre approche est axée suivant trois points importants : la diversification et la pertinence des données, l'évaluation des décisions prises et la garantie d'une amélioration continue des performances. Ce chapitre est composé de deux parties : la première est consacrée aux mécanismes décisionnels des entités actives tandis que la seconde décrit les modèles d'apprentissage pour assurer l'amélioration continue des performances.

1. Notations

Les différentes notations utilisées dans le troisième chapitre sont données dans les tableaux ci-dessous :

- Notations relatives aux entités fixes de traitements :

n_f : Nombre total de cycles de décisions réalisés par une entité fixe de traitement lors de son fonctionnement.

na_f : Nombre de cycles de décisions (dont les stratégies sont générées aléatoirement) nécessaires pour initialiser le processus décisionnel d'une entité fixe de traitement.

M(eft(k,h)) : Matrice de mémorisation des données relatives au cycle de décisions d'indice h effectué par une eft(k).

$\alpha(d,n)$: Coefficient d'évaluation de la stratégie d adoptée par une eft pour prendre la décision n.

E(SEft(d)) : Vecteur d'évaluation de la stratégie d par rapport aux NbD décisions $E(SEft(d) = [\alpha(d,1),..., \alpha(d,n),..., \alpha(d,NbD)]$.

M(SEft) : Matrice d'évaluation des NSf stratégies par rapport aux NbD décisions : $M(SEft) = [E(SEft(1)), \ldots, E(SEft(d)), \ldots, E(SEft(NSf))]$.

$EMF(k,t_f)$: Ensemble des matrices M(eft(k,h)) mémorisées à l'instant t_f.

PEft(k,h,NbD) : Performance globale obtenue suite à la réalisation d'un cycle h (composé de NbD décisions) par une eft(k).

$PMEft(k,NbD,\lambda_f)$: Performance moyenne obtenue suite à la réalisation de λ_f cycles de décisions par une eft(k) ($PMEft(k,NbD,\lambda_f) = \frac{\sum_{h=NbC(k,t_f)-\lambda_f+1}^{h=NbC(k,t_f)} PEft(k,h,NbD)}{\lambda_f}$).

s_f : Seuil de dépassement tolérable nécessaire à la comparaison entre la performance obtenue par une eft (suite à la réalisation d'un cycle de décisions) et la performance moyenne obtenue pour λ_f cycles.

r_f : Signal de renforcement (récompense ou pénalité) intervenant dans les mécanismes d'apprentissage d'une eft ($r_f \in \mathbb{N}$).

$EMF(k,\lambda_f)$: Ensemble des matrices associées aux λ_f derniers cycles de décisions réalisés par une eft(k).

PIF : Population initiale de matrices prises en compte dans les mécanismes génétiques d'une eft.

PAF : Population améliorée de matrices construite grâce aux mécanismes génétiques appliqués pendant le processus décisionnel d'une eft.

m_f : Nombre des meilleures matrices (appartenant à $EMF(k,\lambda_f)$) sélectionnées et insérées dans la population initiale PAF.

p_f : Nombre des matrices (appartenant à $EMF(k,\lambda_f)$) sélectionnées aléatoirement et insérées dans la population initiale PAF.

σ_f : Taux de mutation pris en compte dans les mécanismes génétiques appliqués pendant le processus décisionnel d'une entité fixe de traitement.

croisement(,) : Fonction de croisement prenant en paramètres deux matrices (parents) et retournant deux matrices (offsprings, voir définition dans la partie réservée aux mécanismes génétiques).

mutation(,) : Fonction de mutation prenant en paramètres deux matrices (parents) et retournant une seule matrice (offspring).

Mf : Matrice retournée par la fonction croisement(,) et qui sera sélectionnée et insérée dans la population initiale PIF (mécanismes génétiques relatifs au processus décisionnel d'une eft).

Mf' : Matrice retournée par la fonction mutation(,) et qui sera insérée dans la population initiale PIF (mécanismes génétiques relatifs au processus décisionnel d'une eft).

- Notations relatives aux entités mobiles produits :

n_m : Nombre total d'entités mobiles produits traitées dans un SPBS.

na_m : Nombre d'entités mobiles produits traitées (en adoptant des stratégies générées aléatoirement) afin d'initialiser le processus décisionnel de l'ensemble des entités mobiles produits.

M(emp(i)) : Matrice de mémorisation des données relatives au processus décisionnel d'une emp(i).

$\beta(p,m)$: Coefficient d'évaluation de la stratégie p adoptée par une emp pour exécuter la tâche m.

E(SEmp(p)) : Vecteur d'évaluation de la stratégie p par rapport aux NbT tâches : $E(SEmp(p)) = [\beta(p,1), \ldots, \beta(p,m), ..., \beta(p,NbT)]$.

M(SEmp) : Matrice d'évaluation des NSm stratégies par rapport aux NbT tâches : $M(SEmp) = [E(SEmp(1)), \ldots, E(SEmp(p)), \ldots, E(SEmp(NSm))]$.

$EMM(t_m)$: Ensemble des matrices M(emp(i)), relatives aux emp finies, mémorisées à l'instant t_m.

$I(\lambda_m)$: Ensemble des indices des λ_m-dernières emp ayant quitté le système. Cet ensemble d'indices est défini car les différentes emp sortent selon un ordre différent de leur ordre d'entrée dans le SPBS.

$PMEmp(i,m,\lambda_m)$: Performance moyenne obtenue par λ_m emp finies pour une tâche donnée m, suite à l'exécution de cette tâche($PMEmp(i,m,\lambda_m) = \frac{\sum_{a \in I(\lambda_m)} PEmp(a,m)}{\lambda_m}$).

s_m : Seuil de dépassement tolérable nécessaire à la comparaison entre la performance obtenue par une emp et la performance moyenne obtenue par λ_m emp (pour chaque tâche).

r_m : Signal de renforcement (récompense ou pénalité) intervenant dans les mécanismes d'apprentissage des emp ($r_m \in \mathbb{N}$).

$EMM(t_m,\lambda_m)$: Ensemble des matrices associées aux λ_m dernières emp finies.

PIM : Population initiale de matrices prise en compte dans les mécanismes génétiques des emp.

PAM : Population améliorée de matrices construite grâce aux mécanismes génétiques appliqués pendant le processus décisionnel des emp.

m_m : Nombre des meilleures matrices (appartenant à EMM(t_m,λ_m)) sélectionnées et insérées dans la population initiale PAM.

p_m : Nombre des matrices (appartenant à EMM(t_m,λ_m)) sélectionnées aléatoirement et insérées dans la population initiale PAM.

σ_m : Taux de mutation pris en compte dans les mécanismes génétiques appliqués pendant le processus décisionnel des entités mobiles produits.

Mm : Matrice retournée par la fonction croisement(,) et qui sera sélectionnée et insérée dans la population initiale PIM (mécanismes génétiques relatifs au processus décisionnel des emp).

Mm' : Matrice retournée par la fonction mutation(,) et qui sera insérée dans la population initiale PIM (mécanismes génétiques relatifs au processus décisionnel des emp).

NPS : Nombre d'entités mobiles produits présentes simultanément dans un SPBS.

NPF : Nombre d'entités mobiles produits finis sur une période de simulation.

2. Vue globale du processus décisionnel des entités actives

Ce chapitre débute par une description générale du processus décisionnel des deux types d'entités actives. Cette description s'appuie sur un modèle établi à l'aide des réseaux de Petri (RdP) [David et Alla, 89] que nous avons utilisés en raison de leur puissance de modélisation. Nous choisissons de donner cette modélisation en début de ce chapitre afin que le lecteur s'y réfère (le rôle de chaque place ou transition sera décrit ensuite). Ce choix offre l'avantage de fournir une description globale du processus décisionnel des entités actives.

2.1. Description générale du processus décisionnel d'une entité fixe de traitement

La Figure 29 représente le modèle du processus décisionnel d'une entité fixe de traitement (noté RdP(f)). Il décrit l'enchaînement des étapes suivies par une eft qui doit réaliser n_f cycles de décisions au cours de son fonctionnement total (le marquage initiale de P0 est égal à n_f-1).

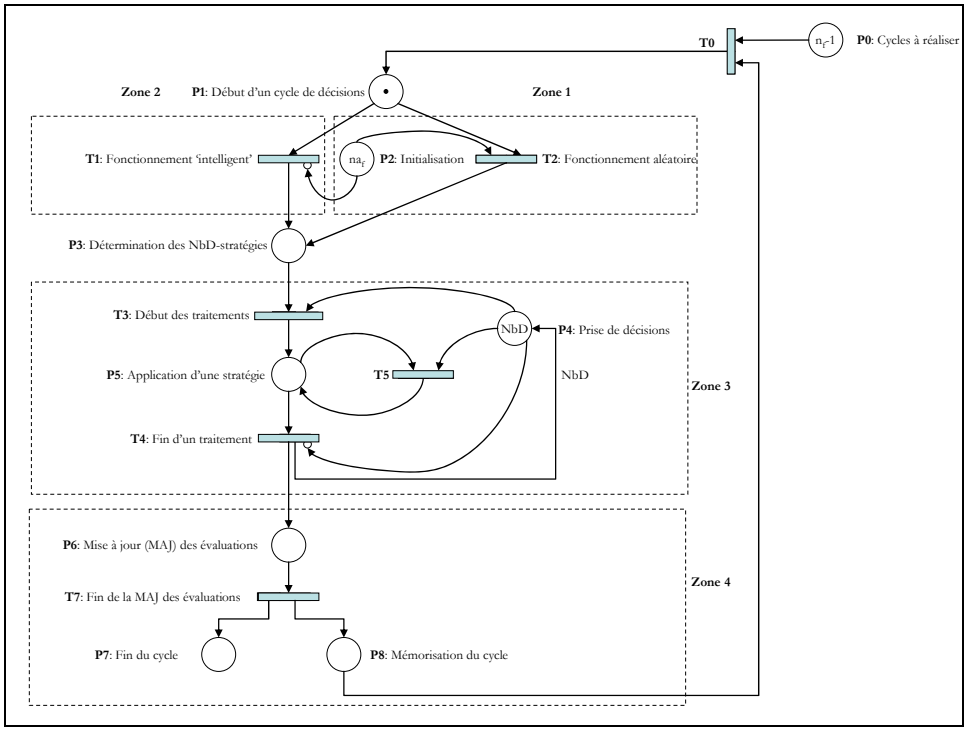


Figure 29- Modélisation du fonctionnement des entités fixes de traitements par réseaux de Petri

Le processus décisionnel est initialisé en générant, aléatoirement, les stratégies nécessaires à la réalisation d'un nombre donné (noté na_f) de cycles de décisions (voir Figure 29-zone 1). À la fin de cette phase d'initialisation du processus décisionnel, un ensemble de données est alors constitué. Ces données sont nécessaires pour activer le fonctionnement 'intelligent'[39] des eft (voir Figure 29-zone 2). Le franchissement de la transition T1 ou de T2 permet de déterminer les NbD stratégies nécessaires à la réalisation d'un cycle de décisions.

L'exécution des NbD décisions en adoptant les différentes stratégies est modélisée avec la zone 3 de ce RdP. Précisons dans ce sens que le franchissement de la transition T4 n'est possible

[39] Cette appellation est attribuée aux entités actives autonomes capables d'apprendre au cours de leur processus décisionnel.

qu'une fois chaque cycle est terminé (grâce à l'arc inhibiteur[40] P4→T4).

Les données générées par le processus de l'entité fixe de traitement sont mises à jour et mémorisées (voir Figure 29-zone 4) pour être utilisées ultérieurement dans les mécanismes décisionnels et d'apprentissage. Ces différentes étapes seront reprises et détaillées dans les parties suivantes de ce chapitre. Auparavant, nous allons présenter le processus de décision d'une entité mobile produit.

2.2. Description générale du processus décisionnel d'une entité mobile produit

Le modèle du processus décisionnel d'une entité mobile produit est donnée sur le RdP de la Figure 30 (noté RdP(m)). Nous supposons que n_m emp sont traitées pendant le fonctionnement global d'un SPBS. Ce nombre définit le marquage initial de la place P0.

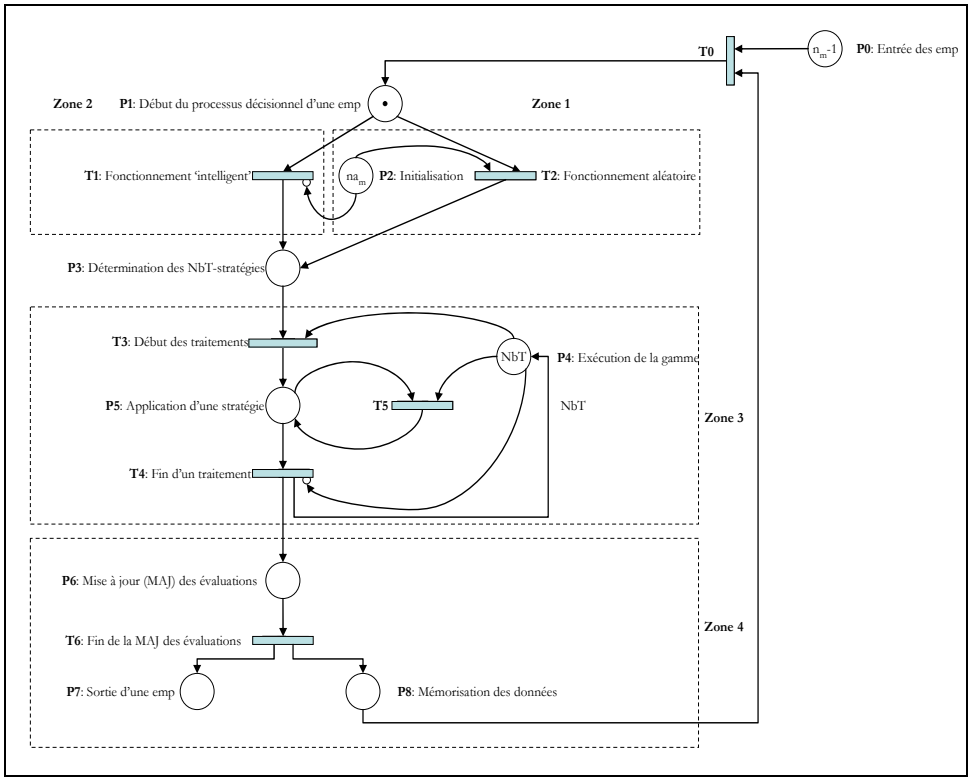


Figure 30- Modélisation du fonctionnement des entités mobiles produits par réseaux de Petri

Le processus décisionnel de l'ensemble des emp est initialisé en générant, aléatoirement, les stratégies nécessaires aux traitements de na_m emp (voir Figure 30-zone 1). En effet, la réalisation de ces traitements permet d'avoir un ensemble de données nécessaires à l'activation du fonctionnement intelligent des nouvelles emp (voir Figure 30-zone 2). Dans les deux cas, le fonctionnement aléatoire ou intelligent déterminent les NbT stratégies nécessaires à la réalisation d'une gamme relative à une emp.

L'exécution des NbT tâches d'une emp (en adoptant les différentes stratégies fournies par l'étape précédente) est modélisée grâce à la boucle représentée par la zone 3 de ce RdP.

[40] Un arc inhibiteur est un arc orienté qui part d'une place P pour aboutir à une transition t. Son extrémité est marquée par un petit cercle. L'arc inhibiteur entre la place P et la transition t signifie que la transition t n'est validée que si la place P ne contient aucune marque.

Les données générées par la prise de décisions d'une entité mobile produit sont mises à jour et mémorisées. Ces deux opérations constituent l'élément central des mécanismes d'apprentissage que nous proposons. Ces différentes étapes seront reprises et détaillées au fur et à mesure de ce chapitre.

Dans la partie suivante, nous présentons les modèles décisionnels des entités fixes de traitement (eft) et des entités mobiles produits (emp). Tout d'abord, la prise de décisions (basée sur l'adoption d'un ensemble de stratégies) de chaque type d'entité active est présentée. Ensuite, les différents mécanismes d'apprentissage relatifs à chaque type d'entités actives sont détaillés. Les différentes notions liées aux processus décisionnels présentées dans le deuxième chapitre et nos propositions sont illustrées à l'aide du même exemple concret pris en compte dans le deuxième chapitre : des ressources (des entités fixes de traitements) et des produits (entités mobiles produits) sont considérés dans cet exemple.

3. Modèle du processus décisionnel d'une entités fixe de traitement

Le processus décisionnel d'une entité fixe de traitement comprend plusieurs phases consécutives. Globalement, nous définissons ce processus par l'enchaînement des phases suivantes : prise de décisions, mémorisation des données et processus d'apprentissage. Ces trois phases interdépendantes sont développées et expliquées avec des exemples.

3.1. Modèle de prise de décision d'une entité fixe de traitement

Afin d'expliquer la prise de décision d'une eft, nous considérons un exemple dans lequel une ressource R(2) dispose d'un ensemble de quatre stratégies (ESF(2)={SEft(1),SEft(2),SEft(3), SEft(4)}, voir Figure 31). Dans la file d'attente couplée à cette ressource, il existe quatre produits en attente de traitement (EMP(2)={P5, P11, P15, P9}). Il s'agit, pour chaque produit, de réaliser une gamme de quatre tâches (dans ce cas de figure, le produit P5 a déjà réalisé 3 tâches tandis que P15 n'a réalisé aucune de ses tâches). Parmi les quatre stratégies disponibles, la ressource R(2) a sélectionné la stratégie SEft(3) (qui consiste à choisir le produit qui a effectué le moins de tâches) pour exécuter le prochain traitement. L'application de la stratégie SEft(3) permet de faire passer le produit P15 en priorité sur cette ressource, voir Figure 31 :

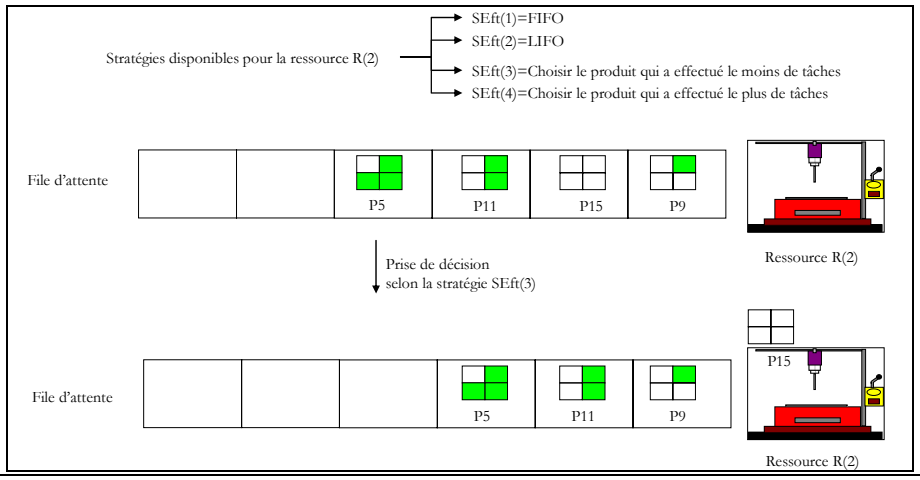


Figure 31- Exemple de prise de décision d'une ressource selon une stratégie

Plus généralement, la prise de décision d'une eft est détaillée comme suit : dans le modèle générique de SPBS que nous avons proposé (voir § 2.3), chaque eft(k) est couplée à une entité fixe de stockage efs(k). Dans cette efs, il y a un ensemble d'entités mobiles produits (noté EMP(k)) qui sont en attente pour être traitées. La prise de décision de eft(k) consiste alors à sélectionner une entité mobile produit appartenant à EMP(k) pour la traiter en priorité (voir Figure 32) :

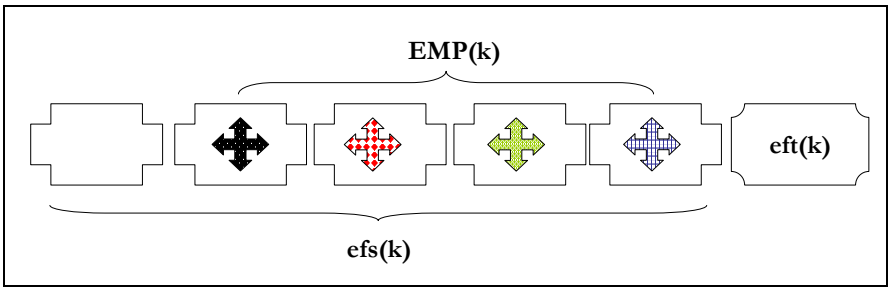


Figure 32- Situation d'une prise de décision d'une eft

Pour ce faire, une eft(k) possède un ensemble de stratégies noté ESF(k)={SEft(1), …, SEft(d), …, SEft(NSf)}. Une stratégie appartenant à cet ensemble sera adoptée pour sélectionner une entité mobile produit notée emp(*) de l'ensemble EMP(k), la Figure 33 présente les entrées-sorties du processus décisionnel d'une entité fixe de traitement :

- en entrée, une eft(k) dispose de deux ensembles ESF(k) et EMP(k),
- en sortie, une eft(k) adopte[41] une stratégie de l'ensemble ESF(k) et traite en priorité une emp(*) appartenant à l'ensemble EMP(k).

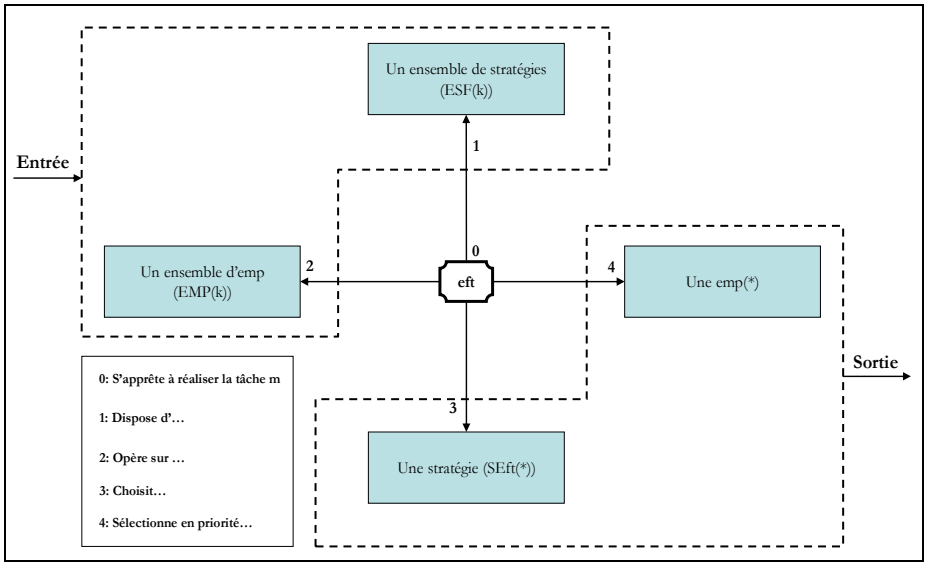


Figure 33- Modèle de prise de décision d'une eft à base d'une stratégie

En outre, nous avons vu dans le chapitre deux qu'une entité active doit mémoriser ses données pour les utiliser lors de l'élaboration de ses décisions futures dans un objectif d'amélioration continue des performances. La partie suivante traite des deux points suivants : quelles données mémoriser et comment le faire.

3.2. Modèle de mémorisation des données associées à une entité fixe de traitement

Dans notre modèle de pilotage, la prise de décision basée sur des stratégies est suivie d'une seconde étape : l'évaluation continue des stratégies adoptées. En effet, pendant le processus décisionnel d'une eft, il ne s'agit pas uniquement d'adopter une stratégie pour sélectionner une

[41] 'Le 'comment' de la sélection d'une stratégie donnée sera expliqué dans les parties suivantes.

emp et la traiter en priorité, mais aussi d'évaluer cette stratégie a posteriori[42]. Pour ce faire, nous attribuons à chaque stratégie SEft(d) un vecteur noté E(SEft(d)) qui quantifie l'évaluation de la stratégie SEft(d) suite à la prise de NbD décisions. Ce vecteur se présente sous la forme suivante : $E(SEft(d)) = [\alpha(d,1),...,\alpha(d,n),...,\alpha(d,NbD)]$[43] (avec $\alpha(d,n)$, $\alpha \in \mathbb{N}$ l'évaluation de la stratégie SEft(d) suite à la prise de la $n^{ième}$ décision). L'évolution des valeurs du E(SEft(d)) est réalisée grâce à des mises à jour successives (ces mises à jour seront détaillées dans le § 3.3.2.3). Ainsi, chaque stratégie appartenant à ESF(k) se voit attribuer un vecteur d'évaluation qui lui est propre. Cette notion d'évaluation des stratégies adoptées pendant le processus décisionnel est un élément principal des mécanismes d'apprentissage que nous proposons.

Afin d'assimiler la signification de ce vecteur, considérons l'exemple présenté précédemment. Avant d'être sélectionnée, la stratégie SEft(3) est associée au vecteur d'évaluation : E(SEft(3))= [1, 12, 5, -10, 3, 0]. Ce qui signifie que SEft(3) est bien adaptée pour la prise de la deuxième décision ($\alpha(3,2)=12$) et qu'elle est médiocre pour la quatrième décision ($\alpha(3,4)=-10$).

En ce qui concerne la procédure de mémorisation des données, nous proposons de mémoriser d'une part les données permettant de réaliser un cycle de décisions (les stratégies) et d'autre part celles générées par ce cycle. À titre de rappel, chaque eft(k) prend ses décisions d'une façon cyclique (voir § 0). Un cycle de décisions, d'indice h, (noté CEft(k,h)) est composé de NbD décisions successives. L'ensemble des données relatives à un cycle de décisions est codé et mémorisé dans une matrice (notée M(eft(k,h))) contenant deux blocs :

- le premier bloc est une matrice de dimension [2*NbD][44] dans laquelle :
  - la première ligne contient les différentes stratégies SEft(*,n) adoptées pendant un cycle de décisions, n étant l'indice d'une décision (voir Figure 34),
  - dans la deuxième ligne, chaque stratégie adoptée est évaluée grâce à un coefficient α[45] afin de quantifier son impact sur la performance globale obtenue.
- le deuxième bloc est représenté par une cellule unique qui contient une seule valeur. Il s'agit de la performance globale du SPBS (notée PEft(k,h,NbD)) obtenue pendant un cycle h (voir Figure 34). Nous avons choisi une quantification unique et globale de la performance du SPBS obtenue suite à l'application des NbD décisions afin d'évaluer l'impact d'un cycle. En effet, une performance locale (réalisée au niveau de chaque eft) est une performance constante pour tous les cycles réalisés (pour NbD décisions prises, il y a NbD emp traitées). Par contre, la performance globale réalisée par tout le SPBS peut varier d'un cycle à un autre. Cet indicateur de performance constitue un moyen adéquat pour évaluer continuellement les différentes décisions prises et par conséquent évaluer les stratégies qui sont à l'origine de ces décisions.

M(eft(k,h))=

SEft(*,1)	SEft(*,2)	…	SEft(*,n)	…	SEft(*,NbD)
α(*,1)	α(*,2)	…	α(*,n)	…	α(*,NbD)
PEft(k,h,NbD)					

Figure 34- Modèle de codage et de mémorisation des données relatives au processus décisionnel d'une eft

[42] Nous verrons par la suite à quel moment et comment une stratégie est évaluée.
[43] À t=0, tous les éléments de E (SEft (d)) sont initialisés à zéro.
[44] Deux lignes et NbD colonnes.
[45] Le processus d'évaluation des différentes stratégies sera présenté dans la partie apprentissage.

Nous utilisons le même exemple de la ressource R(2) présenté précédemment afin d'assimiler la signification de la matrice M(eft(k,h)). La Figure 35 montre la matrice relative au dixième cycle (h=10) réalisé par R(2). Ce cycle est composé de six décisions qui ont été prises par application successive des stratégies SEft(3)→SEft(1)→SEft(4)→SEft(2)→SEft(1)→SEft(3). Concernant la performance globale, nous la quantifions en mesurant le nombre de produits qui quittent le système au cours de ce cycle (PEft(2,10,6)=3, c'est-à-dire entre l'instant de la prise de la première décision et l'instant de la prise de la sixième décision, trois produits ont réalisé la totalité de leurs quatre tâches). Notons par ailleurs que pour cet exemple, la stratégie SEft(3) peut être qualifiée de 'bonne' pour la prise de la sixième décision ($\alpha(3,6)=12$) tandis que SEft(2) peut être qualifiée de 'mauvaise' pour la prise de la quatrième décision ($\alpha(2,4)=-3$).

M(Eft(2,10))=

SEft(3,1)	SEft(1,2)	SEft(4,3)	SEft(2,4)	SEft(1,5)	SEft(3,6)
-1	2	5	-3	1	12
PEft(2,10,6)=3					

Figure 35- Exemple de codage et mémorisation des données relatives à R(2)

La mise en place de la matrice M(eft(k,h)) se déroule selon les deux phases suivantes :

- pour l'ensemble des décisions prises, la première phase consiste à renseigner de proche en proche des différentes cases relatives aux stratégies adoptées SEft(*,n) (voir Figure 36-a),
- la deuxième phase est composée de deux étapes quasi simultanées :
  - en premier lieu, il s'agit de déterminer la valeur de la performance globale PEft(k,h,NbD) du cycle de décisions h et de renseigner la cellule unique réservée à cette valeur (voir Figure 36-b),
  - en second lieu, les évaluations des différentes stratégies adoptées pendant ce cycle (les coefficients $\alpha(*,n)$) sont renseignées dans les cases correspondantes (deuxième ligne de M(eft(k,h))). Nous allons voir ultérieurement que ces évaluations sont déterminées sur la base de la performance PEft(k,h,NbD) obtenue.

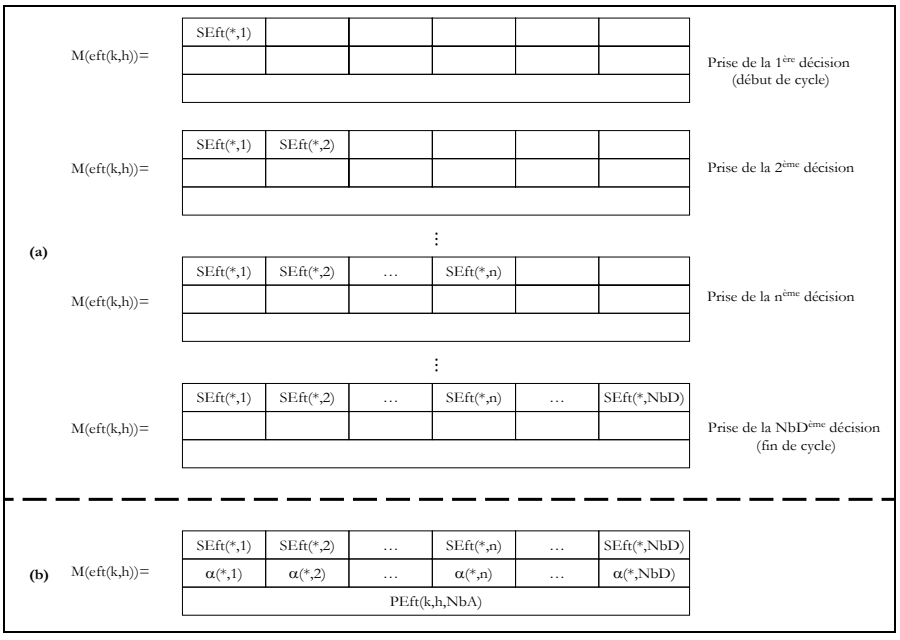


Figure 36- Modèle de renseignement d'une matrice M(eft(k,h)) associée à une entité fixe de traitement

Pour illustrer la prise de décision et la mémorisation des données d'une entité fixe de traitement, considérons la ressource R(2), qui à l'instant t_f commence son 10ème cycle composé de trois décisions. À cet instant, la file d'attente couplée à R(2) contient quatre produits (P5, P11, P15, et P9). Chaque produit est caractérisé par un degré d'avancement (nombre de tâches déjà réalisées) différent (le produit P5 a déjà réalisé deux tâches de sa gamme tandis que P15 va commencer la réalisation de sa première tâche). Pendant ce cycle de décisions, R(2) a adopté, successivement, SEft(3), SEft(4) et SEft(1) qui ont permis de sélectionner tour à tour les produits P15, P5 et P9, voir Figure 37. Notons que dans ce scénario, aucun autre produit n'entre sur la file d'attente pendant ce cycle de décisions, ce qui n'est pas vrai en réalité mais nous avons procédé ainsi uniquement à titre explicatif. Une fois ce cycle est achevé, la matrice M(eft(2,10)) est renseignée avec les différentes valeurs (les trois coefficients α et PEft(2,10,3)).

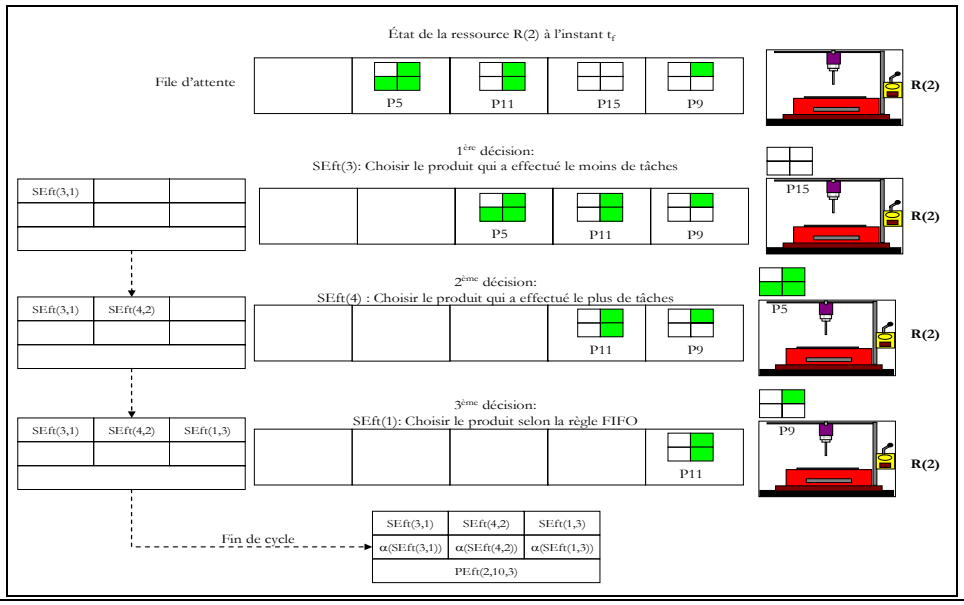


Figure 37- Exemple de renseignement de la matrice associée à la ressource R(2)

Le renseignement de la matrice M(eft(k,h)) constitue une première phase du modèle de mémorisation des données relatives au processus décisionnel d'une eft. Il s'agit dans cette phase de mémoriser les données utilisées pendant ce processus (les SEft(*,n)) et celles générées (les coefficients α et PEft(k,h,NbD)).

Cette matrice est ensuite insérée dans l'ensemble des matrices (noté EMF(k,t_f)) associées aux différents cycles de décisions réalisés auparavant : c'est la deuxième phase du modèle de mémorisation (voir Figure 38). Cette insertion est effectuée à l'instant t_f qui coïncide avec la fin d'un cycle de décisions CEft(k,h)) (dans la suite, nous montrons le rôle joué par l'ensemble de ces matrices dans le processus décisionnel des entités fixes de traitements).

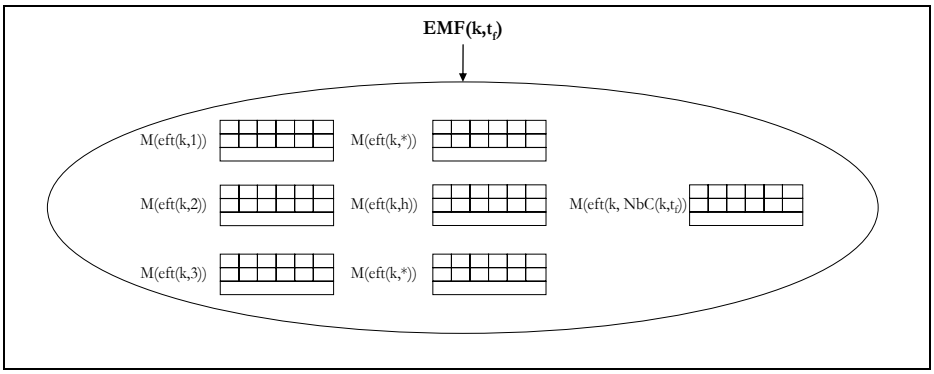


Figure 38- Ensemble des matrices associées aux cycles de décisions mémorisées à l'instant t_f

Dans le modèle de pilotage hétérarchique, les entités actives sont autonomes (prise de décisions) et capables d'apprendre (apprentissage des décisions) pendant leur fonctionnement. Or, jusqu'à maintenant, nous n'avons présenté que les mécanismes de prise de décision d'une entité fixe de traitement ainsi que la mémorisation des données nécessaires et générées par ces mécanismes. Il s'agit alors de montrer comment les différentes stratégies sont déterminées pour la réalisation de chaque cycle de décisions : c'est l'objet de la partie suivante.

3.3. Modèle d'apprentissage d'une entité fixe de traitement

Nous avons identifié, dans le chapitre deux, le processus d'apprentissage d'une entité active par le quadruplet A(EA)=<<source,état>,objet,moment>. Cette définition offre 42 cas de figures différents (cf. tableau Tab. 17). Parmi ces multiples possibilités d'apprentissage, nous choisissons de nous restreindre à une seule caractérisation de l'apprentissage pour chaque type d'entité active (la justification de ce choix est donnée dans la partie suivante).

3.3.1. Caractérisation de l'apprentissage d'une entité fixe de traitement

Dans un SPBS, les unités de production sont généralement différentes (caractéristiques techniques, capacités de traitement, …). Cette propriété nous a orienté vers une source interne d'apprentissage pour les entités fixes de traitements (c'est-à-dire chaque eft apprend de ses propres décisions). Ce choix réduit l'apprentissage d'une eft à un triplet (A(eft)=<<Interne,∅>,Objet,Moment>). Par ailleurs, nous avons montré que la prise de décision d'une eft se base sur l'adoption d'une stratégie, donc l'objet de l'apprentissage est identifié à une stratégie SEft. Enfin, les mécanismes d'apprentissage sont activés au début de chaque nouveau cycle de décisions, voir tableau Tab. 21 :

Apprentissage d'une entité fixe de traitement (eft)	
Source (S)	interne
État (E)	∅
Objet (O)	stratégie
Moment (M)	anticipé

Tab. 21 Caractérisation de l'apprentissage adopté par les entités fixes de traitements

3.3.2. Modèle d'apprentissage par renforcement d'une entités fixe de traitement

Pour une eft, la détermination des différentes stratégies nécessaires à la réalisation d'un cycle de décisions résulte des mécanismes d'apprentissage par renforcement. Nous montrons dans la partie suivante, que ce modèle d'apprentissage est une combinaison de trois modules : modules de pertinence et de diversification des données et module d'évaluation continue des décisions prises.

3.3.2.1 Module de pertinence des données

Nous avons précisé (cf. chapitre II, § 3.2.3) que dans une logique d'amélioration continue des performances, un facteur d'oubli doit être pris en compte pour assurer la pertinence des données. En outre rappelons qu'à l'instant t_f, une entité fixe de traitement d'indice k a déjà réalisé NbC(k,t_f) cycles de décisions et par conséquent a mémorisé NbC(k,t_f) matrices M(eft(k,*)). D'une part, l'exploitation de toutes ces matrices ralentira le processus décisionnel (traitement et analyse des données très longs). D'autre part, les données très anciennes ne sont pas utiles dans le sens où elles ne reflètent plus l'état courant d'une entité fixe de traitement donnée. Pour ces raisons, nous adoptons un facteur d'oubli (noté λ_f) qui définira le nombre de matrices prises en compte, à l'instant t_f, pour déterminer les différentes décisions d'un cycle. Donc, pour la réalisation de chaque nouveau cycle de décisions, seules les matrices associées aux λ_f-derniers cycles de décisions réalisés

(les matrices appartenant à un ensemble noté EMF(k,λ_f)) seront prises en compte (voir Figure 39).

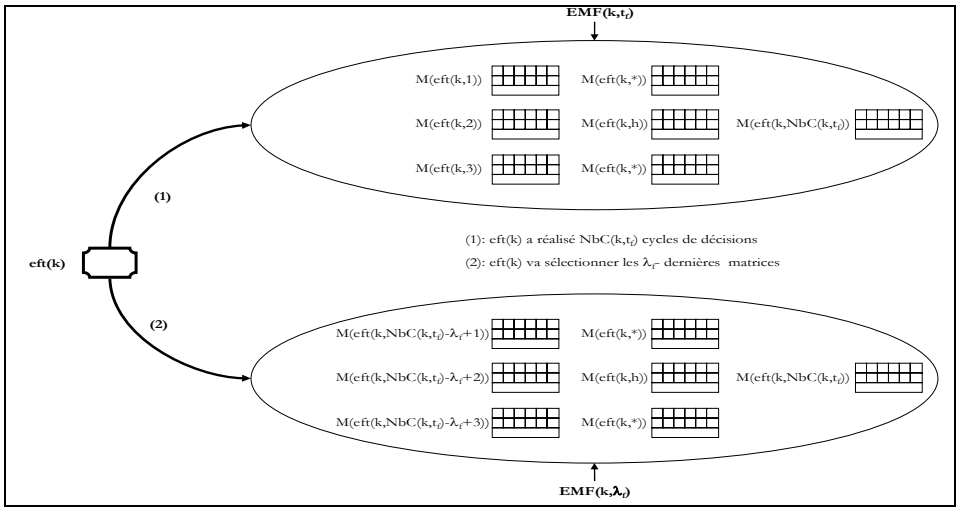


Figure 39- Sélection des cycles de décisions pour une eft

3.3.2.2 Module de diversification des données

En 1975, Holland [Holland, 75] a découvert que le processus de l'évolution naturelle décrit par Darwin pouvait être simulé pour une population d'individus où chaque individu représente une solution possible d'un problème donné : c'est le début de l'utilisation des algorithmes génétiques en tant que méthode de résolution de problèmes (en informatique, productique, …). De manière générale, les algorithmes génétiques utilisent un même principe [Sevaux, 04] : une population d'individus (correspondants à des solutions) évoluent en même temps comme dans l'évolution naturelle en biologie. Pour chacun des individus, sa faculté d'adaptation au milieu extérieur est mesurée par le 'fitness'. Les algorithmes génétiques s'appuient alors sur trois fonctionnalités :

- la sélection qui permet de favoriser les individus qui ont un meilleur fitness,
- le croisement qui combine deux solutions parents pour former deux enfants (appelés 'offsprings') en essayant de conserver les bonnes caractéristiques des solutions parents,
- la mutation qui permet d'ajouter de la diversité à la population en mutant certaines caractéristiques (gènes) d'une solution et elle est aussi nécessaire pour éviter la convergence rapide de l'algorithme génétique[46]. Dans les algorithmes génétiques, si le taux de mutation est trop élevé, les individus de la population vont toujours être modifiés, donc, il n'y aura pas de convergence vers un meilleur individu : la population va être trop hétérogène. Si le taux de mutation est trop faible, alors la population va devenir très homogène, tous les individus vont être identiques ou très proches et par conséquent la qualité de la population ne pourra plus augmenter : nous perdons alors la chance qu'une mutation nous donne un individu plus meilleur. En fonction du problème traité, il faut trouver le taux de mutation adéquat [Sevaux, 04].

Dans un algorithme génétique, la première étape consiste à générer aléatoirement une population initiale d'individus. Chaque individu est testé et se voit attribuer une valeur de fitness correspondant à sa performance. Un algorithme génétique boucle sur un cycle : [sélection des meilleurs individus, croisements stochastiques entre individus, mutation stochastique]. Le critère d'arrêt d'un algorithme génétique dépend de l'objectif recherché, le plus souvent la boucle est

[46] Problème connu sous le nom de minima locaux.

arrêtée lorsqu'une performance donnée est atteinte.

Nous adaptons les caractéristiques des algorithmes génétiques dans notre contribution en distinguant deux points essentiels : la construction d'une population d'individus et la définition d'opérateurs génétiques.

Premièrement, une population initiale (notée PIF) de matrices est construite (ces matrices sont issues de l'ensemble $EMF(k,\lambda_f)$) en tenant tcompte de deux points importants : l'optimisation (sélection des m_f meilleures matrices) et la diversification (sélection aléatoire de p_f[47] matrices). Le but est de construire une population améliorée de matrices (notée PAF) à partir de la population initiale PIF : c'est le rôle des mécanismes génétiques. En ce sens, la construction de la population améliorée passe par les trois phases suivantes :

- initialisation de la population améliorée en insérant la meilleure matrice de PIF,
- application des mécanismes génétiques (croisement et mutation) sur des matrices appartenant à la population initiale[48],
- insertion des matrices qui résultent de ces mécanismes dans PAF. L'application de ces mécanismes est répétée jusqu'à la satisfaction d'une condition d'arrêt. Dans notre approche, les opérateurs génétiques sont appliqués jusqu'à ce que nous obtenons une population améliorée contenant (m_f+p_f) matrices. L'exemple ci-dessous donne une illustration de la construction d'une population initiale de matrices :

Considérons la ressource R(2) qui, à l'instant t_f=100, a réalisé 33 cycles de décisions. Parmi les matrices associées à ces cycles, seules les cinq dernières obtenues sont sélectionnées pour subir des mécanismes génétiques (λ_f=5). Pour ce faire, la population améliorée est initialisée en insérant la matrice M(eft(2,31)) qui possède la meilleure performance (PEft(2,31,6)=6). En suite, le deux matrices M(eft(2,30)) et M(eft(2,33)) sont insérées dans PAF car elle sont associées aux deux meilleures performances (m_f=2, optimisation). Enfin, la matrice M(eft(2,34)) est choisie au hasard (p_f=1, diversification). Ainsi, la population initiale est constituée de M(eft(2,31)), M(eft(2,30)), M(eft(2,33)) et M(eft(2,34)).

Paramètres du modèle	Exemple
NbD	6
t_f	100
λ_f	5
$NbC(k,t_f)$	NbC(2,100)=34
$EMF(k,\lambda_f)$	EMF(2,5)={M(eft(2,30)),M(eft(2,31)),M(eft(2,32)),M(eft(2,33)),M(eft(2,33))}
m_f	2
p_f	1
Performances obtenues PEft(k,h,NbD)	PEft(2,30,6)=3
	PEft(2,31,6)=6
	PEft(2,32,6)=2
	PEft(2,33,6)=4
	PEft(2,34,6)=1

Tab. 22 Exemple de construction d'une population initiale de matrices

Deuxièmement, nous définissons les opérateurs génétiques qui sont appliqués sur les matrices[49] appartenant à la population initiale :

- le mécanisme de croisement opère sur deux matrices (M(eft(k,h)) et M(eft(k,h'))) et consiste à

[47] m_f et p_f sont des paramètres fixés à l'avance et nous avons Card (PIF)=m_f+p_f.

[48] Dans les mécanismes génétiques que nous avons développés, les matrices à croiser ou à muter sont choisies aléatoirement.

[49] Seules les deux premières lignes d'une matrice M (eft (k,h)) interviennent dans ces mécanisme génétiques.

définir un point et une longueur de croisement et d'interchanger localement le bloc délimité par ce point et cette longueur au niveau de chaque matrice, voir l'exemple ci-dessous. Ainsi, nous définissons une fonction croisement(.,.) qui réalise ce mécanisme génétique et qui retourne deux matrices. La matrice qui sera sélectionnée parmi les deux retournées par cette fonction est notée Mf,

- le mécanisme de mutation opère sur deux matrices (M(eft(k,h)) et M(eft(k,h'))) et consiste à 'injecter' localement une colonne issue de M(eft(k,h')) dans M(eft(k,h)). Pour ce faire, il suffit de définir un point de mutation. Dans un algorithme génétique, ce mécanisme est appliqué en fonction d'une probabilité de mutation (notée σ_f). Nous définissons une fonction mutation(.,.) qui effectue cette opération génétique et qui retourne une seule matrice notée Mf', voir Figure 40 :

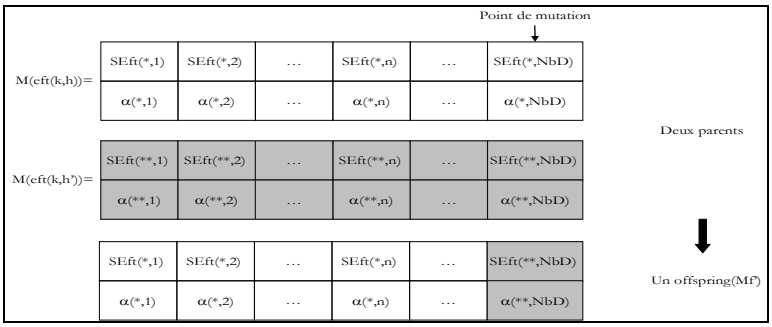


Figure 40- Opération de mutation impliquant deux matrices associées à deux cycles de décisions

Dans les algorithmes génétiques, pour évaluer (et par conséquent sélectionner) un offspring, il est nécessaire de disposer d'un indicateur (un fitness). Dans notre travail, la définition d'une matrice M(eft(k,h)) permet d'avoir deux types de fitness. D'une part, la performance de chaque cycle PEft(k,h,NbD) est prise en compte pour sélectionner les meilleures matrices (voir l'exemple plus haut) et les insérer dans la population initiale. D'autre part, après chaque opération de croisement de deux matrices, la somme $\sum_{n=1}^{NbD} \alpha(*,n)$ (relative aux évaluations des stratégies) est utilisée pour choisir une des deux matrices (offsprings) obtenues. Nous expliquons l'opérateur croisement et l'évaluation d'une matrice au travers de l'exemple suivant :

Considérons la ressource R(2) qui réalise des cycles de six décisions. La Figure 41 illustre l'opérateur de croisement et l'évaluation des offsprings qui résultent de ce mécanisme. En effet, suite au croisement des matrices M(eft(2,31)) et M(eft(2,33)), nous obtenons deux offsprings dont la fitness est égale, respectivement, à 14 et 13. En fonction de l'objectif fixé (maximiser ou minimiser la fitness), un offspring peut être choisi (dans cet exemple, c'est le deuxième offspring qui est choisi car l'objectif est de maximiser la somme des évaluations α).

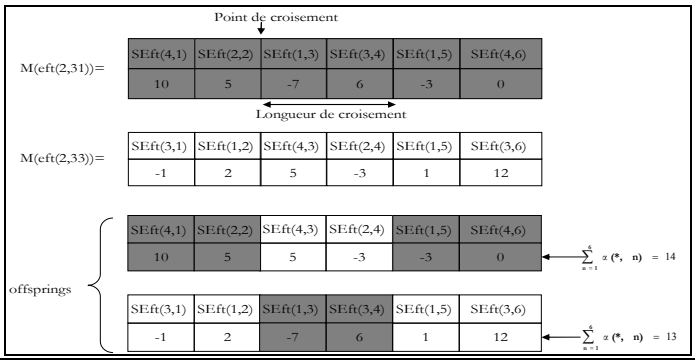


Figure 41- Exemple de croisement de deux matrices relatives à deux cycles de décisions

Pour résumer, les mécanismes génétiques que nous proposons sont composés de trois étapes et est détaillé sur la Figure 42 :

Début
Données initiales utilisées: NbC(k,t_f), m_f, p_f, λ_f
Étape 0 : Sélection des matrices prises en compte
Détermination de l'ensemble EMF(k,λ_f)
Évaluation des matrices appartenant à EMF(k,λ_f)
Étape 1 : Construction d'une population initiale (PIF)
Initialisation : insertion de la meilleure matrice de EMF(k,λ_f)
Optimisation : insertion des m_f meilleures matrices de EMF(k,λ_f)
Diversification : insertion des p_f matrices choisies aléatoirement dans EMF(k,λ_f)
Étape 2 : Construction d'une population améliorée (PAF)
Tant que (card(PAF)< (m_f+p_f))
Choisir aléatoirement deux matrices M(eft(k,h)) et M(eft(k,h')) de PIF
Mf←croisement (M(eft(k,h)),M(eft(k,h')))
Si (Probabilité de mutation > σ_f) **alors**
Choisir aléatoirement une matrice M(eft(k,h")) de PIF
Mf'←mutation(Mf,M(eft(k,h")))
Fin si
Insérer Mf' dans PAF
Fin Tant que
Fin

Figure 42- Mécanismes génétiques adoptés dans le processus décisionnel d'une eft

Dans notre approche, les mécanismes génétiques sont utilisés pour atteindre deux finalités : premièrement, assurer la diversification des données pendant le processus décisionnel d'une eft et deuxièmement la détermination des NbD stratégies nécessaires à la réalisation d'un cycle de décisions. En effet, l'ensemble de ces stratégies sont fournies grâce aux mécanismes génétiques : la meilleure matrice (la matrice qui maximise $\sum_{n=1}^{NbD} \alpha(*,n)$) de la population améliorée est sélectionnée et ses différentes stratégies sont adoptées pour la réalisation d'un nouveau cycle.

3.3.2.3 Module d'évaluation continue des décisions prises

L'apprentissage par renforcement est basé sur la notion de récompense et de pénalité (voir la présentation de cette technique d'apprentissage en annexe 2). Nous adoptons ce principe pour notre module d'évaluation continue des décisions prises. En effet, la stratégie qui donnera une 'bonne performance' se voit récompensée. A contrario, une stratégie qui produira une 'mauvaise performance' est alors pénalisée. Dans ce sens, la récompense ou la pénalisation d'une stratégie est effectuée en modifiant son coefficient d'évaluation (α). Appliqué à notre modèle, ce principe de renforcement nécessite la définition de deux paramètres :

- un signal (ou un scalaire) noté r_f qui est défini pour quantifier le processus de renforcement. En cas de récompense, ce signal est additionné au coefficient d'évaluation initial[50] d'une stratégie. En cas de pénalité, la valeur de r_f est soustraite du coefficient d'évaluation initial (la nouvelle valeur prise par le coefficient d'évaluation est notée α'),
- un seuil de performance noté s_f qui permet de définir un 'intervalle de tolérance' de la performance réalisée PEft(k,h,NbD). La détermination de cet intervalle dépend aussi de la performance moyenne notée PMEft(k,NbD,λ_f), voir illustration Figure 43. PMEft(k,NbD,λ_f)

[50] Le coefficient initial est le coefficient de l'évaluation de la stratégie avant son adoption dans un cycle de décisions.

représente la moyenne des performances calculée pour les λ_f derniers cycles de décisions. Autrement dit, selon l'appartenance de la performance réalisée à une plage de valeurs, les coefficients d'évaluation des NbD stratégies (rappelons que l'adoption de ces stratégie est à l'origine de cette performance) sont soit récompensés, soit pénalisés ou gardés inchangés. D'où, la définition de cet intervalle permet d'identifier un troisième cas, autre que la récompense ou la pénalité, dans lequel le coefficient d'évaluation initial reste inchangé.

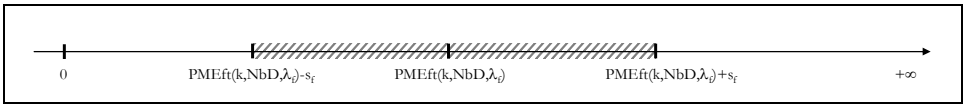


Figure 43- Valeurs tolérables de la performance réalisée suite à la réalisation d'un cycle de décisions

Signalons que dans le cas de l'apprentissage par renforcement d'une entité fixe de traitement, nous retrouvons un problème connu dans ce type d'apprentissage. Il s'agit du 'credit assignment problem' (voir [Glorennec, 03]). En effet, il difficile (voir impossible), dans certains cas, d'imputer l'effet d'une décision prise sur la performance obtenue. Dans notre cas, il est impossible de savoir quelle(s) stratégie(s) a (ont) permis d'obtenir une bonne (ou une mauvaise) performance. Pour contourner ce problème, nous choisissons la solution suivante : toutes les stratégies adoptées dans un cycles de décisions sont récompensées (respectivement pénalisées) si elles produisent de 'bonnes performances' (respectivement si elles produisent de 'mauvaises performances'). Il s'agit alors d'une récompense (ou une pénalisation) 'collective' de ces stratégies.

Ainsi, une fois un cycle de décisions h achevé, l'évaluation a posteriori des décisions prises est réalisée selon l'algorithme donné dans la Figure 44. La présentation de cet algorithme est suivie d'un exemple illustratif.

Données initiales : t_f, NbC(k,t_f), λ_f s_f, r_f, PEft(k,h,NbD) et $PMEft(k, NbD, \lambda_f) = \frac{\sum_{h=NbC(k,t_f)-\lambda_f+1}^{h=NbC(k,t_f)} PEft(k, h, NbD)}{\lambda_f}$

Si PEft(k,h,NbD)∈ [0, PMEft(k,NbD,λ_f)-s_f [alors
  Pour n=1 jusqu'à n=NbD
    Toutes les SEft(*[51],n) sont récompensées
    α'(*,n)← α(*,n)+r_f
  Fin pour
Fin Si
Si PEft(k,h,NbD)∈ [PMEft(k,NbD,λ_f)-s_f , PMEft(k,NbD,λ_f)+s_f] alors
  Pour n=1 jusqu'à n=NbD
    Toutes les SEft(*,n) ne sont ni récompensées ni pénalisées
    α'(*,n)← α(*,n)
  Fin pour
Fin Si
Si PEft(k,h,NbD)∈]PMEft(k,NbD,λ_f)+s_f , +∞ [alors
  Pour n=1 jusqu'à n=NbD
    Toutes les SEft(*,n) sont pénalisées
    α'(*,n)← α(*,n)-r_f
  Fin pour
Fin Si
E'(SEft(d,NbD)) remplace E(SEft(d,NbD))

Figure 44- Évaluation des stratégies relatives à une eft

Afin d'illustrer la notion du signal de renforcement, nous réutilisons les mêmes données présentées dans le tableau

[51] Le signe (*) remplace l'indice de la stratégie adoptée pour prendre la n[ème] décision.

Tab. 22. Ces données ont été utilisées pour déterminer le 35ème cycle de la ressource R(2). Premièrement, les matrices associées aux cinq (λ_f=5) derniers cycles de décisions réalisés sont prises en compte pour effectuer les mécanismes génétiques. La performance moyenne de ces cinq matrices vaut 3.2, voir tableau Tab. 23 :

Paramètres du modèle	Exemple
r_f	1
s_f	2
Performance obtenue à $t_f+\Delta t$ (PEft(k,h,NbD))	PEft(2,35,6)=6
Performance moyenne PMEft(k,NbD,λ_f)	$\frac{\sum_{h=NbC(k,t_f)-\lambda_f+1}^{h=NbC(k,t_f)} PEft(k,h,NbD)}{\lambda_f} = (3+6+2+4+1/5) = 3.2$
[PMEft(k,NbD,λ_f)-s_f,PMEft(k,NbD,λ_f)+s_f]	[1.2,5.2]

Tab. 23 Exemple de comparaison entre la performance moyenne et la performance réalisée

Deuxièmement, pour réaliser le cycle CEft(2,35), nous supposons que la ressource R(2) a adopté, successivement, les stratégies SEft(3), SEft(1), SEft(4), SEft(2), SEft(1) et SEft(3) pour réaliser les six décisions. À la fin de la réalisation de ce cycle (c'est-à-dire à l'instant $t_f+\Delta t$, Δt étant la durée de réalisation de ce cycle), la performance obtenue est égale à six. Cette valeur est jugée bonne car elle est supérieure à la somme de la performance moyenne et du seuil s_f (6>5.2). Ainsi, toutes les stratégies adoptées pendant ce cycle sont récompensées, c'est-à-dire que leur évaluation (par rapport à chaque décision) est augmentée de 1 (r_f=1). Sur la Figure 45-a, nous avons les différentes stratégies et leur vecteur d'évaluation initial (avant leur utilisation). Une fois le cycle de décisions achevé, l'évaluation de chaque stratégie adoptée est mise à jour, voir Figure 45-b. Par exemple, les coefficients d'évaluation de la stratégie SEft(3) par rapport à la première et à la dernière décision sont augmentés de 1 car cette stratégie a été adoptée pour la prise de ces deux décisions et la nouvelle valeur de α(3,1) (respectivement de α(3,6)) est égale à -1 (respectivement -5). Notons enfin que les valeurs changées sont soulignées dans chaque vecteur.

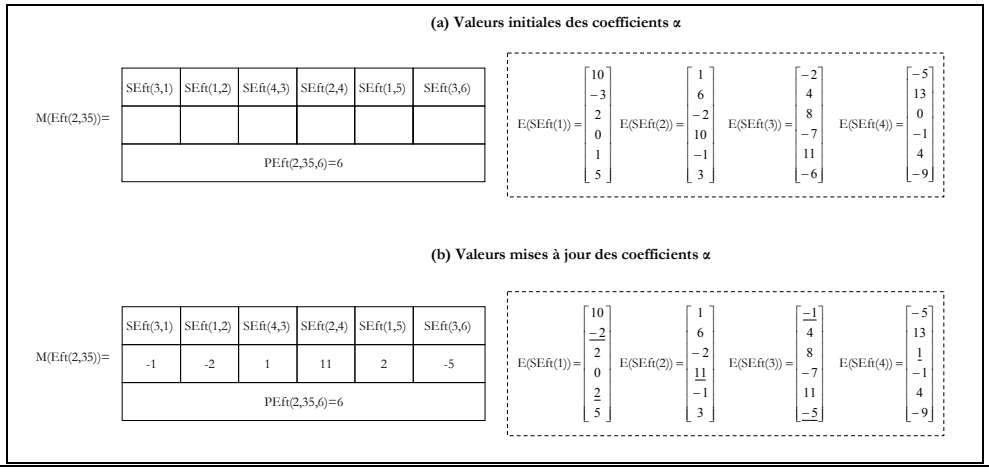


Figure 45- Exemple de mise à jour des coefficients d'évaluation des stratégies (SEft)

Les mécanismes d'évaluation des différentes stratégies constituent la dernière étape du processus décisionnel d'une entité fixe de traitement. Il est judicieux alors de résumer les différentes étapes de ce processus. Pour ce faire, nous dressons un schéma récapitulatif (voir Figure 46) comportant deux blocs :

- le premier bloc (Figure 46-I) représente la phase d'initialisation du processus décisionnel. Pendant cette phase, chaque eft génère aléatoirement les stratégies nécessaires à la réalisation de na_f cycles de décisions. L'exécution de ces cycles permet l'obtention d'un ensemble de matrices indispensable pour démarrer la deuxième phase,
- le deuxième bloc (Figure 46-II) représente la phase du fonctionnement intelligent d'une entité fixe de traitement (mécanismes génétiques). Ce fonctionnement est activé au début de chaque

nouveau cycle et se base sur trois étapes successives :

- la détermination des NbD stratégies nécessaires à la réalisation d'un cycle de décisions grâce aux mécanismes génétiques décrits précédemment,
- l'adoption de ces NbD stratégies qui permet d'obtenir une performance globale PEft(k,h,NbD),
- la mise à jour des coefficients d'évaluation des NbD stratégies adoptées en comparant la performance globale obtenue avec la performance moyenne calculée sur un ensemble de λ_f cycles de décisions.

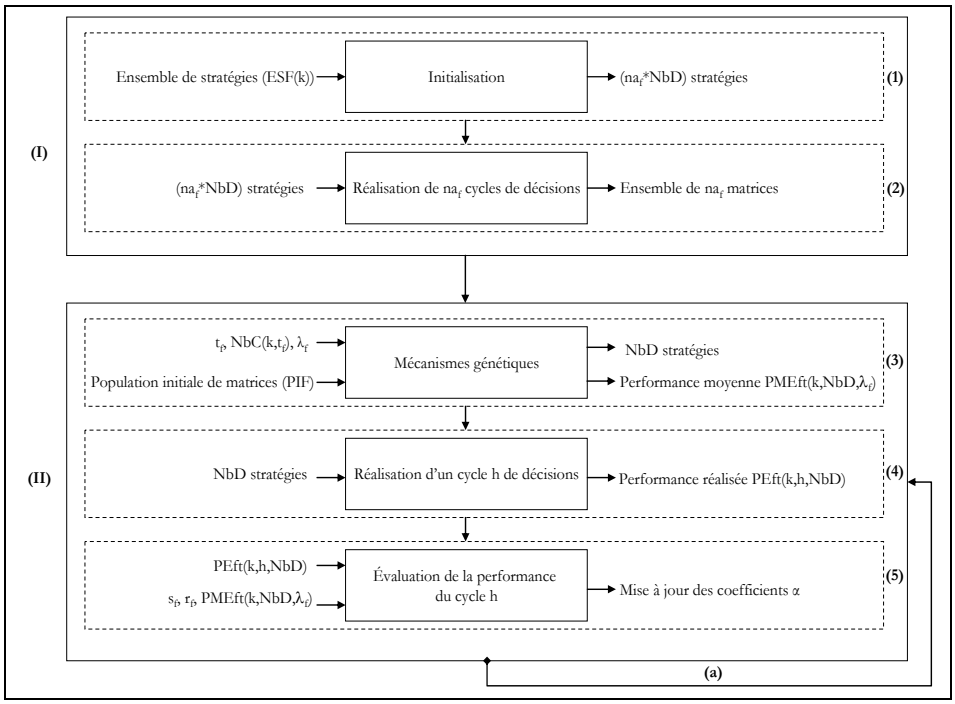


Figure 46- Récapitulatif du processus décisionnel global d'une entité fixe de traitement

Ainsi, il est possible maintenant de faire la correspondance entre le schéma de la Figure 46 et le RdP présenté sur la Figure 29. En effet, nous faisons la liaison entre les blocs et les boucles de chaque représentation, voir tableau Tab. 24 :

		RdP(f), Figure 29
Schéma, Figure 46	Bloc(1)	Zone 1, P3
	Bloc(2)	Zone 3
	Bloc(3)	P3
	Bloc(4)	P5, T5
	Bloc(5)	P6, T6
	Boucle(a)	Zone 3, P6, T6
	Bloc(I)	Zone 1
	Bloc(II)	Zone 2

Tab. 24 Correspondance entre le schéma de la Figure 46 et le RdP(f)

Nous détaillons dans la partie suivante le processus décisionnel d'une entité mobile produit. D'abord, nous présentons le mécanisme de prise de décision d'une emp par adoption de stratégies, ensuite les différents mécanismes d'apprentissage proposés.

4. Modèle du processus décisionnel d'une entité mobile produit

Le processus décisionnel d'une entité mobile produit comporte un ensemble d'étapes successives. Ces étapes permettent de mettre en œuvre deux propriétés essentielles : une prise de décision localisée au niveau d'une emp (autonomie) et une capacité de chaque emp à évoluer en fonction du contexte dans lequel elle opère (apprentissage). Ces étapes sont présentées et argumentées grâce à des exemples dans les parties suivantes.

4.1. Modèle de prise de décision d'une entité mobile produit

Nous introduisons cette section avec un exemple comportant un produit Pi et un ensemble de quatre ressources (R(1), R(2), R(3) et R(4)). À t_m=0, Ce produit se trouve sur le point d'entrée du système et doit réaliser une gamme de quatre tâches (T1→T2→T3→T4). Pour ce faire, deux ressources sont candidates à la réalisation de la première tâche (EFT(1)={R(1),R(3)}. Les temps nécessaires pour transférer ce produit sur ces deux ressources sont égaux, respectivement, à 2 et 4 unités de temps (UT). En outre, Pi dispose de quatre stratégies (ESM={SEmp(1),SEmp(2),SEmp(3),SEmp(4)}). Pour l'exécution de la première tâche, ce produit adopte la stratégie SEmp(1) (aller sur la ressource la plus proche). L'application de cette stratégie oriente Pi vers la ressource R(1), voir Figure 47 :

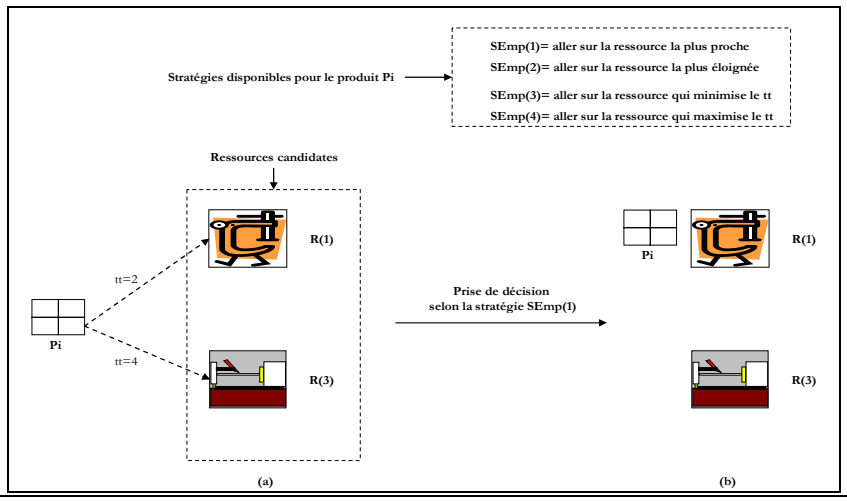


Figure 47- Exemple de prise de décision d'un produit à partir d'une stratégie

Pour généraliser ce mode de fonctionnement, considérons une entité mobile produit emp(i) qui doit réaliser une tâche donnée m. Or, plusieurs eft sont capables d'exécuter cette tâche : ce sont les entités fixes de traitements candidates à l'exécution de la tâche m (ces eft appartiennent à l'ensemble EFT(m) spécifié dans le chapitre deux). La Figure 48 illustre un EFT(m) composé de trois eft candidates eft(k), eft(k') et eft(k") :

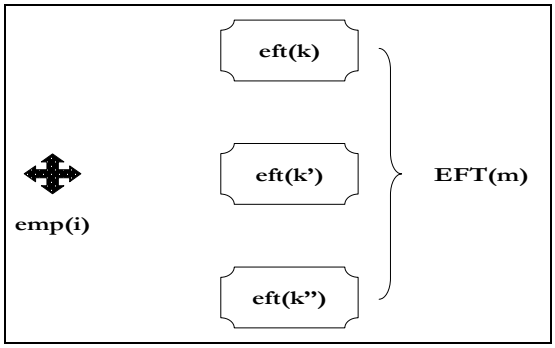


Figure 48- Situation de prise de décision d'une entité mobile produit

Parmi ces eft candidates, emp(i) doit choisir une eft (notée eft(*)) qui est sélectionnée pour exécuter la tâche m. Cette sélection est réalisée comme suit : chaque emp dispose d'un ensemble de

stratégies noté ESM={SEmp(1), …, SEmp(p), …, SEft(NSm)} et pour sélectionner eft(*), emp(i) adopte une stratégie SEmp(*,m) appartenant à l'ensemble ESM. Ce choix détermine sur quelle entité fixe de traitement la tâche m va être exécutée. La Figure 49définit les entrées-sorties de la fonction de prise de décision d'une entité mobile produit :

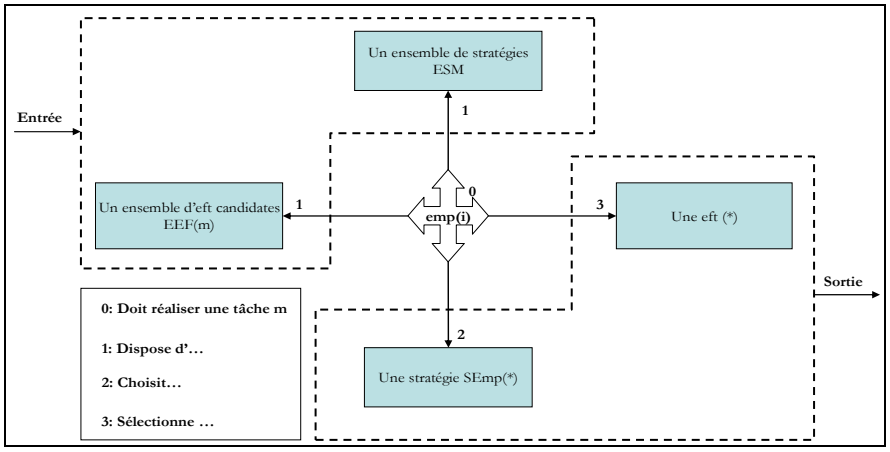


Figure 49- Modèle de prise de décision d'une emp à base d'une stratégie

Par ailleurs, le processus décisionnel d'une emp nécessite et génère un ensemble de données qui seront mémorisées. Dans la partie suivante nous détaillons les deux points suivants : d'abord quelles sont les données qui sont mémorisées et ensuite comment nous mettons en œuvre cette mémorisation.

4.2. Modèle de mémorisation des données associées à une entité mobile produit

Les données mémorisées pendant le processus décisionnel d'une emp sont de deux types : les données qui ont permis à une emp de prendre ses décisions et les données qui résultent de ces prises de décisions. Pour le premier type de données, il s'agit des différentes stratégies adoptées permettant l'orientation d'une emp vers un ensemble d'entités fixes de traitements. Le second type de données représente une quantification des décisions prises. Cette quantification est réalisée par rapport à l'exécution de chaque tâche. Autrement dit, la réalisation d'une tâche m par une emp(i) est évaluée grâce à un indicateur de performance noté PEmp(i,m). En se basant sur cet indicateur, la stratégie adoptée pour la réalisation de la tâche m est évaluée grâce à un coefficient noté $\beta(*,m)$, $\beta \in \mathbb{N}$. Ainsi, nous définissons, pour chaque stratégie appartenant à ESM, un vecteur noté E(SEmp(p)) de dimension NbT et qui est défini sous la forme suivante : $E(SEmp(p)) = [\beta(p,1), \ldots, \beta(p,m), \ldots, \beta(p,NbT)]$ [52]. Ce vecteur représente l'évaluation de l'adoption de la stratégie p pour l'exécution de l'ensemble des NbT tâches[53].

Ces deux types de données sont codées dans une matrice notée M(emp(i)) de dimension [3*NbT] qui présente les caractéristiques suivantes :

- la première ligne contient les différentes stratégies SEmp(*,m) adoptées pour la réalisation des NbT tâches,
- la deuxième ligne contient les performances intermédiaires PEmp(i,m) obtenues lors de la réalisation de chaque tâche,

[52] À t=0, tous les éléments de E (SEmp (p)) sont initialisés à zéro.
[53] L'illustration du vecteur E (SEmp (p)) suit le même principe que l'exemple donné dans le § .

- la troisième ligne contient les évaluations (les coefficients β) des différentes stratégies adoptées lors du processus décisionnel d'une emp, voir Figure 50 :

M(emp(i))=	SEmp(*,1)	SEmp(*,2)	…	SEmp(*,m)	…	SEmp(*,NbT)
	PEmp(i,1)	PEmp(i,2)	…	PEmp(i,m)	…	PEmp(i,NbT)
	β(*,1)	β(*,2)	…	β(*,m)	…	β(*,NbT)

Figure 50- Modèle de codage et de mémorisation des données relatives à une emp

Pour illustrer cette notion de matrice de mémorisation des données, considérons le produit P100 (voir Figure 51). Pour ce produit, nous supposons que les stratégies d'indice 2, 4, 1 et 2 ont été adoptées successivement pour réaliser les quatre tâches. Cette illustration met l'accent sur une propriété importante des mécanismes d'évaluation que nous proposons. En effet, dans cet exemple, la stratégie SEmp(2) a été adoptée deux fois (pour réaliser la première et la quatrième tâche) mais son évaluation est différente pour chaque tâche. Cette stratégie s'avère excellente pour la réalisation de la première tâche (β(2,1)=12) et médiocre pour la réalisation de la dernière tâche (β(2,4)=-6).

	SEmp(2,1)	SEmp(4,2)	SEmp(1,3)	SEmp(2,4)
M(emp(100))=	2	5	12	1
	12	10	0	-6

Figure 51- Exemple de codage de la matrice M(emp(100))

Le renseignement de la matrice M(emp(i)) relative au processus décisionnel d'une entité mobile produit d'indice i se déroule en deux phases :

- en premier lieu, les différentes stratégies adoptées ainsi que les performances intermédiaires obtenues sont renseignées dans la matrice M(emp(i)). Cette phase est composée de NbT étapes successives. Précisons que dans l'étape m, SEmp(*,m) et PEmp(i,m-1)[54] sont renseignées dans cette matrice, voir Figure 52-a,

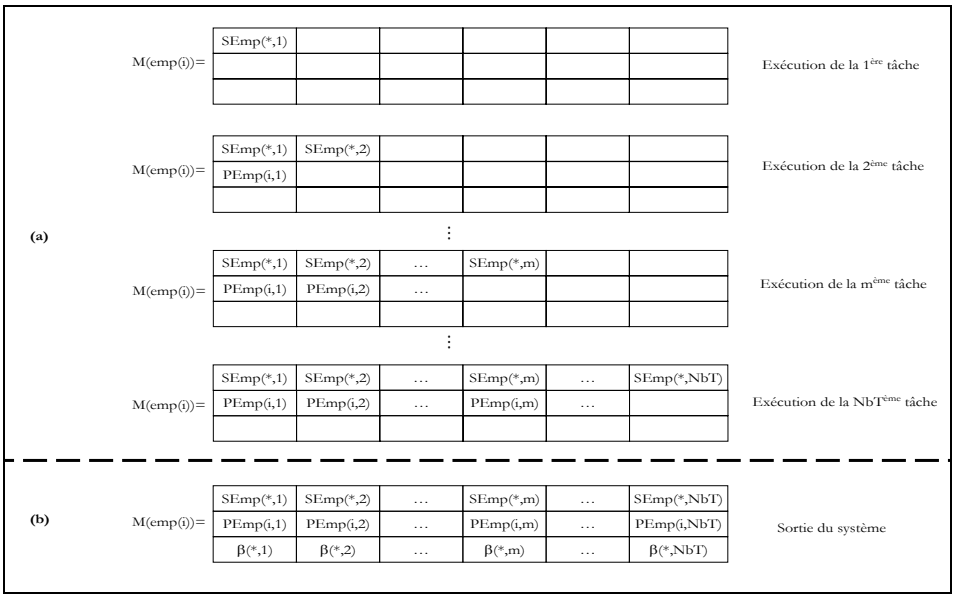
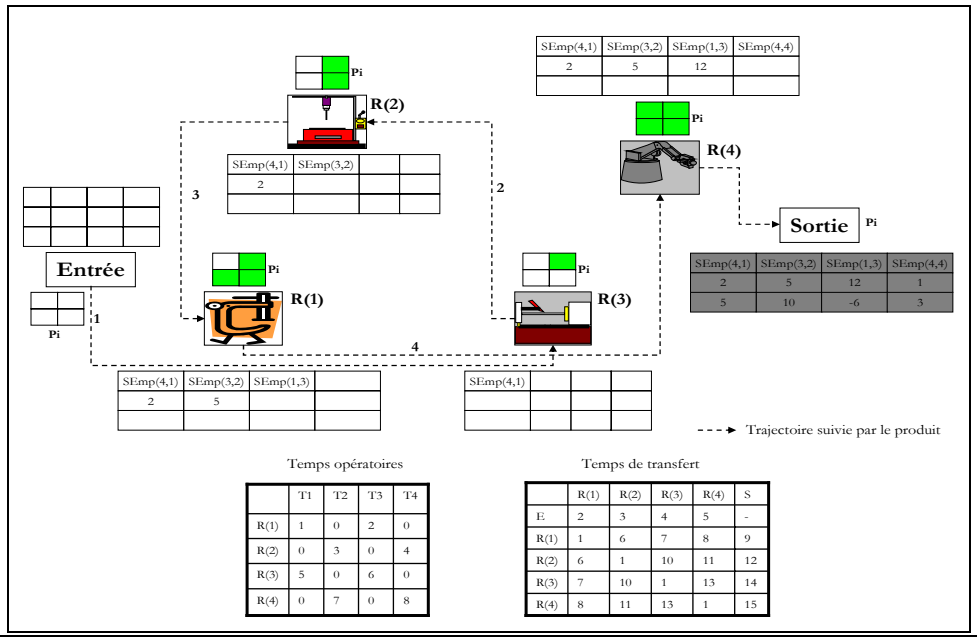


Figure 52- Modèle de renseignement de la matrice M(emp(i)) associée à une emp

[54] C'est uniquement lorsqu'une emp (i) adopte une stratégie pour réaliser la tâche m, que la performance réalisée pour la tâche précédente (m-1) est assignée dans M (emp (i)).

- en second lieu, la troisième ligne de M(emp(i)) est renseignée avec les coefficients β(*,m) lorsque emp(i) a réalisée la totalité de ses NbT tâches. Par conséquent, la totalité de la matrice M(emp(i)) est renseignée une fois que emp(i) quitté le système, voir Figure 52-b.

Considérons le produit Pi présenté précédemment : dans cet exemple, chaque machine est capable de réaliser deux tâches différentes (par exemple la machine R(3) peut exécuter les tâches T1 et T3 respectivement en 5 et 6 UTs). À l'entrée, ce produit est associé à une matrice (3 lignes, 4 colonnes) vide. Pour exécuter les quatre tâches, nous supposons que ce produit adopte successivement les stratégies SEmp(4), SEmp(3), SEmp(1), et SEmp(4). La trajectoire suivie par Pi ainsi que le renseignement de la matrice de mémorisation des données sont présentés sur la Figure 53 :



	T1	T2	T3	T4
R(1)	1	0	2	0
R(2)	0	3	0	4
R(3)	5	0	6	0
R(4)	0	7	0	8

	R(1)	R(2)	R(3)	R(4)	S
E	2	3	4	5	-
R(1)	1	6	7	8	9
R(2)	6	1	10	11	12
R(3)	7	10	1	13	14
R(4)	8	11	13	1	15

Figure 53- Exemple de renseignement de la matrice M(emp(i)) associée à un produit Pi

Le renseignement de la matrice M(emp(i)) constitue une première étape dans la mémorisation des données relatives aux entités mobiles produits. La deuxième étape consiste à insérer cette matrice M(emp(i)) dans l'ensemble des matrices (noté EMM(t_m)) déjà mémorisées, voir Figure 54. Ainsi, à l'instant t_m NbM(i,t_m) matrices sont déjà mémorisées dans l'ensemble EMM(t_m). Nous verrons par la suite que cet ensemble joue un rôle important dans le processus d'apprentissage des entités mobiles produits.

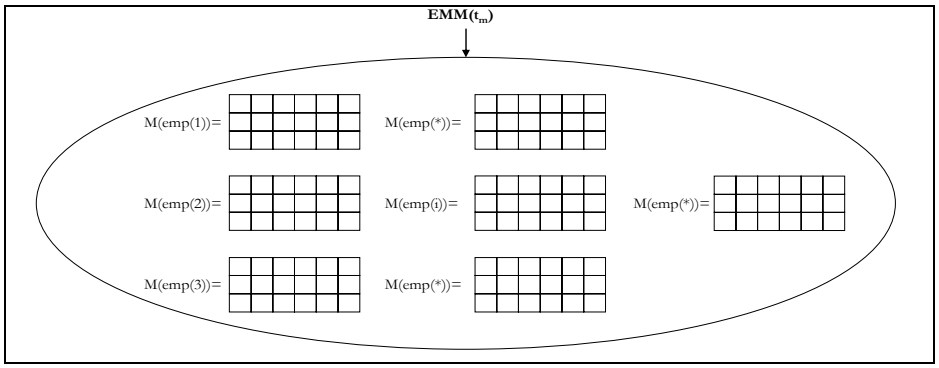


Figure 54- Ensemble des matrices mémorisées à l'instant t_m

Notons enfin que la mémorisation des données, telle que nous la proposons, diffère pour chaque type d'entités actives. Autant, la forme (matrice) de cette mémorisation est semblable pour

les emp et les eft, autant l'exploitation des matrices mémorisées est différente. Force est de constater que, pour une entité mobile produit, les matrices M(emp(i)) sont mémorisées d'une façon globale dans le SPBS (c'est-à-dire que toutes les emp peuvent accéder à ces matrices). A contrario, chaque entité fixe de traitement mémorise ses données localement. Dans ce cas, nous avons un 'partage de données' dans le processus décisionnel des entités mobiles produits et que chaque eft se base sur ses 'propres données' pendant son processus décisionnel. Le tableau Tab. 25 récapitule cette distinction :

entité active	mémorisation/exploitation des données
entité fixe de traitement	mémorisation locale/ données privées
entité mobile produit	mémorisation globale/ données partagées

Tab. 25 Typologie de la mémorisation des données selon l'entité active

Outre la proposition de ce mécanisme de prise de décision et de mémorisation des données, nous intégrons des capacités d'apprentissage au niveau de chaque entité mobile produit. La partie suivante est consacrée à la présentation de ces mécanismes.

4.3. Modèle d'apprentissage d'une entité mobile produit

Le processus d'apprentissage d'une entité mobile produit été défini dans le chapitre deux par le quadruplet A(EA)=<<source,état>,objet,moment>. Ces quatre éléments sont identifiés dans la partie suivante.

4.3.1. Caractérisation de l'apprentissage d'une entité mobile produit

Pour les entités mobiles produits, nous avons émis l'hypothèse dans notre travail que chaque emp doit réaliser la même gamme de tâches. Intuitivement, cette hypothèse justifie l'adoption d'une source externe pour l'apprentissage puisque toutes les emp sont identiques. En d'autres termes, chaque emp conçoit ses mécanismes d'apprentissage en se basant sur les décisions prises par les autres emp. Plus précisément, ce sont les emp ayant fini leur gamme qui sont prises en compte lors du processus d'apprentissage. Par ailleurs, les décisions apprises sont des décisions construites dynamiquement (objet=stratégie). Enfin les décisions sont apprises à l'instant t_m (date d'entrée d'une emp dans le système), voir tableau Tab. 26 :

Apprentissage d'une entité mobile produit (emp)	
Source (S)	externe
État (E)	décalé
Objet (O)	stratégie
Moment (M)	anticipé

Tab. 26 Caractérisation de l'apprentissage adopté par les entités mobiles produits

4.3.2. Modèle d'apprentissage par renforcement d'une entité mobile produit

Les arguments présentés dans le § 3.3.1 justifient aussi l'utilisation d'un apprentissage par renforcement pour les entités mobiles produits. En ce sens, les mécanismes d'apprentissage que nous proposons mettent en relations trois composantes essentielles : des données pertinentes (facteur d'oubli), des données diversifiées (mécanismes génétiques) et des données continuellement évaluées (signal de renforcement).

4.3.2.1 Module de pertinence des données

À l'instant t_m (rappelons que c'est l'instant de prise de la première décision) une entité mobile produit entrant dans le SPBS dispose de $NbM(i,t_m)$ matrices relatives aux différentes emp ayant fini

leurs traitements. La prise en compte de la totalité de ces matrices n'est pas adéquate pour l'élaboration des décisions de la nouvelle emp. D'une part, l'analyse et l'exploitation de toutes les matrices peuvent consommer un temps très long. D'autres part, la majorité de ces matrices ont été obtenues dans un contexte différent de celui caractérisant le SPBS à l'instant t_m. Donc, nous considérons un facteur d'oubli et seules les matrices associées aux λ_m dernières emp finies (ces matrices appartiennent à un ensemble noté $EMM(t_m,\lambda_m)$) seront prises en compte dans la détermination des stratégies qui seront adoptées, voir Figure 55 :

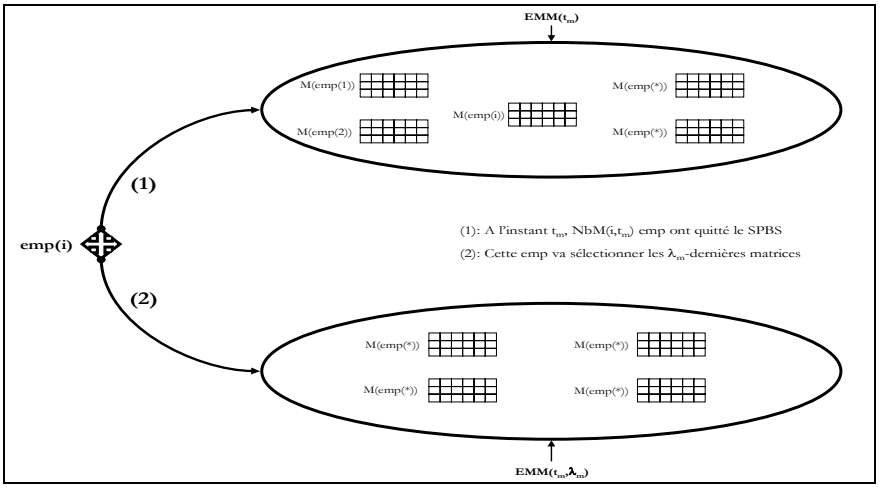


Figure 55- Notion de facteur d'oubli relatif à une emp

4.3.2.2 Module de diversification des données

Nous utilisons des mécanismes génétiques pour garantir la diversification des données prises en compte pour élaborer les NbT stratégies relatives à la prise de décision des nouvelles emp (la justification de ce choix est identique à celle donnée dans le § 3.3.2.2). Nous adaptons ces mécanismes génétiques dans le cadre du processus décisionnel d'une emp en définissant deux phases principales : la construction d'une population d'individus (de matrices) et la caractérisation des opérateurs génétiques présentés précédemment.

En premier lieu, nous construisons une population initiale (notée PIM) de matrices M(emp(*)) (ces matrices appartiennent de l'ensemble $EMM(t_m,\lambda_m)$). Nous tenons compte lors de cette construction de deux aspects importants : l'aspect optimisation en sélectionnant les m_m meilleures matrices de $EMM(t_m,\lambda_m)$ et l'aspect diversification en choisissant aléatoirement p_m matrices). L'objectif est alors d'obtenir une population améliorée de matrices (notée PAM) en se basant sur la population initiale PIM. Cette amélioration est assurée par des opérateurs génétiques. Pour ce faire, nous définissons trois étapes successives de la construction de la population améliorée :

- initialisation : la meilleure matrice de PIM est insérée dans la population améliorée,
- mécanismes génétiques : des opérateurs génétiques (croisement et mutation) sont appliqués sur des matrices M(emp(*)) appartenant à la population initiale,
- insertion : les matrices obtenues suite à l'application des mécanismes génétiques sont insérées dans PAM. Ce cycle (sélection, croisement, mutation et insertion) est recommencé jusqu'à l'obtention d'une population améliorée contenant (m_m+p_m) matrices.

En second lieu, nous avons adapté les opérateurs génétiques pour le processus décisionnel d'une entité mobile produit comme suit :

- l'opérateur de croisement (grâce à une fonction croisement(.,.)) combine deux matrices M(emp(i)) et M(emp(i')) choisies au hasard et appartenant à la population initiale. Cette combinaison consiste à fixer un point et une longueur de croisement et d'échanger localement le bloc délimité par ce point et cette longueur au niveau de chaque matrice, voir Figure 56. L'application de cet opérateur produit deux matrices (offsprings) et celle qui sera sélectionnée entre ces deux offsprings est notée Mm,

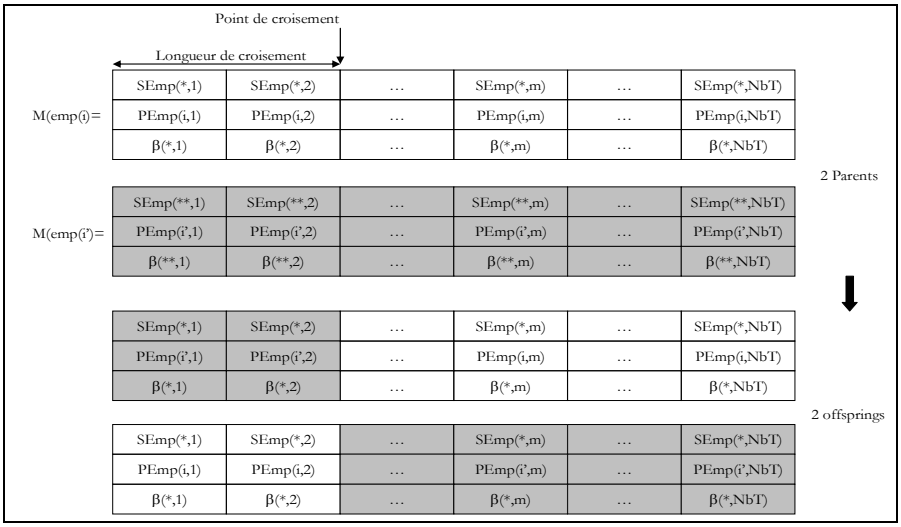


Figure 56- Opération de croisement avec deux matrices relatives à 2 emp différentes

- l'opérateur de mutation consiste à changer localement (grâce à une fonction notée mutation(.,.)) une colonne[55] de M(emp(i)) par une autre colonne appartenant à M(emp(i')). Cette opération nécessite la définition d'un point de mutation et la prise en compte d'une probabilité appelée taux de mutation (notée $\sigma_m \in [0,1]$). Enfin, une seule matrice notée Mm' résulte de cette opération de mutation (le principe de mutation relative au processus décisionnel d'une entité mobile produit est identique à celui présenté pour une entité fixe de traitement Figure 40).

Les phases de construction de la population améliorée PAM (initialisation et insertion des matrices résultantes des mécanismes génétiques) nécessitent un fitness. La matrice de mémorisation de données M(emp(i)) que nous avons proposée définit deux fitness $\sum_{m=1}^{NbT} \beta(*,m)$ (somme des évaluations des différentes stratégies adoptées) et $\sum_{m=1}^{NbT} PEmp(*,m)$ (somme des performances intermédiaires obtenues pour les différentes tâches). Notons que ces deux fitness sont équivalents car, comme nous le verrons par la suite, les coefficients β sont calculés sur la base des performances intermédiaires PEmp(*,m). Dans notre contribution, nous avons adopté une évaluation basée sur la somme $\sum_{m=1}^{NbT} \beta(*,m)$.

Ainsi, les mécanismes génétiques intégrés dans le processus décisionnel d'une entité mobile produit et composé de trois étapes et il est résumé sur la Figure 57 :

> **Début**
> Données initiales utilisées: $NbM(i,t_m)$, m_m, p_m, λ_m

[55] Généralement le point de mutation est choisi au hasard.

Étape 0 : Sélection des matrices prises en compte
Détermination de l'ensemble EMM(t_m,λ_m)
Évaluation des matrices appartenant à EMM(t_m,λ_m)
Étape 1 : Construction d'une population initiale (PIM)
Initialisation : Insertion de la meilleure matrice de EMM(t_m,λ_m)
Optimisation : Insertion des m_m meilleures matrices de EMM(t_m,λ_m)
Diversification : Insertion des p_m matrices choisies aléatoirement dans EMM(t_m,λ_m)
Étape 2 : Construction d'une population améliorée (PAM)
Tant que (card(PAM)< (m_m+p_m))
Choisir aléatoirement deux matrices M(emp(i)) et M(emp(i')) de PIM
Mm←croisement(M(emp(i)),M(emp(i')))
Si (Probabilité de mutation > σ_m) **alors**
Choisir aléatoirement une matrice M(emp(i"))
Mm'←mutation(Mm,M(emp(i")))
Fin si
Insérer Mm' dans PAM
Fin Tant que
Fin

Figure 57- Mécanismes génétiques adoptés dans le processus décisionnel d'une emp

Les mécanismes génétiques que nous venons de présenter ont un double rôle : garantir la diversification des données grâce au renouvellement des populations de matrices et la détermination des stratégies nécessaires à la réalisation des NbT tâches. Ces stratégies sont données par la meilleure matrice (la matrice qui maximise $\sum_{m=1}^{NbT} \beta(*,m)$) de la population améliorée PAM est sélectionnée et ses différentes stratégies sont adoptées.

4.3.2.3 Module d'évaluation continue des décisions prises

Des mécanismes d'apprentissage par renforcement sont intégrés dans le processus décisionnel d'une entité mobile produit. Ces mécanismes assurent une évaluation continue des décisions prises par une emp. L'intégration de ces mécanismes requiert la définition de deux paramètres :

- un signal de renforcement noté r_m qui est nécessaire pour exprimer numériquement la pertinence d'une décision prise. Or, les prises de décisions sont basées sur l'adoption d'un ensemble de stratégie. Donc ce sont les coefficients d'évaluation des différentes stratégies qui sont modifiés par ce signal de renforcement,
- un seuil noté s_m qui est nécessaire à la définition d'un intervalle permettant d'évaluer, pour chaque tâche m, la performance intermédiaire PEmp(i,m) réalisée. La définition de cet intervalle est basée sur la performance moyenne notée PMEmp(i,m,λ_m) : pour une tâche m donnée, PMEmp(i,m,λ_m) représente la moyenne de performances obtenues par les λ_m dernières emp finies à l'instant t_m. Ainsi, suivant la position de PEmp(i,m) sur cet intervalle (voir Figure 58), le coefficient d'évaluation relatif à la stratégie qui a permis l'obtention de cette performance est soit récompensé, soit pénalisé ou gardés inchangé.

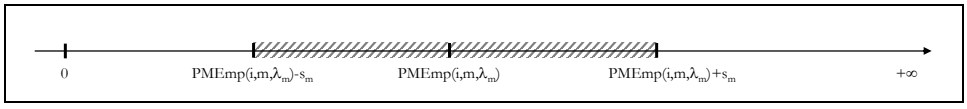


Figure 58- Valeurs tolérables des performances réalisées pour une emp

Précisons par ailleurs, que pour l'apprentissage par renforcement d'une entité mobile

produit, le problème de credit assignment problem ne se pose pas. En effet, il est possible, pour chaque tâche m réalisée, de quantifier la performance intermédiaire obtenue PEmp(i,m). Ainsi, lorsqu'une entité mobile produit quitte le SPBS, l'évaluation des décisions prises (et par conséquent l'évaluation des stratégies adoptées) est effectuée selon l'algorithme de la Figure 59 :

Données initiales utilisées: t_m, NbM(i,t_m), λ_m, I(λ_m), s_m, r_m
m←0
Début
Tant que m<NbT
m←m+1

$$\text{Calculer} \quad PMEmp(i, m, \lambda_m) = \frac{\sum_{a \in I(\lambda_m)} PEmp(a, m)}{\lambda_m}$$

Si PEmp(i,m)∈ [0, PMEmp(i,m,λ_m)-s_m [alors
SEmp(*,m) est récompensée
β'(*,m)← β(*,m)+r_m
Fin Si
Si PEmp(i,m)∈ [PMEmp(i,m,λ_m)-s_m, PMEmp(i,m,λ_m)+s_m] alors
SEmp(*,m) n'est ni récompensée ni pénalisée
β'(*,m)← β(*,m)
Fin Si
Si PEmp(i,m)∈]PMEmp(i,m,λ_m)+s_m , +∞ [alors
SEmp(*,m) est pénalisée
β'(*,m)← β(*,m)-r_m
Fin Si
β'(*,m) remplace β(*,m)
Fin Tant que
Fin

Figure 59- Évaluation des stratégies relatives à une emp

Nous expliquons ces notions de seuil, de signal de renforcement et de performance moyenne au travers de l'exemple suivant : considérons le processus décisionnel de l'entité mobile produit emp(105) qui entre dans le SPBS à l'instant t_m=150 et qui doit réaliser une gamme de quatre tâches (NbT=4). À cet instant, 100 emp ont déjà fini leurs traitements (NbM(i,t_m)=100), voir tableau Tab. 27-a. En outre, nous tenons compte d'un facteur d'oubli égal à cinq. Nous supposons par ailleurs que l'ensemble EMM(t_m,λ_m) contient les matrices associées aux emp suivantes : emp(100), emp(99), emp(102), emp(95) et emp(97). Les différentes performances intermédiaires obtenues par chaque emp appartenant EMM(t_m,λ_m) (pour les quatre tâches) sont mentionnées dans le tableau Tab. 27-b (par exemple PEmp(95,3)=2). Ainsi, la performance moyenne est calculée pour chaque tâche réalisée (par exemple PMEmp(105,3,5)= (2+7+3+2+5)/5=3.8).

Paramètres du modèle	Exemple
λ_m	5
t_m	150
r_m	1
s_m	1
I(λ_m)	{100,99,102,95,97}
NbM(i,t_m)	NbM(105,150)=100

(a)

		m=1	m=2	m=3	m=4
PEmp(i,m)	emp(100)	2	10	2	12
	emp(99)	0	3	7	2
	emp(102)	1	5	3	4
	emp(95)	3	1	2	0
	emp(97)	2	8	5	3
	emp(105)	0,3	7	3	2
	PMEmp(i,m,λ_m)	1.6	5.4	3.8	4.2
	[PMEmp(i,m,λ_m)-s_m, PMEmp(i,m,λ_m)+s_m]	[0.6,2.6]	[4.4,6.4]	[2.8,4.8]	[3.2,5.2]
	r_m	+1	-1	0	+1

(b)

Tab. 27 Exemple de comparaison entre la performance moyenne et la performance réalisée

Une fois emp(105) a réalisé la totalité de sa gamme, les coefficients d'évaluation des quatre

stratégies adoptées peuvent être modifiés (mis à jour). Cette mise à jour est possible grâce à l'ensemble de résultats numériques mentionnés dans les tableaux Tab. 27-a et Tab. 27-b. En effet, l'appartenance d'une performance intermédiaire à un intervalle précis (parmi les intervalles définis précédemment dans la Figure 58) déterminera le signal de renforcement attribué à chaque stratégie adoptée. Cette mise à jour des coefficients β est illustrée comme suit :

Nous supposons que emp(105) a adopté, respectivement, les stratégie SEmp(2), SEmp(4), SEmp(1) et SEmp(3). L'adoption de ces stratégies a permis d'obtenir l'ensemble de performances intermédiaires mentionnées dans le tableau Tab. 27-b (par exemple PEmp(105,4)=2). Nous remarquons que la performance intermédiaire PEmp(105,1) est excellente (elle appartient à [0,PMEmp(i,m,λ_m)]) et par conséquent la stratégie SEmp(2,1) est récompensée. Dans la Figure 60, les vecteurs relatifs aux quatre stratégies utilisées sont mis à jour comme suit (les valeurs changées sont soulignées):

(a) Valeurs initiales des coefficients β

M(emp(105))=

SEmp(2,1)	SEmp(4,2)	SEmp(1,3)	SEmp(3,4)
2	5	12	1

$E(SEmp(1))=\begin{bmatrix}10\\-3\\1\\0\end{bmatrix}$ $E(SEmp(2))=\begin{bmatrix}-5\\-2\\10\\-3\end{bmatrix}$ $E(SEmp(3))=\begin{bmatrix}6\\7\\-9\\4\end{bmatrix}$ $E(SEmp(4))=\begin{bmatrix}0\\-4\\12\\8\end{bmatrix}$

(b) Valeurs mises à jour des coefficients β

M(emp(105))=

SEmp(2,1)	SEmp(4,2)	SEmp(1,3)	SEmp(3,4)
2	5	12	1
-4	-5	1	5

$E(SEmp(1))=\begin{bmatrix}10\\-3\\1\\0\end{bmatrix}$ $E(SEmp(2))=\begin{bmatrix}\underline{-4}\\-2\\10\\-3\end{bmatrix}$ $E(SEmp(3))=\begin{bmatrix}6\\7\\-9\\\underline{5}\end{bmatrix}$ $E(SEmp(4))=\begin{bmatrix}0\\\underline{-5}\\12\\8\end{bmatrix}$

Figure 60- Exemple de mise à jour des coefficients d'évaluation des stratégies (SEmp)

Pour résumer, le processus d'évaluation des différentes stratégies est effectué lorsqu'une entité mobile produit quitte le SPBS (fin de la réalisation de la gamme) et marque la fin du processus décisionnel d'une emp. Nous récapitulons les différentes phases de ce processus en dressant le schéma de la Figure 61 qui comporte deux blocs I et II :

- le bloc I représente la phase d'initialisation du processus décisionnel des entités mobiles produits. Pendant cette phase, les stratégies nécessaires aux traitements des na_m premières emp sont générées aléatoirement. Les traitements de cet ensemble d'emp engendrent un ensemble de matrices M(emp(*)) nécessaires au démarrage de la deuxième phase,
- le bloc II est une boucle qui représente la phase du comportement intelligent dune entité mobile produit (mécanismes génétiques et apprentissage). Les données nécessaires à la réalisation de ce comportement sont déterminées lorsqu'une emp est sur le point d'entrée dans le SPBS. Pendant ce comportement intelligent, il s'agit premièrement de déterminer les NbT stratégies nécessaires à la réalisation d'une gamme de tâches (grâce aux mécanismes génétiques décrits précédemment). Deuxièmement, l'adoption de ces NbT stratégies qui permet d'obtenir un ensemble de performances intermédiaires PEmp(i,m). Troisièmement, la mise à jour des coefficients d'évaluation des NbT stratégies adoptées est réalisée en comparant la performance moyenne (calculée sur un ensemble de λ_m matrices M(emp(*))) avec la performance intermédiaire réalisée.

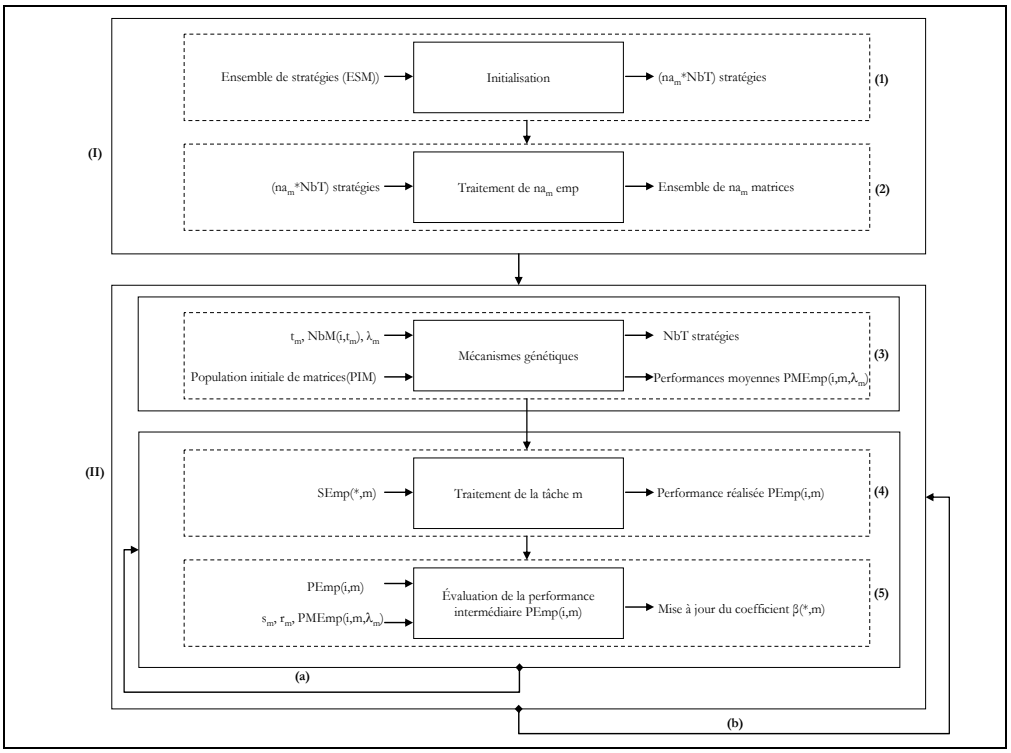


Figure 61- Récapitulatif du processus décisionnel global d'une entité fixe de traitement

Enfin, nous pouvons faire la correspondance entre le schéma de la Figure 61 et le RdP présenté sur la Figure 30, voir tableau Tab. 28 :

		RdP(m), Figure 30
Schéma, Figure 61	Bloc(1)	Zone 1, P3
	Bloc(2)	Zone 3
	Bloc(3)	P3
	Bloc(4)	P5, T5
	Bloc(5)	P6, T6
	Boucle(a)	Zone 3, P6, T6
	Boucle(b)	Zone 2, P3
	Bloc(I)	Zone 1
	Bloc(II)	Zone 2

Tab. 28 Correspondance entre le schéma de la Figure 61 et le RdP(m)

5. Récapitulatifs de notre approche

Tout d'abord, précisons que l'application des mécanismes génétiques n'est pas possible pour les 42 modes d'apprentissage spécifiés dans le chapitre deux. Prenons l'exemple d'une entité mobile produit qui adopte le type d'apprentissage suivant : A(emp(i))=<<externe,parallèle>,stratégie,simultané>.

Le fait d'apprendre à partir d'un ensemble d'entités mobiles qui sont encore sous traitement (état=parallèle) pose un problème pour l'application des mécanismes génétiques. En effet, les différentes matrices M(emp(*)) relatives à ces emp ne sont pas totalement remplies. Dans ce cas, il y a un apprentissage mais la population initiale PIM est définie autrement : premièrement, pour chaque tâche m il faut définir une population initiale et deuxièmement cette population est composée de vecteurs et non pas de matrices. La sélection des 'bonnes stratégies' se fait alors par optimisation directe (notée OP, choix du meilleur vecteur). Autrement dit, le choix se fait en

sélectionnant un vecteur (SEmp(*,m),PEmp(i,m),β(*,m)) parmi un ensemble de vecteur pour chaque tâche m. Le tableau Tab. 29 illustre cette précision :

Source de l'apprentissage (S)	Objet de l'apprentissage (O)		Moment de l'apprentissage (M)	Mécanismes génétiques (MG)/Optimisation (OP)
interne	stratégie		anticipé	AG
			simultané	AG
			mixte	AG
	étape		anticipé	AG
			simultané	AG
			mixte	AG
externe	État des autres entités (E)			
	parallèle (à partir des entités encore sous traitement)	stratégie	anticipé	OP
			simultané	OP
			mixte	OP
		étape	anticipé	OP
			simultané	OP
			mixte	OP
	décalé (à partir des entités qui ont fini leurs traitements)	stratégie	anticipé	AG
			simultané	AG
			mixte	AG
		étape	anticipé	AG
			simultané	AG
			mixte	AG
	mixte	stratégie	anticipé	AG et/ou OP
			simultané	AG et/ou OP
			mixte	AG et/ou OP
		étape	anticipé	AG et/ou OP
			simultané	AG et/ou OP
			mixte	AG et/ou OP
mixte	parallèle (à partir des entités encore sous traitement)	stratégie	anticipé	OP
			simultané	OP
			mixte	OP
		étape	anticipé	OP
			simultané	OP
			mixte	OP
	décalé (à partir des entités qui ont fini leurs traitements)	stratégie	anticipé	AG
			simultané	AG
			mixte	AG
		étape	anticipé	AG
			simultané	AG
			mixte	AG
	mixte	stratégie	anticipé	AG et/ou OP
			simultané	AG et/ou OP
			mixte	AG et/ou OP
		étape	anticipé	AG et/ou OP
			simultané	AG et/ou OP
			mixte	AG et/ou OP

Tab. 29 Applicabilité des mécanismes génétiques pour les différents modes d'apprentissage

Pour résumer, le tableau Tab. 30 récapitule les différents traits de notre approche pour l'autonomie et l'apprentissage relatifs aux deux types d'entités actives :

entité fixe de traitement (eft)	entité mobile produit (emp)
mémorisation des données : matrice [2*NbD]+une cellule	mémorisation des données : matrice [3*NbT]
mécanismes génétiques : sélection+croisement+mutation	
mémorisation des données : locale	mémorisation des données : globale
nombre de modes d'apprentissage possibles : 42	
A(eft(k))= <<interne, ∅>,stratégie,anticipé>	A(emp(i))= <<externe,décalé>,stratégie,anticipé>
apprentissage par renforcement : Récompense/pénalité collectives	apprentissage par renforcement : Récompense/pénalité individuelles
fréquence de l'apprentissage : tous les λ_f cycles	fréquence de l'apprentissage : tous les λ_m emp finies
correspondances (spécifications-propositions)	
autonomie de prise de décision (S1)	adoption d'une stratégie
cohérence des décisions prises (S2)	entités actives (emp et eft)
évaluation des décisions prise (S3)	tous les NbD décisions prises pour une emp, tous les NbT tâches réalisées pour une emp.
pertinence des décisions prises (S4)	facteurs d'oubli λ_f et λ_m
diversification des décisions prises (S5)	mécanismes génétiques
évaluation continue des décisions prises (S6)	mécanismes génétiques et de renforcement.

Tab. 30 Tableau récapitulatif des différentes propositions présentées dans notre travail

6. Discussions

Dans cette section, nous précisons deux points essentiels : la propriété d'intelligence collective des entités actives et le besoin de leur interopérabilité.

6.1. Intelligence collective des entités actives

Penalva [Penalva, 06] définit l'intelligence collective comme la capacité d'un groupe d'agents cognitifs (dans le cas général, ces agents peuvent être de nature humaine, animale ou artificielle) à atteindre dans l'action une performance d'un niveau supérieur. Cette intelligence sous-tend l'existence et la mise à profit de processus cognitifs d'apprentissage, de représentation, de décision, mais aussi de processus sociaux comme le partage, l'échange, la négociation, l'auto-organisation. Le comportement des entités actives, tel qu'il était présenté dans ce chapitre (prise de décision, apprentissage), s'intègre parfaitement dans le cadre de cette définition.

6.2. Interopérabilité des entités actives

L'interopérabilité est le fait que plusieurs systèmes, qu'ils soient identiques ou radicalement différents, puissent communiquer sans ambiguïté et opérer ensemble. L'interopérabilité est considérée comme très importante voire critique dans de nombreux domaines, dont l'informatique, le médical au sens large, les activités ferroviaires, l'électrotechnique, l'aérospatiale, le domaine militaire et l'industrie en général [Iung, 02]. Les différents systèmes, appareils et éléments divers utilisés doivent pouvoir interagir sans heurts. En productique, nous reprenons la définition de l'interopérabilité définie dans [Baïna et Morel, 06] 'l'interopérabilité représente la capacité de communiquer, de coopérer et d'échanger des modèles ou des données entre deux ou plusieurs 'agents' pour atteindre leurs objectifs et ceci malgré leurs différences'. Donc, les entités actives doivent être interopérable pour que leur intelligence collective garantisse l'amélioration continue des performances. La mise en œuvre d'entités actives interopérables (solutions techniques, protocoles de communications) dépasse le cadre de ce travail.

7. Conclusion

Dans ce chapitre, nous avons présenté notre approche basée sur l'intégration des capacités décisionnelles et d'apprentissage, localement, au niveau des entités actives. Chaque entité active prend ses propres décisions en se basant sur des stratégies. Ces décisions sont évaluées a posteriori pour quantifier leur pertinence. Ce processus d'évaluation représente l'ossature des mécanismes d'apprentissage de notre approche.

Nous avons proposé des mécanismes d'apprentissage de type 'par renforcement' à partir d'une génération de données par approche génétique. Ces mécanismes sont originaux dans le sens où ils sont simples à mettre en œuvre et utilisent un processus décisionnel déterministe. Les différents modèles présentés dans ce chapitre répondent aux spécifications définies dans le deuxième chapitre, pour la garantie de l'amélioration continue des performances des SPBS.

Dans le chapitre suivant, nous présentons la mise en œuvre que nous avons réalisée au cours de notre travail afin d'implémenter les différents modèles proposés.

CHAPITRE IV : MISE EN ŒUVRE

Ce chapitre est consacré à la présentation de la mise en œuvre des différents modèles (SPBS et structure de pilotage) présentés dans le deuxième et troisième chapitre. Pour ce faire, nous avons développé un simulateur informatique intégrant l'ensemble des caractéristiques de la modélisation proposée d'un SPBS et de son fonctionnement. Ainsi, et après avoir justifié le type de simulation adopté, nous décrivons les différents modules qui composent le simulateur. Basée sur la programmation orientée objet, cette description intègre les différentes classes définies ainsi que les algorithmes qui permettent de mettre en œuvre les différents mécanismes de prise de décisions et d'apprentissage. Le formalisme de modélisation unifié UML est utilisé pour donner une vue statique et dynamique du programme de simulation développé.

1. Conception d'un simulateur séquentiel orienté objet

La difficulté à valider les différentes propositions présentées dans ce travail sur un SPBS réel a orienté notre intérêt vers la simulation à événements discrets. Dans ce domaine, deux alternatives sont envisageables :

- utiliser un simulateur (commercial) existant et l'adapter pour mettre en œuvre les différents modèles décisionnels,
- concevoir intégralement un simulateur en utilisant un langage général de programmation ou un langage dédié à la simulation.

Avant de faire un choix entre ces deux alternatives, nous avons effectué une analyse (non exhaustive) des outils informatiques capables de simuler le fonctionnement d'un SPBS. Dans ce cadre, nous distinguons trois types d'outils complémentaires : les langages de programmation généraux, les langages de programmation dédiés à la simulation et les logiciels commerciaux pour la simulation des systèmes de production. Un aperçu de ces outils est donné dans le tableau Tab. 31 :

	exemple	lien web (site de référence)
Langages de programmation généraux	C, C++, Java, Visual Basic, Matlab, Fortran, …	-
Langages de programmation dédiés à la simulation	GPSS	www.minutemansoftware.com
	SIMSCRIPT	www.simscript.com
	BETA	www.daimi.au.dk/~beta
	QNAP (Queueing Network Analysis Package)	www.inria.fr
	SLAM II et AweSim	Pritsker Corporation
Simulateurs généraux	ManSim	www.mansim.com
	Siman Arena	www.software.rockwell.com
	Quest/Delmia	www.dassault.com
	FACTOR/AIM	www.csc.com
	ProcessModel	www.processmodel.com
	Taylor II	www.flexsim.com
	ProModel	www.promodel.com
	Simple++	www.tecnomatix.com
	Extend	www.imaginethatinc.com
	Micro Saint Sharp Simulation Software	www.maad.com
	SIMUL8	www.novasim.com
	WITNESS	www.lanner.com
	AutoMod	www.brookssoftware.com

Tab. 31 Panorama des langages et logiciels dédiés à la simulation[56]

Dans notre travail, nous avons conçu un simulateur en utilisant un langage de programmation général. Les trois raisons suivantes justifient ce choix :

- les logiciels de simulation disponibles sont généralement extensibles (c'est-à-dire qu'il est possible d'intégrer des modules (code) dans le simulateur, c'est le cas par exemple de Quest ou de Siman Arena). Néanmoins, cette extensibilité pose deux problèmes :
  - premièrement, la nécessité de décortiquer et maîtriser le fonctionnement et les mécanismes du simulateur, cette tâche est loin d'être aisée,

[56] Nous pouvons également y ajouter la suite des produits Ilog (Scheduler, Dispatcher, Cplex…) qui sont adaptés pour les problèmes d'optimisation.

  - deuxièmement, la nécessité de maîtriser le langage de programmation spécifique et/ou compatible de chaque simulateur.

- l'approche par langage de simulation (Qnap, Slam, Arena, etc.) est moins fine car elle propose des éléments de modélisation pré-définis : certes la modélisation est plus rapide mais le niveau de précision est plus faible,

- la majorité des simulateurs, essentiellement pour des raisons d'animation visuelle, utilisent une simulation 'temps réel' basée sur la notion de Timer (autrement dit, il s'agit souvent d'utiliser des threads, des mutex, des sémaphores, …). Cette spécificité engendre deux inconvénients :

  - outre la programmation, la gestion des priorités (ordonnancement des tâches dans le sens informatique) représente une difficulté supplémentaire qui est non négligeable. Dans la majorité des simulateurs, c'est le processeur informatique qui gère ces priorités,

  - la vitesse d'exécution et la quantité des entités simulées sont limitées (contrainte liée aux capacités de traitements du processeur).

- l'approche par langages de programmation généraux (Fortran, Pascal, C, C++, Lisp, Prolog, etc.) est très souple, très modulaire et son intérêt réside dans les possibilités de paramétrage qu'elle offre : de cette manière un modèle très précis peut-être établi,

Pour cet ensemble de raisons, nous avons développé un simulateur[57] en utilisant le langage de programmation orienté objet C++. La propriété 'objet' du langage C++ est idéalement adaptée au contexte de notre travail : d'une part, pour mettre en œuvre la modélisation d'un SPBS proposée (une entité est identifiée à un objet) et d'autre part pour simuler les différentes interactions entre ces entités. En ce qui concerne le type de simulation, nous avons choisi de concevoir un simulateur 'séquentiel à temps échantillonné'. Par rapport à un simulateur 'temps réel', ce type de simulateurs présente les avantages suivants :

- la durée de simulation est largement inférieure dans le cas d'un simulateur séquentiel,

- la gestion des priorités est plus facile à mettre en œuvre que dans le cas d'un simulateur dont le fonctionnement est étroitement lié au gestionnaire des tâches du processeur.

Pour décrire le simulateur que nous avons conçu, nous utilisons le formalisme UML (Unified Modeling Language, une présentation de ce formalisme de modélisation est donnée en annexe 3). UML est devenu le standard de facto pour la modélisation de logiciels informatiques et sa popularité grandissante a fait de lui un outil incontournable en génie logiciel et dans bien d'autres domaines. Dans ce sens, Blaha et Rumbaugh [Blaha et Rumbaugh, 05] recommandent trois composants pour la modélisation d'un 'logiciel' à l'aide du formalisme UML :

- le modèle de classes qui décrit la structure des objets d'un système : leur identité, leurs relations avec les autre objets, leurs attributs et leurs opérations. Il fournit un contexte pour les deux autres modèles (modèles d'états et d'interactions). Les diagrammes de classes permettent d'exprimer le modèle de classes,

- le modèle d'états qui décrit les aspects temporels des objets et l'ordonnancement des opérations qui les font passer d'un état à un autre : ce sont les événements qui marquent des changements d'états et qui définissent le contexte d'événements. Le modèle d'états est représenté par les diagrammes d'états,

- le modèle d'interactions qui décrit les interactions entre les objets qui collaborent pour

[57] Nous avons développé le simulateur sous l'environnement linux.

reproduire le comportement global du système informatique. Les modèles d'états et d'interactions sont complémentaires et décrivent différents aspects du comportement du système. En UML, ce sont les cas d'utilisation, les diagrammes de séquence et les diagrammes d'activités qui documentent le modèle d'interactions.

2. Le modèle des classes

La modélisation et la conception orientées objet se basent principalement sur la notion de classe. Dans le simulateur développé, nous distinguons deux familles de classes : celles qui permettent de modéliser un SPBS (que nous appelons classes de conception) et celles, appelées classes de calcul, qui permettent de mettre en œuvre le processus décisionnel des entités actives (tels que les prises de décisions, mécanismes génétiques, mécanismes d'apprentissage, …). Ces deux familles de classes sont interdépendantes. Nous commençons par présenter le diagramme des classes que nous avons définies puis nous les détaillons au fur et à mesure.

2.1. Diagramme des classes

La Figure 62 illustre une vue globale des principales classes ainsi que les relations qui existent entre elles :

- nous définissons une classe (appelée `workshop`) qui modélise le fonctionnement d'un SPBS. Un workshop est composé d'un ensemble de NbP produits[58], un ensemble de NbR ressources (`processor`) et d'une aire de stockage (`storage`) dans laquelle tous les produits finis sont stockés,
- les NbT tâches que doit réaliser chaque produit sont modélisées avec la classe `task`. Chaque `processor` est capable d'exécuter une ou plusieurs tâches.

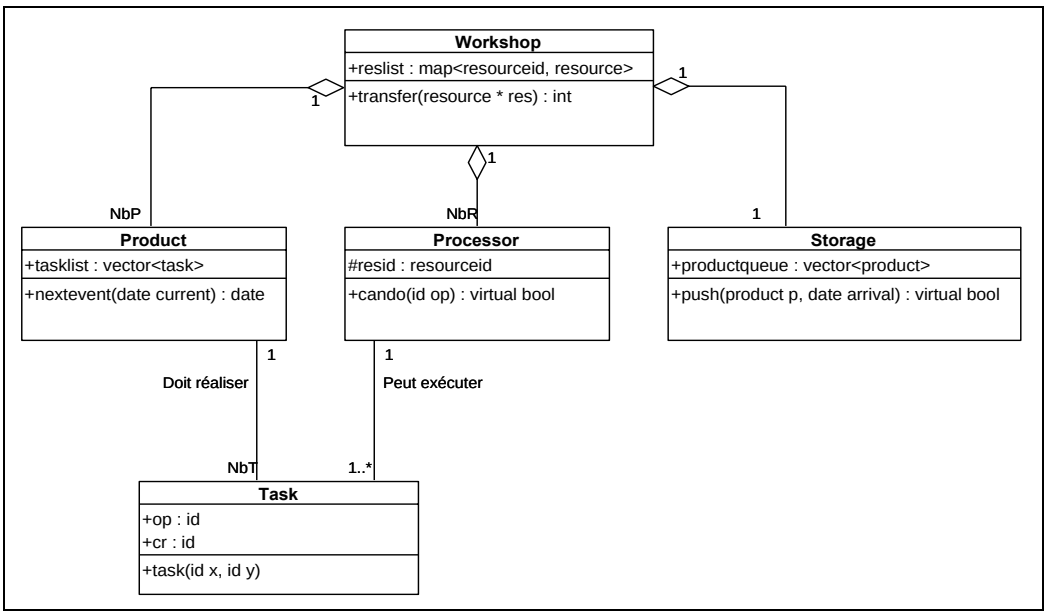


Figure 62- Diagramme des classes

2.2. Les classes de conception

[58] Pour des raisons de commodité, nous employons dans ce chapitre les termes suivants : ressource au lieu d'entité fixe de traitement, produit pour signifier une entité mobile produit et file d'attente au lieu d'entité fixe de stockage.

La première famille de classes est dédiée à la modélisation d'un SPBS. Celle-ci est caractérisée par la définition d'une classe `workshop` qui synchronise le fonctionnement des entités actives (définies par les classes `product` et `resource`). Nous commençons par présenter dans la partie suivante la modélisation d'une tâche qui lie une ressource à un produit dans un SPBS.

2.2.1. La classe task

Une tâche modélise une opération effectuée sur un produit (faite ou à faire) par une ressource appartenant au SPBS. Dans notre modélisation, nous définissons une classe `task` caractérisée par les attributs suivants :

- un ensemble d'identificateurs tels que l'identificateur de la tâche (noté op), l'identificateur de la ressource qui va assurer l'exécution de cette tâche (noté res) et l'identificateur de la stratégie de sélection ayant permis de choisir cette ressource (noté cr),
- un ensemble de dates telles que la date d'arrivée du produit sur la ressource (arrival) pour exécuter cette tâche, la date de début d'exécution de la tâche (start) et la date de départ du produit de la ressource (departure),
- un ensemble de durées notamment le temps opératoire (plan), le temps écoulé depuis la date de début d'exécution (done) et le temps d'exécution restant (todo),

En outre, l'identificateur de l'opération à réaliser (noté x) et celui de la stratégie adoptée pour réaliser cette opération (noté y) sont renseignés lors de la création d'une tâche (grâce au constructeur task(id x, id y), voir Figure 63 :

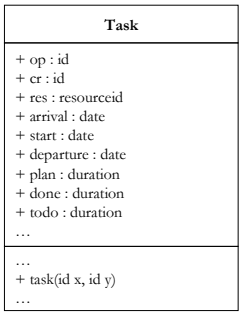


Figure 63- La classe task

2.2.2. La classe product

Un produit représente une entité qui circule dans le SPBS pour y subir séquentiellement les opérations de traitement (tâches) inscrites sur une liste. Chaque produit est une instance d'une classe `product` qui est caractérisée par :

- un ensemble d'attributs tels que :
  - l'indentificateur du produit (noté prodid) qui permet d'identifier chaque produit tout au long de son cycle de vie,
  - la liste des tâches à réaliser (notée tasklist[59]) qui représente une gamme ordonnée de toutes les opérations à réaliser et qui est modélisée avec un containeur de type vecteur[60],

[59] Les tâches réalisées sont conservées dans cette liste pour une analyse post-opératoire.
[60] Nous avons utilisé des conteneurs de la Standard Template Library (STL, voir www.sgi.com/tech/stl/index.html). Ce type de conteneurs (par exemple pair, vector, set, deque, map et multimap) facilite la mise en œuvre informatique.

- l'état d'avancement de la réalisation de la gamme, c'est-à-dire le nombre de tâches réalisées à une date donnée (noté currenttask), voir Figure 64.

- un ensemble d'opérations :
  - un constructeur de la classe `product` qui admet comme paramètres le nombre total des tâches à réaliser, la liste des tâches et les stratégies utilisées pour réaliser ces tâches. Rappelons que, pour chaque produit, les NbT stratégies adoptées pour réaliser une gamme de tâches sont déterminées au moment d'entrée du produit dans le SPBS (donc au moment de la création d'un objet `product`),
  - des opérations qui gèrent la liste des tâches (accès à la tâche courante (front()), passage à la tâche suivante (pop()), vérification si les NbT tâches sont réalisées (empty()), …),
  - des opérations qui gèrent les événements temporels relatifs à un produit (date du prochain événement significatif[61] (nextevent(date current)), avance de l'exécution de la tâche courante (step(duration x))), voir annexe 4 pour la description de ces opérations.

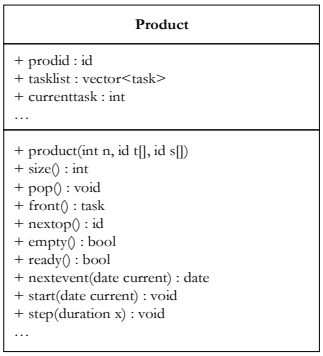


Figure 64- La classe product

En résumé, la création d'un objet de type produit est accompagnée par la détermination des NbT stratégies nécessaires à la réalisation de la gamme opératoire. Nous verrons par la suite qu'une fois créé, chaque produit est géré en fonction de son état (en transfert ou en traitement) : par le workshop s'il a fini la réalisation d'une tâche, et par la ressource s'il se trouve physiquement dans sa file d'attente.

2.2.3. La classe resource

Les entités fixes de traitements d'un SPBS sont modélisées grâce à une classe notée `resource`. Une ressource assure les traitements sur les différents produits qui circulent dans un SPBS. Chaque ressource gère une file d'attente, dont le premier élément est en cours de traitement. De cette classe, héritent deux sous-classes appelées `processor` et `storage` qui sont spécifiées comme suit :

- la classe `processor` est conçue pour modéliser le point d'entrée[62] ainsi que les différentes ressources qui composent un SPBS et qui assurent l'ensemble des traitements,

- le point de sortie (ou aire de stockage) est modélisé avec la classe `storage`. L'aire de stockage accueille tous les produits finis. Elle ne propose jamais de produit à transférer. Pour ce faire, les

[61] Par exemple le début ou la fin d'un traitement ou le transfert sur une ressource. En revanche, le changement de position d'un produit dans une file d'attente n'est pas un événement significatif.
[62] Nous considérons ce point comme une ressource qui assure l'introduction des produits dans le système (tâche 0). Par ailleurs, le point d'entrée est une ressource dont le temps opératoire est nul.

méthodes (initialement définies dans la classe `resource`) qui permettent à une ressource d'évoluer sont redéfinies, de façon à empêcher la sortie des produits.

Premièrement, la classe `resource` est caractérisée par un ensemble d'attributs tels que :

- la file d'attente couplée à chaque ressource (notée productqueue) qui est modélisée par un vecteur de produits. Dans cette queue, les produits sont triés par ordre d'arrivée : l'insertion d'un nouveau produit se fait avant les produits arrivant plus tard mais après les produits qui sont déjà dans la file d'attente et/ou qui peuvent avoir la même date d'arrivée,
- un vecteur de durées (noté processingtime) qui modélise les temps opératoires. Par exemple, le vecteur [0, 3, 0, 4, 0] est relatif à une gamme de trois tâches entrée→t(1)→t(2)→t(3)→sortie, donc la réalisation de t(3) dure 4 UTs,
- un conteneur de type map (noté failures) qui modélise les perturbations. Ce map est caractérisé par deux clefs : la date d'occurrence d'une perturbation et la quantification de cette perturbation (voir chapitre V, § 1.1.5 pour les perturbations prises en compte),
- un ensemble d'attributs tels que l'identificateur d'une ressource (noté resid), la date courante (notée currentdate), …
- les paramètres liés au processus décisionnel d'une ressource tels que le nombre de stratégies disponibles, le nombre de décisions par cycle (taille_cycle), le facteur d'oubli (taille_memoire), le seuil de performance (seuil_perf), la performance réalisée pour chaque cycle (notée pp)),
- les données nécessaires et générées par les mécanismes génétiques : l'ensemble des cycles de décisions est modélisé par un vecteur de 'cycle' noté memocycles (voir la partie suivante pour la définition de la classe `cycle`) et la liste de stratégies générées par les mécanismes génétiques (notée listestrat), voir Figure 65 :

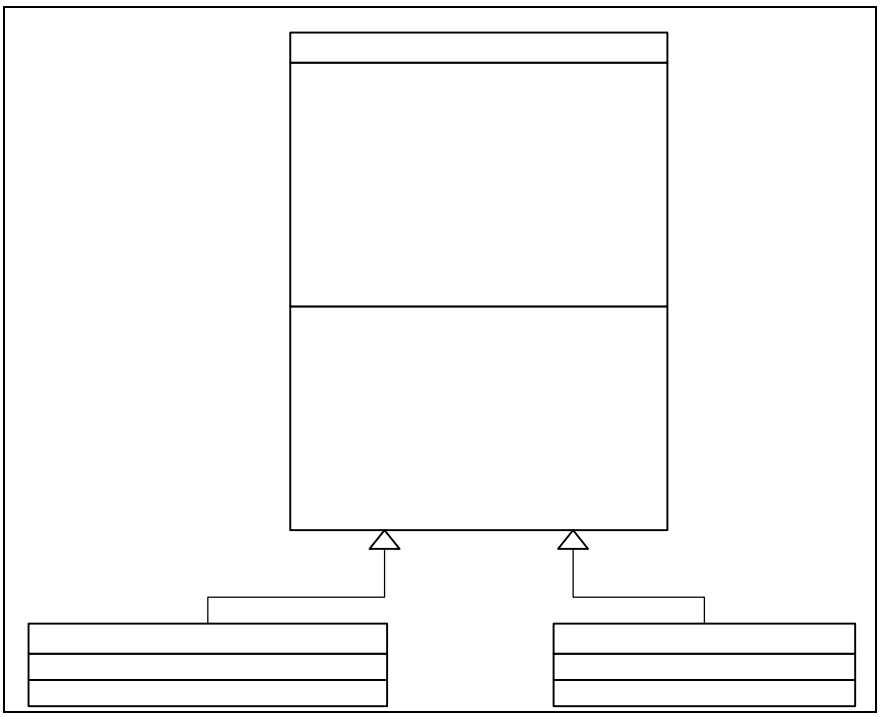


Figure 65- La classe resource

Deuxièmement, nous avons défini un ensemble d'opérations pour gérer l'ordre de passage

de chaque produit dans la file d'attente correspondante :

- une gestion des produits avant leur traitement : l'insertion d'un produit dans la file d'attente (push(product p, date arrival)), accès au produit courant (top()), …
- une gestion des événements temporels tels que le prochain événement significatif (nextevent()) ou l'avance de l'exécution d'une opération à une date donnée (advance(date current)), voir annexe 4,
- une opération pour la génération des NbD stratégies nécessaires à la réalisation d'un nouveau cycle de décisions (notée makenewstrat()),
- une opération pour déclencher une perturbation (notée fail(date d, processingtime delta)). Il s'agit d'un échéancier (sous forme d'un map, voir Figure 66) qui permet de modéliser l'apparition ou la disparition d'une ou plusieurs perturbations. Cet échéancier est caractérisé par deux clefs : la date de l'occurrence d'une perturbation (ou de sa réparation) et le changement observé du temps opératoire[63]. Par exemple :
  - l'échéancier <1000, [0, +100, 0, +100, 0]> indique qu'à la date 1000, les deux temps opératoires relatifs à la réalisation des tâches t(1) et t(3) sont augmentés de 100 UTs (début de la perturbation),
  - l'échéancier <1500, [0, -100, 0, -100, 0]> indique qu'à la date 1500, les deux temps opératoires relatifs à la réalisation des tâches t(1) et t(3) retrouvent leur valeur normale (fin de la perturbation).

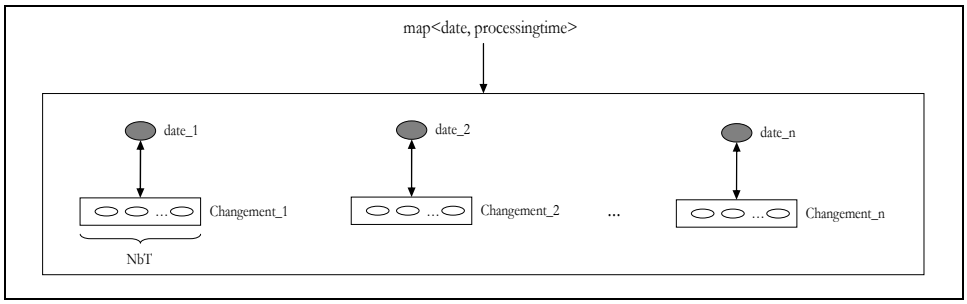


Figure 66- Représentation graphique de l'échéancier des perturbations (failures)

En résumé, chaque ressource effectue, localement, son processus décisionnel : les NbD stratégies sont générées cycliquement pour effectuer les sélections successives des produits présents dans la file d'attente correspondante. En outre, les perturbations sont gérées au niveau de chaque ressource.

2.2.4. La classe workshop

Nous avons modélisé un SPBS avec la classe `workshop`. Un `workshop` contient un ensemble de ressources et gère le déplacement des produits en cours de traitement. Cette classe englobe :

- la liste des ressources (notée reslist) qui composent le SPBS. Cette liste est modélisée sous la forme d'un containeur de type map défini par deux clefs : un identificateur d'une ressource et un objet de type ressource,

[63] Définie ainsi, la structure de l'échéancier des perturbations offre la possibilité d'intégrer des dates déterministes ou totalement aléatoires.

- l'ensemble des temps de transfert (noté transfertime) qui quantifient les temps de transfert entre les différentes ressources, voir la définition de cet attribut Figure 67. Par exemple le map : <2,[<1,6>,<2,1>,<3,8>]> indique qu'il faut 8 UTs pour transférer un produit se trouvant sur R(2) vers la file d'attente couplée à la ressource R(3),
- une fonction qui retourne la date du plus proche futur changement d'état significatif (notée nextevent()), voir annexe 4,
- le nombre de stratégies relatives au mécanisme décisionnel des produits (nb_strategies),
- la fonction transfer(resource res) qui permet de transférer un produit en sélectionnant une ressource parmi un ensemble de ressources candidates. Cette sélection est effectuée en adoptant une stratégie.

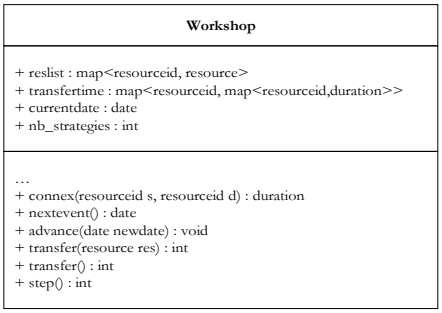


Figure 67- La classe workshop

Définie ainsi, la classe workshop gère le transfert des différents produits : c'est au niveau de cette classe qu'est défini l'ensemble des stratégies relatives aux produits.

2.3. Les classes de calculs

Les mécanismes décisionnels des entités actives proposés dans le troisième chapitre se basent principalement sur des mécanismes génétiques et d'apprentissage. Pour mettre en œuvre ces différents mécanismes, nous avons défini un ensemble de classes dédiées à cette mise en œuvre.

2.3.1. Les classes relatives au processus décisionnel d'un produit

Nous avons défini trois classes pour la mise en œuvre du processus décisionnel d'un produit : ces classes concernent la mémorisation des données, les mécanismes génétiques et la détermination des NbT stratégies pour chaque produit.

2.3.1.1 Les classes tvalue et individu

Pour mémoriser les données générées par le processus décisionnel d'un produit, nous avons défini une matrice M(emp(i)), voir chapitre III, § 4.2. Or cette matrice peut être considérée comme une combinaison de NbT vecteurs <SEmp(p,m),PEmp(i,m), β(p,m)>. Deux éléments de ce vecteur (SEmp et β) sont modélisés avec la classe `tvalue`, voir Figure 68. En effet, les trois attributs de cette classe notés first, second et third correspondent respectivement à p (identificateur de la stratégie), β (son évaluation) et m (identificateur de la tâche). Pour associer une matrice M(emp(i)) à un produit, nous définissons une classe `individu` : cette association est obtenue en définissant un vecteur de tvalue (noté gnome) qui est caractérisé par un fitness. Par ailleurs, c'est la classe individu qui assure les opérateurs génétiques (croisement et mutation) dans le processus décisionnel d'un produit, voir Figure 69.

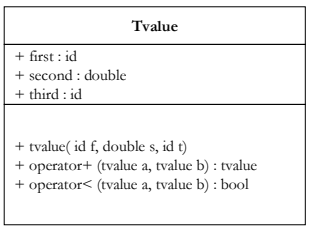


Figure 68- La classe tvalue

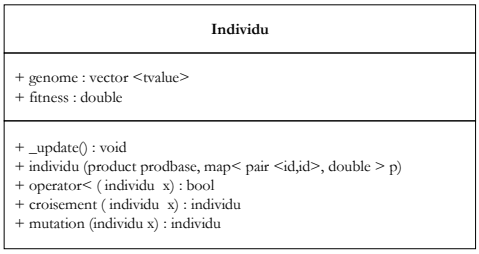


Figure 69- La classe individu

2.3.1.2 La classe optim

Les mécanismes génétiques et d'apprentissage sont mis en œuvre au travers de la classe `optim`, voir Figure 70. Cette classe permet de :

- modéliser la population initiale (ipop) et la population améliorée (apop), voir chapitre III, § 4.3.2.2,
- mettre en œuvre les mécanismes de renforcement : définition du seuil de performance (verybad), calcul des performances intermédiaires (wta), les performances moyennes (avgt(int n)),
- déterminer les NbT stratégies nécessaires à l'élaboration du processus décisionnel d'un produit (grâce aux opérations compute(int n) et best(id strategy[])).

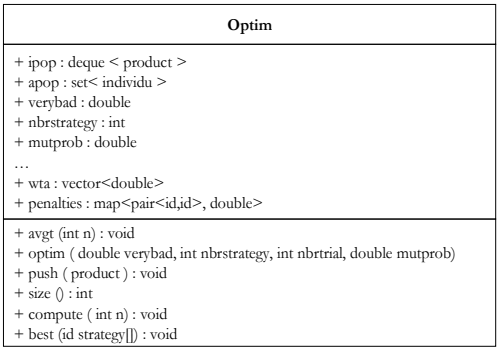


Figure 70- La classe optim

2.3.2. Les classes relatives au processus décisionnel d'une ressource

Chaque ressource génère cycliquement NbD stratégies pour la réalisation d'un nouveau cycle de décisions. Cette génération de stratégies est obtenue grâce aux mécanismes génétiques qui opèrent sur des matrices M(eft(k,h)). Cette spécificité est mise en œuvre dans notre simulateur garce à deux classes : `gene` et `cycle`.

2.3.2.1 La classe gene

Les éléments de la matrices M(eft(k,h)) sont une combinaison d'objets de type gene : une paire d'entiers. Le premier entier représente l'identificateur de la stratégie SEft(d) adoptée pour prendre une décision donnée et le deuxième représente le coefficient d'évaluation α relatif à cette stratégie. Précisons par ailleurs que la performance obtenue pour chaque cycle (notée pp) est un attribut de la classe resource, voir Figure 65.

2.3.2.2 La classe cycle

Chaque cycle de décision CEft(k,h) (noté chromosome) est modélisé avec un vecteur de gene de la classe `cycle`. Cette classe définit en outre deux opérations qui permettent de réaliser les deux opérateurs génétiques de croisement et de mutation.

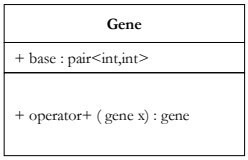


Figure 71- La classe gene

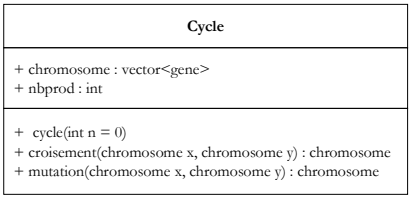


Figure 72- La classe cycle

En résumé, le fonctionnement du simulateur est assimilé plus aisément en examinant en premier lieu sa structure statique, autrement dit la structure de ses objets et les relations que ces derniers entretiennent à un moment donné (le modèle de classes). Il est ensuite préférable d'étudier les modifications des objets et de leurs relations au fil du temps : c'est le modèle d'états. Celui-ci décrit la séquence des opérations réalisées en réponse à des stimuli externes. La partie suivante est consacrée aux modèles d'états relatifs aux entités actives (produits et ressources).

3. Le modèle d'états

Le modèle d'états est constitué de plusieurs diagrammes d'états, un pour chaque classe dotée d'un comportement temporel significatif pour le simulateur. Le diagramme d'états relie des événements et des états : les événements représentent les stimuli externes (représentés par des flèches) et les états représentent les valeurs des objets (représentés par des boîtes aux coins arrondis). Un diagramme d'états est un graphe orienté dont les nœuds sont des états[64] et caractérisé par des transitions entre états et des événements :

- une transition est le passage d'un état à un autre qui est représentée par une ligne allant de l'état d'origine à l'état cible avec une flèche qui pointe vers l'état cible,
- plusieurs sortes d'événements existent, les types que nous avons utilisés sont :
  - un événement de changement, qui est causé par la satisfaction d'une expression booléenne, est représenté sous forme d'étiquette accolée à une transition. Un événement de changement est exprimé avec le mot clé 'when' suivi d'une expression booléenne,
  - un événement temporel qui est causé par l'occurrence d'un temps absolu représenté par le mot-clé 'when' suivi d'une expression indiquant un moment précis ou par le mot-clef 'after' suivi d'une expression exprimant une durée.

3.1. États d'un produit

Pendant son cycle de vie, chaque produit peut être caractérisé avec cinq états possibles :

- le produit est sur le point d'entrée : cet état représente l'introduction d'un produit dans le SPBS,

[64] UML dispose d'une notation spéciale pour les états initiaux (un cercle plein) et les états finaux (un cercle contenant un cercle plein).

- le produit est en transfert entre deux ressources s et d[65] : cet état (dont la durée du transfert est égale à transfertime[s][d]) représente le déplacement d'un produit entre une ressource s et la file d'attente d'une ressource d,
- le produit est en attente de traitement : cet état est dû à la présence simultanée de plusieurs produits sur une file d'attente. Donc, un produit reste en attente jusqu'à sa sélection (traitement) par la ressource correspondante : cette attente est représentée par une auto-transition au niveau de l'état 'Attente', voir Figure 73,
- une fois sélectionnée, la tâche relative au produit est traitée sur la ressource correspondante : c'est l'état 'Traitement' qui dure un temps égal à processingtime[task.id]. À la fin de cet état, deux transitions peuvent être franchies :
  - dans le cas où l'ensemble des NbT tâches sont réalisées (currenttask=NbT), le produit est transféré vers l'aire de stockage,
  - dans le cas où il reste encore des tâches à réaliser (currenttask<NbT), le produit est transféré vers la prochaine ressource destination.
- toutes les tâches sont réalisées et le produit est dans l'aire de stockage : c'est l'état final de chaque produit.

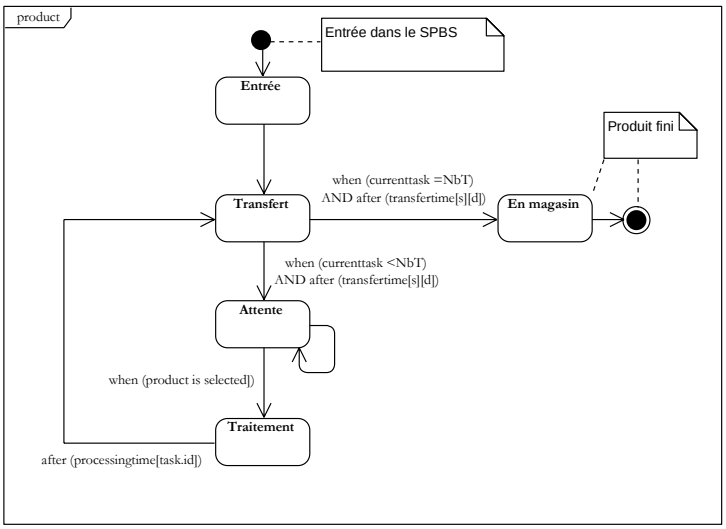


Figure 73- Diagramme d'états d'un produit

3.2. États d'une ressource

Dans le § 2.2.3, nous avons défini deux types de ressources : processor et storage qui modélisent respectivement les ressources qui assurent les traitement et l'aire de stockage. Dans cette section, nous nous intéressons à un processor[66] qui peut avoir trois états distincts :

- libre : un processor est libre lorsqu'il n'existe pas de produits en attente de traitement dans sa file d'attente (productqueue is empty),
- fonctionnement normal : un processor est en fonctionnement normal si, outre l'existence d'un ou plusieurs produits dans la file s'attente, les temps opératoires ne sont pas modifiés,

[65] s pour source et d pour destination.
[66] Un storage ne réalise aucun traitement et accepte tous les produits finis, donc possède un seul état : disponible.

(augmentation ou diminution des temps opératoires),

- fonctionnement perturbé : un processor est en fonctionnement perturbé si, outre l'existence d'un ou plusieurs produits dans la file s'attente, les temps opératoires sont modifiés.

À titre d'exemple, pour un processor, le passage de l'état 'libre' à l'état 'fonctionnement normal' est effectué uniquement s'il existe un ou plusieurs produits dans la file d'attente correspondante et si ses temps opératoires ne sont pas modifiés, voir Figure 74.

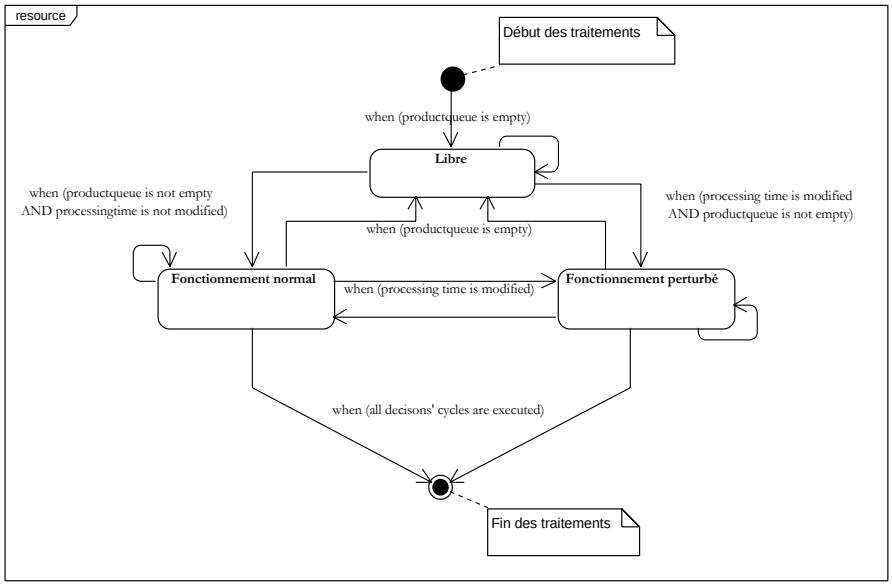


Figure 74- Diagramme d'états d'une ressource

Dernier des trois piliers de la modélisation prônés par Blaha et Rumbaugh, le modèle d'interactions décrit les interactions qui ont lieu au sein d'un système (dans notre cas c'est le simulateur). Le modèle de classes représente les objets et leurs relations, le modèle d'états décrit le cycle de vie des objets. Le modèle d'interactions, quant à lui, exprime la façon dont les objets interagissent pour produire des résultats utiles à l'application. Il représente une vue holistique du comportement du système à travers plusieurs objets alors que le modèle d'états est une vue réductionniste qui examine chaque objet individuellement. Ces deux modèles sont nécessaires pour décrire l'intégralité du comportement. Ils se complètent en présentant le comportement de deux points de vue différents.

4. Le modèle des interactions

Le formalisme UML définit trois types de diagrammes pour modéliser les différentes interactions : les cas d'utilisation, les diagrammes d'activités et les diagrammes de séquence. Dans ce qui suit, nous détaillons les trois types de diagrammes relatifs au simulateur.

4.1. Diagramme des cas d'utilisation

Nous pouvons modéliser les interactions du système à différents niveaux d'abstraction. Au niveau le plus élevé, les cas d'utilisation décrivent comment un système interagit avec les acteurs extérieurs (les utilisateurs du simulateur). Chaque cas d'utilisation représente une tranche de fonctionnalité que le système fournit à ces utilisateurs. Trois cas d'utilisation peuvent être distingués dans le simulateur proposé (voir Figure 75) :

- l'utilisateur saisit l'ensemble des données nécessaires au déroulement de la simulation : il s'agit d'une part de fixer les paramètres caractérisant un SPBS (nombre de ressources, temps opératoires, temps de transfert, gamme de tâches à réaliser, nombre de produits présents simultanément dans le système (noté NPS, voir chapitre V, § 1.1.1), date(s) d'occurrence(s) des perturbations, ….). D'autre part, les différents paramètres relatifs au système de pilotage proposé sont saisies (facteurs d'oubli, seuils de performance, taux de mutation, signaux de renforcement, …),
- les données saisies dans l'étape précédente sont enregistrées et prises en compte lors de l'exécution du programme de simulation,
- l'exécution du programme de simulation génère un ensemble de données qui sont enregistrées sous forme de fichiers (de type Excel). Essentiellement, il s'agit :
  - d'enregistrer, pour chaque produit, les stratégies adoptées, la trajectoire suivie, les performances intermédiaires obtenues, l'évolution des coefficients d'évaluation β, …
  - d'enregistrer, pour chaque ressource, les cycles de décisions réalisés, les stratégies adoptées, les différentes performances obtenues, l'évolution des coefficients d'évaluation α, …

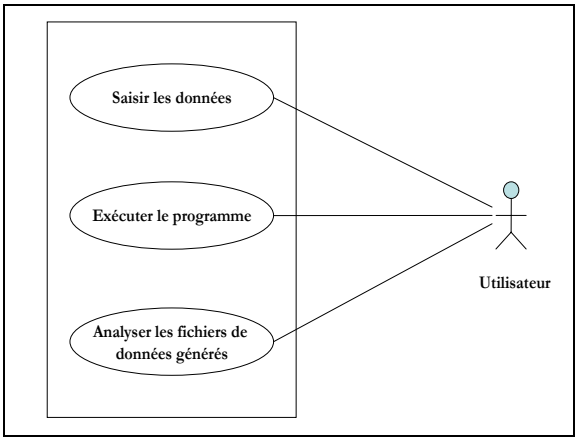


Figure 75- Diagramme des cas d'utilisation du simulateur réalisé

Nous détaillons dans ce qui suit le troisième cas d'utilisation. En effet, afin d'analyser les résultats obtenus, le simulateur conçu permet d'enregistrer des données. Celles-ci représentent les données relatives au processus décisionnel de chaque type d'entités actives.

4.1.1. Enregistrement des données générées par le processus décisionnel d'un produit

Deux types de données relatives au processus décisionnel d'un produit sont mémorisés. Premièrement, les données utilisées et engendrées par ce processus : pour chaque produit, sont enregistrées :

- les NbT stratégies adoptées pour la réalisation de l'ensemble des tâches,
- les NbT étapes qui définissent la trajectoire suivie par un produit,
- les NbT performances intermédiaires obtenues, successivement, lors de la réalisation de chaque tâche,
- la performance globale obtenue par chaque produit, voir Tab. 32.

	SEmp(*,1)	…	SEmp(*,m)	…	SEmp(*,NbT)
	EEmp(*,1)	…	EEmp(*,m)	…	EEmp(*,NbT)
	PEmp(1,1)	…	PEmp(1,m)	…	PEmp(1,NbT)
emp(1)[67]	PEmp(1)				
	SEmp(*,1)	…	SEmp(*,m)	…	SEmp(*,NbT)
	EEmp(*,1)	…	EEmp(*,m)	…	EEmp(*,NbT)
	PEmp(2,1)	…	PEmp(2,m)	…	PEmp(2,NbT)
emp(2)	PEmp(2)				
	SEmp(*,1)	…	SEmp(*,m)	…	SEmp(*,NbT)
	EEmp(*,1)	…	EEmp(*,m)	…	EEmp(*,NbT)
	…	…	…	…	…
…	…				
	SEmp(*,1)	…	SEmp(*,m)	…	SEmp(*,NbT)
	EEmp(*,1)	…	EEmp(*,m)	…	EEmp(*,NbT)
	PEmp(i,1)	…	PEmp(i,m)	…	PEmp(i,NbT)
emp(i)	PEmp(i)				
	SEmp(*,1)	…	SEmp(*,m)	…	SEmp(*,NbT)
	EEmp(*,1)	…	EEmp(*,m)	…	EEmp(*,NbT)
	…	…	…	…	…
…	…				
	SEmp(*,1)	…	SEmp(*,m)	…	SEmp(*,NbT)
	EEmp(*,1)	…	EEmp(*,m)	…	EEmp(*,NbT)
	PEmp(*,1)	…	PEmp(*,m)	…	PEmp(*,NbT)
emp(*)[68]	PEmp(*)				

Tab. 32 Enregistrement du processus décisionnel de l'ensemble des produits

Ces données sont nécessaires pour analyser les différentes stratégies adoptées, les ressources utilisées pour l'exécution de la gamme, ou encore l'évolution des performances globales des produits finis.

Le produit emp(1), pour réaliser sa deuxième tâche, a adopté la stratégie d'indice 4, ce qui l'a orienté vers la ressource d'indice 1. Ce produit a passé 3 UTs sur la file d'attente avant l'exécution de cette tâche. Le temps d'attente total relatif à ce produit est égal à 10 UTs, voir Tab. 33. Ce temps d'attente total représente un indicateur de performance.

	SEmp(2,1)	SEmp(4,2)	SEmp(1,3)	SEmp(3,4)
	EEmp(3,1)	EEmp(1,2)	EEmp(1,3)	EEmp(2,4)
	1	3	2	4
emp(1)	10			

Tab. 33 Exemple de l'enregistrement du processus décisionnel relatif à emp(1)

Deuxièmement, pour l'ensemble des stratégies appartenant à ESM, le simulateur génère un fichier qui mémorise l'historique (noté MUM(d)) de l'adoption de chaque stratégie SEmp(d). Cet historique permet, d'une part, de mémoriser l'évolution des coefficients β d'évaluation des stratégies et d'autre part de déterminer le nombre d'utilisations d'une stratégie (noté nm(p)) pendant le processus décisionnel, voir Tab. 34 :

	β(1,1)	…	β(1,m)	…	β(1,NbT)
	…	…	…	…	…
	β(1,1)	…	β(1,m)	…	β(1,NbT)
MUM(1)	nm(1)				

[67] Le bloc en gris sera systématiquement illustré d'un exemple.

[68] Rappelons que les emp quittent le SPBS selon un ordre différent de celui de leur entrée : par exemple emp(1100) peut finir l'ensemble de ses tâches avant emp(1099).

	…	…	…	…	…
	…	…	…	…	…
	…	…	…	…	…
…	…				
	β(p,1)	…	β(p,m)	…	β(p,NbT)
	…	…	…	…	…
	β(p,1)	…	β(p,m)	…	β(p,NbT)
MUM(p)	nm(p)				
	…	…	…	…	…
	…	…	…	…	…
	…	…	…	…	…
…	…				
	β(NSm,1)	…	β(NSm,m)	…	β(NSm,NbT)
	…	…	…	…	…
	β(NSm,1)	…	β(NSm,m)	…	β(NSm,NbT)
MUM(NSm)	nm(NSm)				

Tab. 34 Enregistrement de l'évolution des coefficients d'évaluation d'une stratégie SEmp

Ces données sont utiles pour observer le comportement d'une stratégie donnée suite à ses adoptions successives par les différents produits ainsi que la répartition de l'utilisation des différentes stratégies.

Le tableau Tab. 35 illustre l'évolution des coefficients d'évaluation de la stratégie SEmp(1) (les valeurs modifiées (mises à jours) sont colorées en gris). Nous remarquons que cette stratégie a été utilisée 5 fois pendant le processus décisionnel et a donné de bonnes performances pour l'exécution de la deuxième tâche. En revanche, cette stratégie n'a pas été adoptée pour l'exécution des tâches t(3) et t(4).

	0	0	0	0	+1
	-1	0	0	0	+1
	-1	+1	0	0	+1
	0	+1	0	0	+1
	0	+2	0	0	+1
MUM(1)	5				

Tab. 35 Exemple de l'enregistrement de l'évolution des coefficients d'évaluation de la stratégie SEmp(1)

4.1.2. Enregistrement des données générées par le processus décisionnel d'une ressource

Le simulateur mis en œuvre permet de mémoriser deux types de données relatives au processus décisionnel d'une ressource. Premièrement, les données utilisées et générées par ce processus : pour chaque ressource k, sont mémorisés :

- les NbC(k,t_f) cycles réalisés pendant son processus décisionnel (t_f correspond à la date de fin de la réalisation du dernier cycle de décisions, en l'occurrence t_f coïncide dans ce cas à la fin de la simulation),
- pour chaque cycle de décision h : les NbD stratégies adoptées pour réaliser ce cycle, voir Tab. 36 :

	CEft(1,1)	SEft(*,1)	…	SEft(*,n)	…	SEft(*,NbD)
	…	…	…	…	…	…
	CEft(1,h)	SEft(*,1)	…	SEft(*,n)	…	SEft(*,NbD)
	…	…	…	…	…	…
eft(1)	CEft(1,NbC(1,t_f))	SEft(*,1)	…	SEft(*,n)	…	SEft(*,NbD)
…	…	…	…	…	…	…
	…	…	…	…	…	…

	…	…	…	…	…	…
	…	…	…	…	…	…
	…	…	…	…	…	…
	CEft(k,1)	SEft(*,1)	…	SEft(*,n)	…	SEft(*,NbD)
	…	…	…	…	…	…
	CEft(k,h)	SEft(*,1)	…	SEft(*,n)	…	SEft(*,NbD)
	…	…	…	…	…	…
eft(k)	CEft(k,NbC(k,t_f))	SEft(*,1)	…	SEft(*,n)	…	SEft(*,NbD)
	…	…	…	…	…	…
	…	…	…	…	…	…
	…	…	…	…	…	…
	…	…	…	…	…	…
…	…	…	…	…	…	…
	CEft(NbR,1)	SEft(*,1)	…	SEft(*,n)	…	SEft(*,NbD)
	…	…	…	…	…	…
	CEft(NbR,h)	SEft(*,1)	…	SEft(*,n)	…	SEft(*,NbD)
	…	…	…	…	…	…
eft(NbR)	CEft(NbR,NbC(NbR,t_f))	SEft(*,1)	…	SEft(*,n)	…	SEft(*,NbD)

Tab. 36 Enregistrement des cycles de décisions réalisés par les eft

Ces données sont utiles pour analyser l'utilisation des différentes stratégies adoptées pour la réalisation des cycles de décisions par chaque ressource.

À la fin de la simulation, la ressource eft(1) a réalisé 6 cycles composés chacun de 5 décisions. Par exemple, la réalisation du 3ème cycle a été effectuée en adoptant respectivement les stratégies d'indices 1, 6, 4, 2 et 1, voir Tab. 37 :

	CEft(1,1)	SEft(2,1)	SEft(1,2)	SEft(2,3)	SEft(3,4)	SEft(4,5)
	CEft(1,2)	SEft(3,1)	SEft(4,2)	SEft(4,3)	SEft(1,4)	SEft(2,5)
	CEft(1,3)	SEft(1,1)	SEft(6,2)	SEft(4,3)	SEft(2,4)	SEft(1,5)
	CEft(1,4)	SEft(5,1)	SEft(3,2)	SEft(2,3)	SEft(6,4)	SEft(3,5)
	CEft(1,5)	SEft(3,1)	SEft(5,2)	SEft(7,3)	SEft(1,4)	SEft(6,5)
eft(1)	CEft(1,6)	SEft(2,1)	SEft(3,2)	SEft(4,3)	SEft(5,4)	SEft(2,5)

Tab. 37 Exemple de l'enregistrement des cycles de décisions réalisés par eft(1)

Deuxièmement, pour l'ensemble des stratégies appartenant à ESF(k), le simulateur génère un fichier qui mémorise l'historique (noté MUF(k,d)) de l'adoption de chaque stratégie SEft(d) par une ressource eft(k). Cet historique permet, d'une part, de mémoriser l'évolution des coefficients α d'évaluation des stratégies et d'autre part de déterminer le nombre d'utilisations d'une stratégie (noté nf(k,d)) pendant le processus décisionnel, voir Tab. 38 :

	α(1,1)	…	α(1,n)	…	α(1,NbD)
	…	…	…	…	…
	α(1,1)	…	α(1,n)	…	α(1,NbD)
MUF(k,1)	nf(k,1)				
	…	…	…	…	…
	…	…	…	…	…
	…	…	…	…	…
…	…				
	α(d,1)	…	α(d,n)	…	α(d,NbD)
	…	…	…	…	…
	α(d,1)	…	α(d,n)	…	α(d,NbD)
MUF(k,d)	nf(k,d)				
…	…	…	…	…	…
	…	…	…	…	…
	…	…	…	…	…

	…				
MUF(k, NSf)	α(NSf,1)	…	α(NSf,n)	…	α(NSf,NbD)
	…	…	…	…	…
	α(NSf,1)	…	α(NSf,n)	…	α(NSf,NbD)
	nf(k,NSf)				

Tab. 38 Mémorisation de l'évolution des coefficients d'évaluation des stratégies SEft relatives à une eft(k)[69]

Ces données sont utiles pour observer le comportement d'une stratégie donnée suite à ses adoptions successives pour la réalisation des différents cycles par une ressource donnée ainsi que la répartition de l'utilisation des différentes stratégies.

L'évolution des coefficients d'évaluation de la stratégie SEft(1), utilisée 7 fois pendant le processus décisionnel d'une eft(k), montre que cette stratégie a été utilisée 3 fois pour prendre la troisième décision et qu'elle a donné de bonnes performances, voir Tab. 39 :

MUF(k,1)	0	0	0	0	+1
	0	0	+1	0	+1
	0	0	+2	0	+1
	0	0	+3	0	+1
	0	0	+3	0	+2
	-1	0	+3	0	+2
	-2	0	+3	0	+2
	7				

Tab. 39 Exemple d'enregistrement de l'évolution des coefficients d'évaluation de SEft(1) relative à eft(k)

4.2. Diagrammes d'activités des entités actives

Les diagrammes d'activités fournissent des détails supplémentaires et représentent le flux de contrôle entre les étapes d'un traitement. Ils montrent aussi bien des flux de données que des flux de contrôle. Ils décrivent également les étapes nécessaires à l'implémentation d'une opération ou d'un processus métier référencé dans un diagramme de séquence. Dans ce qui suit, nous détaillons les diagrammes d'activités des entités actives (seules les phases de fonctionnement 'intelligent' des produits et des ressources sont représentées sur ces diagrammes, la première phase (fonctionnement aléatoire) étant une phase transitoire).

4.2.1. Diagramme d'activités d'un produit

La première activité effectuée par un produit, au moment de sa création, consiste à appliquer les mécanismes génétiques pour déterminer les NbT stratégies nécessaires à la réalisation de sa gamme de tâches. La réalisation de cette activité constitue une condition nécessaire pour débuter d'autres activités. En effet, tant que la totalité des tâches n'est pas réalisée, trois activités successives sont effectuées :

- la stratégie SEmp(*,m) est adoptée pour la réalisation d'une tâche m : cette activité permet de sélectionner une ressource destination pour réaliser cette tâche,
- une fois sélectionnée et après un temps de transfert, la ressource destination accueille ce produit : un temps d'attente (variable) est observé avant l'exécution de la tâche m,
- le produit est sélectionné : il est pris en charge par la ressource correspondante pour l'exécution de la tâche m.

[69] Les mêmes données sont générées NbR fois, c'est-à-dire pour chaque ressource.

La fin de la réalisation des NbT tâches déclenche les activités de mise à jours des coefficients d'évaluation des stratégies ainsi que la mémorisation des données relatives au processus décisionnel du produit, voir Figure 76.

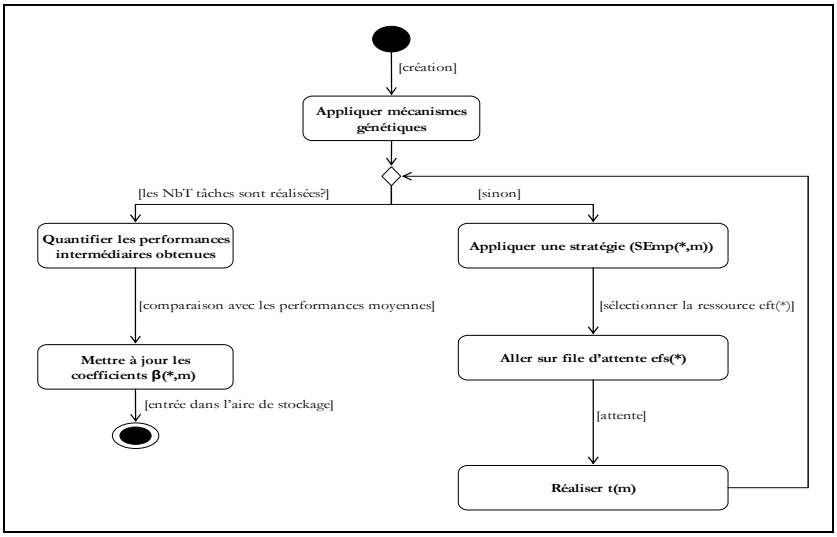


Figure 76- Diagramme d'activités d'un produit

4.2.2. Diagramme d'activités d'une ressource

L'enchaînement des activités effectuées par une ressource est globalement semblable à celui relatif aux produits à une différence près : cet enchaînement est répétitif. En effet, pour chaque nouveau cycle de décisions, une ressource applique des mécanismes génétiques pour déterminer les NbD stratégies nécessaires à la réalisation de ce cycle. La détermination de ces stratégies permet d'entamer d'autres activités. En effet, tant que la totalité des décisions n'est pas prise, deux activités successives sont effectuées :

- première activité : la stratégie SEft(*,n) est appliquée pour sélectionner un produit parmi ceux qui sont présents dans la file d'attente couplée à cette ressource,
- deuxième activité : la tâche m relative au produit sélectionnée est traitée,

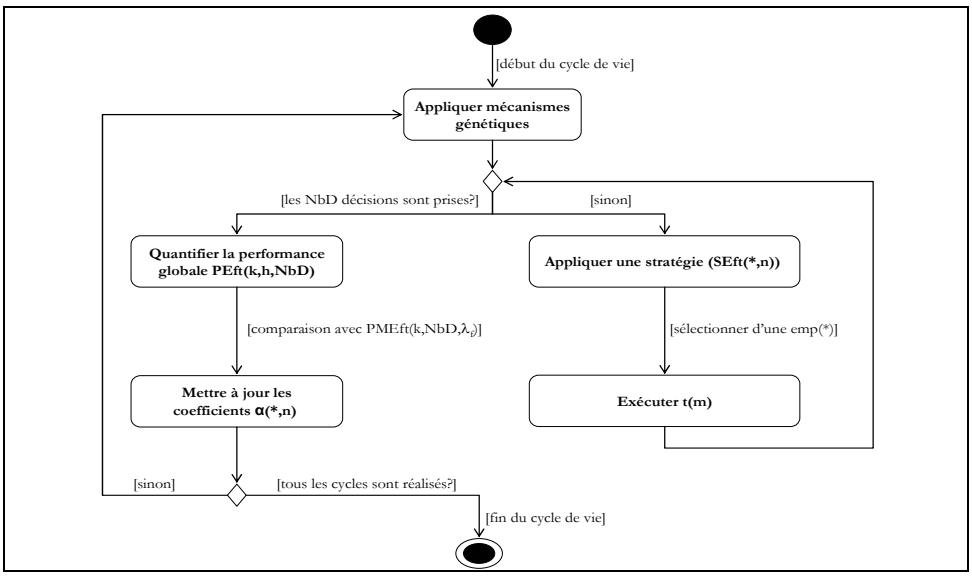


Figure 77- Diagramme d'activités d'une ressource

La prise de la totalité des NbD décisions, pour un cycle h, entraîne les activités de mise à jours des coefficients d'évaluation des stratégies ainsi que la mémorisation des données relatives au processus décisionnel de la ressource, voir Figure 77

4.3. Diagrammes de séquences des entités actives

Les diagrammes de séquences fournissent plus de détails et représentent les messages que s'échange un ensemble d'objets au fil du temps. Les messages comprennent à la fois les signaux asynchrones et les appels de procédures. Les diagrammes de séquences sont bien adaptés pour représenter les séquences de comportement perçues par les utilisateurs d'un système informatique.

Le déroulement de la simulation est assuré par un programme principal appelé 'main'. Ce programme fait appel aux différentes classes présentées précédemment pour simuler le fonctionnement d'un SPBS. Pour des raisons de lisibilité, nous représentons séparément les diagrammes de séquences de chaque type d'entités actives.

4.3.1. Diagramme de séquences d'un produit

La saisie et la validation des données nécessaires à la définition d'un SPBS (premier cas d'utilisation, voir Figure 75) constitue la première séquence du simulateur. Ensuite, ces données sont utilisées pour créer les objets qui composent un SPBS :

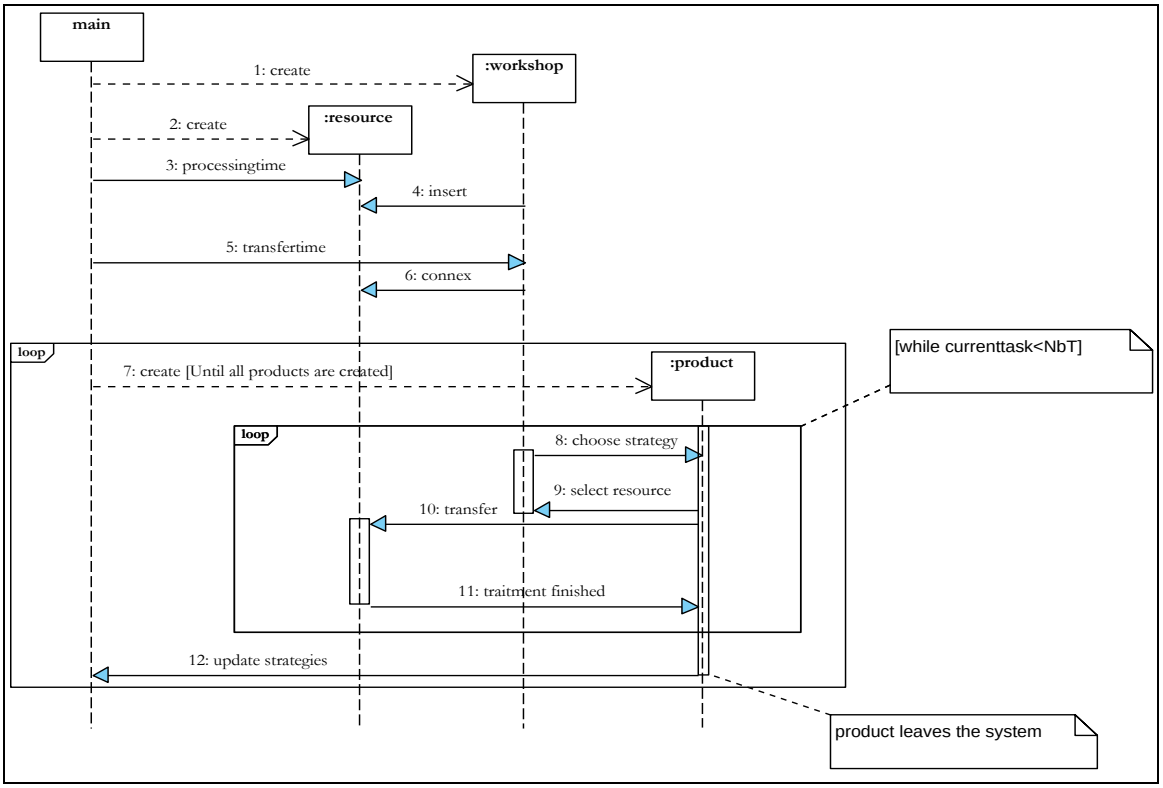


Figure 78- Diagramme de séquences d'un produit

- création d'une instance de type workshop et NbR objets de type resource,
- création et initialisation des temps opératoires sur chaque ressource,
- insertion des différentes ressources dans le workshop,
- création et initialisation des temps de transfert dans le workshop,

- connexion des différentes ressources, deux à deux, avec les temps de transfert.

Les séquences relatives aux produits sont représentées par deux boucles imbriquées :

- création d'un objet de type product : cette séquence est répétée tant qu'il y a possibilité d'en créer (simulation en cours),
- ordonnancement de chaque produit par le workshop : cette séquence est répétée tant que la totalité des tâches n'est pas encore exécutée, voir Figure 78 :

4.3.2. Diagramme de séquences d'une ressource

Principalement, les séquences relatives aux ressources sont représentées par deux boucles imbriquées :

- chaque ressource génère les stratégies nécessaires à la réalisation de ses cycles de décisions : l'adoption de ces stratégies, qui permettent la sélection des produits, est effectuée NbD fois,
- les cycles de décisions sont réalisés successivement : cette séquence est répétée durant le cycle de vie d'une ressource (durée de la simulation),

Nous avons représenté aussi, sur la Figure 79, l'enregistrement des données utilisées et générées par le processus décisionnel d'une ressource telque nous l'avons défini dans le § 4.1.2 sous forme d'une boucle pour toutes les ressources.

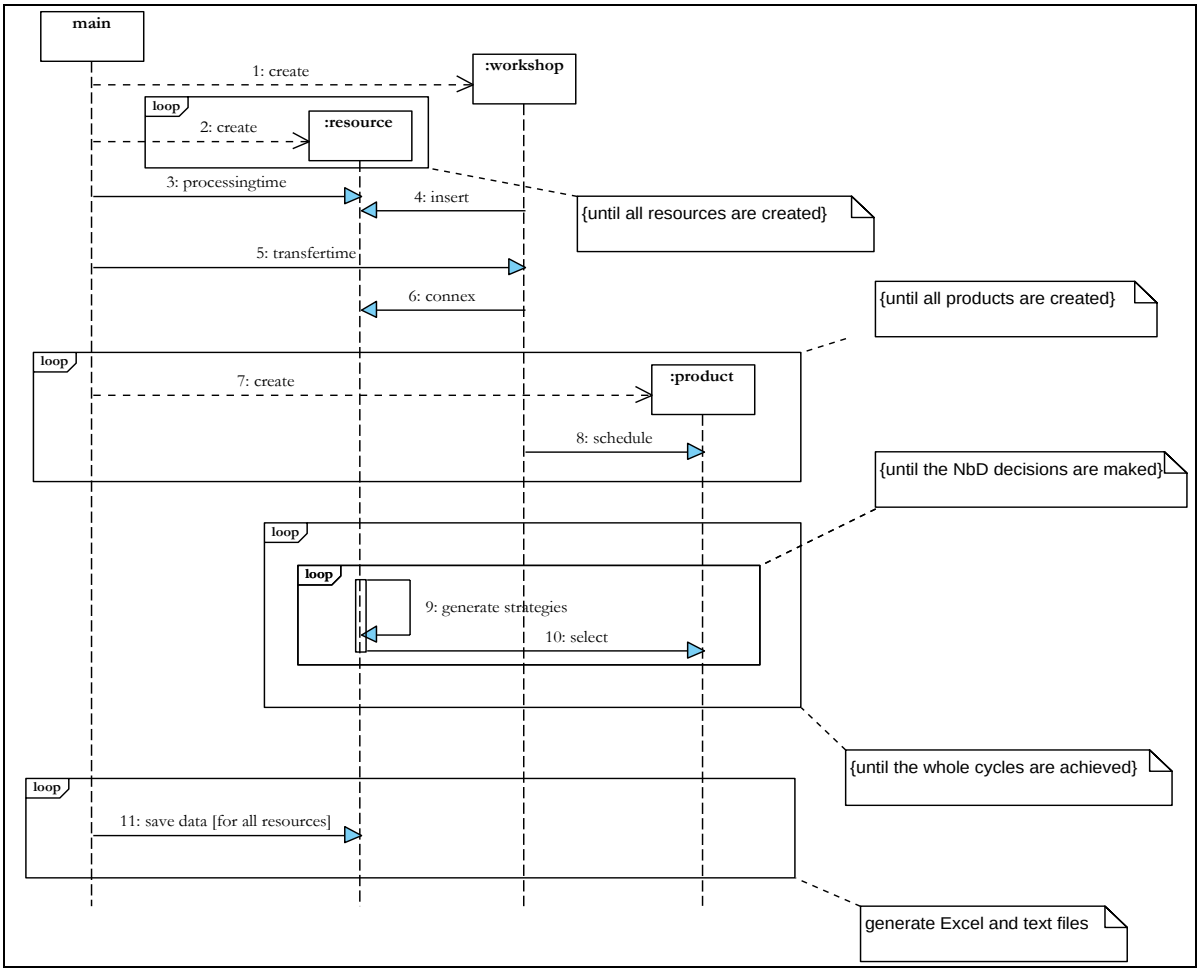


Figure 79- Diagramme de séquences d'une ressource

5. Tableau récapitulatif

Nous donnons, dans le tableau Tab. 40, les correspondances entre le modèle informatique réalisé et les paramètres du modèle de pilotage proposé dans le troisième chapitre :

Modèles (chapitre III)		Simulateur
entité fixe de traitement		classe resource
entité fixe de stockage		productqueue (attribut de la classe resource)
entité mobile produit		classe product
entité mobile de transfert		transfertime (attribut de la classe workshop)
stratégie ressource		attribut de la classe resource
stratégie produit		attribut de la classe task
perturbation		opération fail (classe resource)
NPS		donnée globale
facteurs d'oubli	λ_m	donnée globale
	λ_f	taille_memoire (classe ressource)
taux de mutation	σ_m	mutprob (classe optim)
	σ_f	mut_prob (opération makenewstrat())
seuils de performance	s_m	verybad (classe optim)
	s_f	seuil_perf (classe ressource)
matrices	M(eft(k,h))	classe cycle
	M(emp(i))	classe individu
génération des stratégies	emp	best (id strategy[]) (classe optim)
	eft	makenewstrat() (classe ressource)

Tab. 40 Correspondance entre le modèle de pilotage et sa mise en œuvre

6. Conclusion

Nous avons présenté dans ce chapitre une mise en œuvre du simulateur de fonctionnement d'un SPBS en intégrant les différents modèles décisionnels et d'apprentissage proposés dans notre travail. Une description des différents composants du simulateur a été présentée à l'aide du formalisme de modélisation UML.

Le simulateur développé permet :

- de dimensionner un SPBS (nombre de ressources, paramètres de fonctionnement, …),
- d'intégrer les mécanismes décisionnels (adoption des stratégies) et d'apprentissage (renforcement, opérateurs génétiques, …),
- d'enregistrer, sous forme de fichier Excel, les résultats obtenus pour les analyser.

Dans le chapitre suivant, nous montrons les résultats des simulations réalisées afin d'évaluer nos propositions.

CHAPITRE V : ÉVALUATION

Dans ce chapitre nous évaluons et analysons les modèles décisionnels des entités actives que nous avons présentés dans le troisième chapitre. Cette évaluation est menée au travers d'un ensemble de de simulations.

En premier lieu, nous décrivons le SPBS qui a servi de support pour l'évaluation. Cette description se focalise sur les caractéristiques d'un SPBS ainsi que les hypothèses prises en compte pour son fonctionnement.

En second lieu, nous présentons et analysons les différents résultats de simulations obtenus :

- nous montrons l'influence des différents paramètres qui définissent le système de pilotage hétérarchique proposé,
- nous présentons l'apport des mécanismes d'apprentissage en terme d'amélioration continue des performances par rapport à des approches d'ordonnancement classiques.

Enfin, une étude plus approfondie sur le comportement du système de pilotage, appliquée à un exemple étendu, est présentée dans le dernier volet de ce chapitre.

1. Simulations préliminaires

La première partie de ce chapitre est consacrée à l'étude de l'influence des différents paramètres qui définissent le système de pilotage hétérarchique proposé. Il est judicieux, compte tenu de la multitude de ces paramètres, de réaliser des simulations préliminaires. La détermination des 'bonnes valeurs' des différents paramètres a un double objectif : observer l'effet de ces paramètres en faisant varier leurs valeurs et observer aussi l'efficacité des algorithmes que nous avons définis en fonction de ces paramètres.

1.1. Méthodologie de simulation

1.1.1. Cas d'étude

Nous avons simulé un SPBS constitué d'un ensemble de ressources et de produits. Les produits circulent dans le SPBS afin de réaliser une gamme de tâches. Dans ce modèle, les liaisons entre les différentes ressources sont modélisées par des temps de transfert et chaque ressource dispose d'une file d'attente de capacité infinie. La Figure 80 illustre un exemple avec trois ressources :

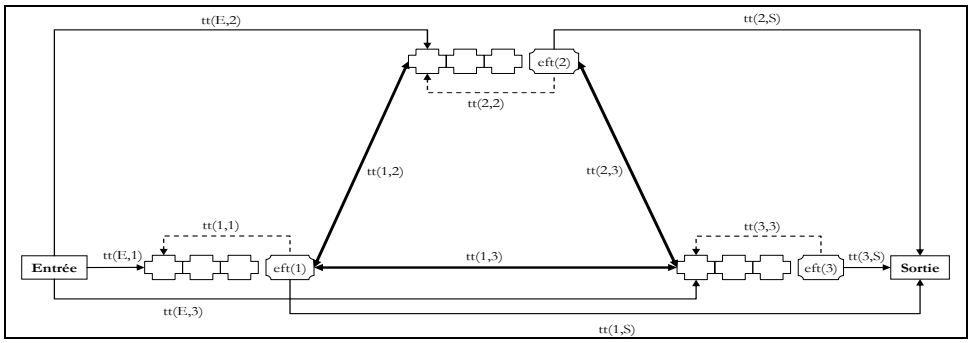


Figure 80- Exemple simulé avec trois ressources[70]

Nous avons simulé un exemple avec trois ressources (R1, R2 et R3). Chaque produit est associé à une gamme composée de trois tâches (T1→T2→T3). Les temps opératoires et de transfert sont mentionnés respectivement dans les tableaux Tab. 41 et Tab. 42 :

	T1	T2	T3
R1	1	2	-
R2	-	3	4
R3	5	-	6

Tab. 41 Temps opératoires (3 ressources, 3 tâches)

→	R1	R2	R3	Sortie
Entrée	2	3	4	-
R1	1	5	6	7
R2	5	1	8	9
R3	6	8	1	10

Tab. 42 Temps de transfert (3 ressources)

Dans cet exemple, chaque ressource est capable d'exécuter deux tâches. Nous avons volontairement choisi d'utiliser des temps opératoires et de transfert différents les uns des autres : cette première hypothèse permettra une application 'plus simple' des différentes stratégies. La deuxième hypothèse concerne le nombre de produits présents simultanément dans le système. En effet, deux critères doivent être pris en compte lors de la détermination de ce paramètre :

- un nombre réduit de produits dans le SPBS provoquera une sous-exploitation des capacités

[70] Rappelons les deux hypothèses relatives aux temps de transfert : tt(k,k')= tt(k',k) et tt(k,k)=1.

réelles du système. Ce cas de figure peut inhiber l'effet de l'apprentissage des produits et/ou des ressources,

- un grand nombre de produits dans le système aura pour conséquence la saturation du système. Sous cette condition, l'apprentissage des entités actives n'apportera rien car le système atteint rapidement ses limites.

Pour ces deux raisons, il est nécessaire de trouver un compromis entre ces deux cas extrêmes. Pour cet exemple, nous avons fixé ce paramètre (NPS) à 10. Dans le § 1.3.7, nous étudions plus en détail l'effet de ce paramètres. La partie suivante présente les différentes hypothèses prises en compte dans les simulations. Ces hypothèses englobent notamment les stratégies utilisées par les produits et les ressources lors de l'exécution de leur processus décisionnel, le mode d'alimentation du système avec des produits ou encore les perturbations prises en compte.

1.1.2. Indicateurs de performances

Le fonctionnement de cet exemple de SPBS donne lieu à trois indicateurs de performances interdépendants et relatifs, respectivement, au SPBS, aux ressources et aux produits :

- la performance globale d'un SPBS est quantifiée par le nombre de produits finis récupérés dans l'aire de stockage sur un intervalle de temps donné (en l'occurrence la durée de simulation). Cet indicateur permet d'évaluer l'efficacité du processus décisionnel global des entités actives,
- la performance PEft(k,h,NbD) d'un cycle de décisions (d'indice h) effectué par une ressource (d'indice k) et composé de NbD décisions est mesurée à l'aide du nombre de produits finis ayant quitté le SPBS pendant la réalisation de ce cycle. Cet indicateur permet d'évaluer la pertinence des décisions prises par chaque ressource,
- la performance PEmp(i) relative à un produit fini (d'indice i) est égale à la somme des temps d'attente intermédiaires passés sur les différentes files d'attente avant la réalisation des NbT tâches (voir chapitre III, § 2.4). Cet indicateur permet d'évaluer la pertinence des décisions prises pour chaque produit.

1.1.3. Stratégies mises en œuvre dans le processus décisionnel des entités actives

Le modèle de pilotage hétérarchique proposé implique que les entités actives utilisent des stratégies pour élaborer leur processus décisionnel. En ce sens, nous avons défini 11 stratégies (NSf=11) qui permettent aux ressources de sélectionner un produit parmi ceux présents dans sa file d'attente. Pour ce faire, nous avons retenu cinq critères (NCf=5) relatifs aux produits (date d'arrivée, nombre de tâches restantes, indice) et aux caractéristiques propres d'une ressource (durée ou fin du traitement). Une onzième stratégie, qui ne se base sur aucun critère (choix aléatoire), est rajoutée à cette liste, voir tableau Tab. 43 :

	Critère (Cf(x))		Stratégie (SEft(d))	
	x	désignation	d	désignation : sélectionner…
Ensemble des critères (ECF)	1	date d'arrivée du produit	1	le produit qui arrive le premier sur la file d'attente (Premier arrivé premier servi FIFO)
			2	le produit qui arrive le dernier sur la file d'attente (Dernier arrivé premier servi LIFO)
	2	nombre de tâches déjà réalisées par chaque produit	3	le produit qui a le moins de tâches restantes
			4	le produit qui a le plus de tâches restantes
	3	temps de traitement	5	le produit qui nécessite le plus de temps de traitement
			6	le produit qui nécessite le moins de temps de traitement

	4	indice du produit	7	le produit qui a le plus petit identificateur
			8	le produit qui a le plus grand identificateur
	5	date de fin du traitement[71]	9	le produit dont le traitement se termine le plus tôt
			10	le produit dont le traitement se termine le plus tard
	-	aléatoire	11	un produit au hasard

Tab. 43 Liste des stratégies utilisées par les ressources

Concernant les produits, nous avons établi une liste qui contient 11 stratégies (NSm=11). Ces stratégies sont basées exclusivement sur des critères relatifs aux caractéristiques des différentes ressources candidates à la réalisation d'une tâche donnée. Signalons par ailleurs que nous avons défini un critère qui agrège deux autres critères (le critère 3 est une agrégation des critères 1 et 2, voir tableau Tab. 44).

	Critère (Cm(y))		Stratégie (SEmp(p))	
	y	désignation	p	désignation : sélectionner…
Ensemble des critères (ECM)	1	temps de traitement	1	la ressource qui minimise le temps de traitement
			2	la ressource qui maximise le temps de traitement
	2	proximité de la ressource	3	la ressource la plus proche
			4	la ressource la plus lointaine
	3	proximité de la ressource et le temps de traitement	5	la ressource qui minimise le temps de traitement et le temps de transfert
			6	la ressource qui maximise le temps de traitement et le temps de transfert
	4	le taux d'occupation d'une file d'attente	7	la ressource la moins chargée
			8	la ressource la plus chargée
	5	fin de l'ensemble des traitements[72]	9	la ressource qui promet de finir le traitement le plus tôt
			10	la ressource qui promet de finir le traitement le plus tard
	-	aléatoire	11	une ressource au hasard

Tab. 44 Liste des stratégies utilisées par les produits[73]

1.1.4. Mode d'entrée des produits dans le SPBS

Afin de simuler le fonctionnement de cet exemple, il est primordial de définir le flux d'entrée des produits dans le SPBS. En effet, dans notre exemple, trois modes d'entrées peuvent être distingués :

- entrée d'un seul produit à la fois dans le système selon une cadence constante (régulière) ou aléatoire au cours du temps,
- entrée des produits par lot selon une cadence constante ou aléatoire au cours du temps,
- entrée des produits de façon à garantir un nombre constant de produits dans le système (par exemple, à chaque instant, il y a 20 produits dans le système). Autrement dit, à chaque fois que n produits sortent du SPBS, n nouveaux produits y entrent.

Nous avons opté pour le troisième mode décrit ci-dessus. Ce choix permet d'éliminer une

[71] Il s'agit de minimiser la somme (date d'arrivée + temps de traitement).
[72] Traitement de tous les produits présents dans une file d'attente.
[73] Les stratégies relatives aux ressources et aux produits que nous avons définies sont 'générales'. En effet, ces stratégies sont applicables indépendamment de l'exemple étudié. Autrement dit, pour les produits, nous aurions pu définir la stratégie : 'Aller sur la ressource qui est capable de réaliser deux tâches consécutives'. Or, l'applicabilité de cette stratégie dépend des capacités de chaque ressource et n'est possible que s'il existe une ressource capable de réaliser deux tâches consécutives. Par contre, la stratégie 'Aller sur la ressource qui minimise le temps opératoire' est une stratégie générale. Cette propriété confirme la généricité de notre approche.

tâche supplémentaire qui consiste à régler la fréquence et la taille du lot des produits en entrée[74].

1.1.5. Perturbations prises en compte

Nous nous sommes interesssés aux perturbations liées aux entités fixes de traitement en faisant abstraction des perturbations d'origines externes (par exemple la rupture des stocks ou les commandes urgentes). Avant de préciser les perturbations que nous avons simulées, nous établissons dans la partie suivante une typologie des perturbations qui affectent le fonctionnement des entités fixes de traitement. En effet, nous distinguons un(e) :

- arrêt complet et définitif de l'eft : aucune tâche ne peut être réalisée sur cette eft,
- arrêt complet mais temporaire de l'eft : après un certain temps, l'eft retrouve son fonctionnement normal,
- dégradation définitive ou temporaire de la flexibilité de l'eft : une ou plusieurs tâche(s) ne peu(ven)t plus être réalisée(s) sur cette eft et ceci durant un intervalle de temps infini ou fini,
- amélioration définitive ou temporaire de la flexibilité de l'eft : une ou plusieurs nouvelle(s) tâche(s) peu(ven)t être réalisée(s) sur cette eft et ceci durant un intervalle de temps infini ou fini,
- augmentation sensible du temps opératoire de l'eft : les emp sont traitées sur cette eft mais avec une durée beaucoup plus longue. Ce type de perturbations peut avoir deux formes :
  - augmentation constante et finie : dans ce cas, le temps opératoire relatif à une tâche est multiplié par un facteur f (f très grand) pendant un intervalle de temps [a,c], voir Figure 81,
  - augmentation progressive et finie (dérive) : cette augmentation du temps opératoire est observée sur un intervalle [a,b]. L'instant b (retour à la normale) peut varier dans l'intervalle [a,c] d'une dérive à une autre. C'est le cas par exemple d'une machine qui tombe lentement en panne et qui est par la suite réparée rapidement à partir de l'instant b (dans ce cas b>(a+c)/2)).

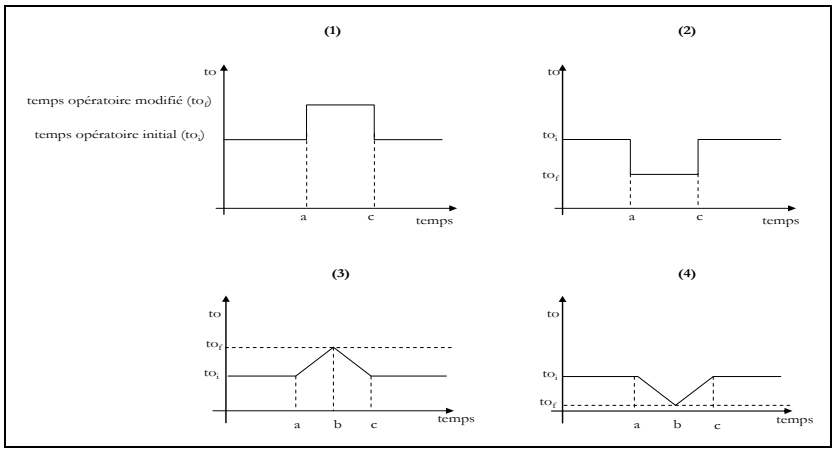


Figure 81- Typologie des perturbations

- diminution sensible du temps opératoire[75] : il s'agit d'une amélioration (suite à des

[74] Nous verrons par la suite qu'il y a d'autres paramètres plus importants à fixer dans notre modèle.

[75] Ce type de perturbations est très fréquent dans le cas des systèmes de transport (bus, tramway, …). En effet, l'avance d'un bus est considérée comme une perturbation. Ce même problème se pose aussi pour certains types de SPBS notamment les chaînes d'assemblage automobiles : l'arrivée avancée d'un composant peut engendrer une désynchronisation de l'ensemble de la chaîne d'assemblage.

modifications techniques par exemple) de la vitesse d'exécution d'un eft. Nous distinguons, comme pour le cas précédent, deux types de diminutions du temps opératoires :

- diminution constante et finie du temps opératoire illustrée par le cas-2 de la Figure 81,
- diminution progressive et finie du temps opératoire avec un retour progressif à la normale, voir Figure 81, cas-4.

Dans notre travail, nous avons pris en considération deux types de perturbations : l'augmentation et la diminution constantes et finies du temps opératoire. Ce choix permet d'observer, grâce à des simulations, l'effet d'une dégradation ou d'une amélioration de la vitesse de traitement d'une entité fixe de traitement sur un intervalle de temps fini. En outre, ce type de perturbations peut être intégré dans le simulateur à événements discrets que nous avons développé.

Après avoir présenté notre méthodologie de simulation, la partie suivante est consacrée à la présentation et à l'analyse des résultats obtenus. Il s'agit, en premier lieu, d'observer l'influence des différents paramètres liés au processus décisionnel des entités actives : l'importance des stratégies adoptées (SEmp et SEft), les facteurs d'oubli (λ_f et λ_m), les taux de mutation (σ_f et σ_m), les seuils de performance (s_f et s_m) et les signaux de renforcement (r_f et r_m). Ensuite, nous analysons le comportement des différents algorithmes décisionnels et d'apprentissage en fonctionnement normal et perturbé.

1.2. Étude de l'influence des paramètres liés au processus décisionnel des produits

Pour pouvoir évaluer les performances obtenues suite à l'application des mécanismes décisionnels définis dans le troisième chapitre, il est nécessaire d'avoir un cadre de référence par rapport auquel ces performances peuvent être comparées. Or, étant donné la spécificité de l'exemple que nous traitons (polyvalence des ressources, prises en compte des temps de transfert, mode d'entrée des produits ou encore l'occurrence de perturbations), il est difficile de trouver des références, dans la littérature scientifique correspondante, par rapport auxquelles nous pouvons comparer les résultats obtenus. Alors, nous avons effectué une comparaison avec des heuristiques largement utilisées en ordonnancement (SPT, LPT, FIFO, …) et qui sont mises en œuvre dans notre travail au travers des différentes stratégies que nous avons définies (voir tableaux Tab. 43 et Tab. 44). Nous pensons que nos résultats peuvent représenter un 'benchmark' pour d'autres travaux ultérieurs.

Par ailleurs, et pour bien observer l'influence des différents paramètres, nous avons opté pour une double simulation : une simulation en fonctionnement normal (sans perturbation **SP**) et une autre en mode perturbé (avec perturbation **AP**). Dans le deuxième type de simulation, nous avons simulé une perturbation relative à la ressource R2 qui se produit entre les instants 1000 et 1400 (sachant que la durée totale de la simulation est de 4000 UTs). Cette perturbation est assimilée à une augmentation de 50 UTs des temps opératoires de la ressource R2 (relatifs à l'exécution des tâches t(2) et t(3)).

1.2.1. Importance des différentes stratégies

Nous avons commencé par observer les résultats donnés par chaque stratégie, appliquée séparément, puis ceux obtenus en appliquant les différents modes d'apprentissage. Cette observation permet de nous donner une idée sur l'importance de chaque stratégie. Précisons aussi que les premières simulations ont été effectuées en tenant compte des données mentionnées dans le tableau Tab. 45 :

entité active	produit	ressource
facteur d'oubli	λ_m=20	λ_f=20 (taille d'un cycle NbD=5)
taux de mutation	σ_m=0.1	σ_f=0.1
seuil de performance	s_m=2	s_f=5
signal de renforcement	$(+r_m,0,-r_m)=(+1,0,-1)$[76]	$(+r_f,0,-r_f)=(+1,0,-1)$

Tab. 45 Données utilisées pour les premiers tests de simulation

Pour évaluer l'importance de chaque stratégie, nous avons réalisé des tests de simulation de trois types :

- application d'une stratégie unique (les stratégies 1 à 11) par tous les produits, voir bloc 1 du tableau Tab. 46,
- simuler le comportement du système en utilisant les différents mécanismes d'apprentissage proposés dans ce travail (apprentissage des ressources uniquement (AR), apprentissage des produits uniquement (AP) et un apprentissage mixte des ressources et des produits (AMix)), voir bloc 2 du tableau Tab. 46,
- application aléatoire d'une stratégie Ran (ce qui est différent par rapport au choix aléatoire d'une ressource candidate (stratégie 11)), voir bloc 3 du tableau Tab. 46.

Graphiquement, cette disparité des performances obtenues (nombre de produits finis) pour les différents modes décisionnels est représentée sur la Figure 82 :

	mode décisionnel	SP	AP
Bloc 1	Stratégie 1	997	975
	Stratégie 2	360	292
	Stratégie 3	997	797
	Stratégie 4	360	303
	Stratégie 5	997	975
	Stratégie 6	360	292
	Stratégie 7	1387	1283
	Stratégie 8	575	460
	Stratégie 9	1373	1289
	Stratégie 10	572	458
	Stratégie 11	727	590
Bloc 2	AR[77]	1253	1184
	AP	1046	927
	AMix	803	662
Bloc 3	Ran	635	534

Tab. 46 Nombre de produits finis obtenus selon le mode décisionnel pris en compte

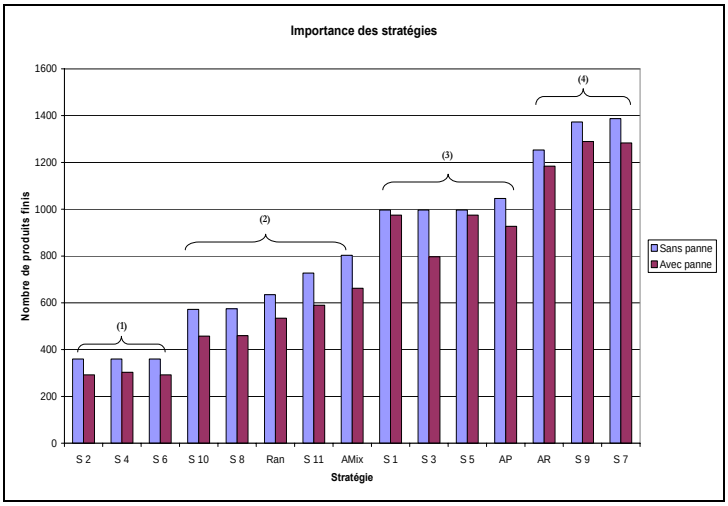


Figure 82- Variation des performances selon le mode décisionnel pris en compte

L'analyse de ces résultats mène aux constats suivants :

- une première appréciation des différentes stratégies peut être faite. En effet, en fonction des performances globales obtenues, nous pouvons distinguer quatre catégories de stratégies :
  - celles qui donnent de très bonnes performances (stratégies 7 et 9),
  - celles qui donnent des performances acceptables (stratégies 1, 3 et 5)

[76] +1 pour une récompense, -1 pour une pénalité, 0 sinon.
[77] Pour cette simulation, tous les produits adoptent la stratégie 7.

- celles qui donnent des performances médiocres (stratégies 8 et 10),
- enfin celles qui donnent des performances globales très médiocres[78] (2, 4 et 6).

- l'application aléatoire des stratégies (bloc 3) ou le choix aléatoire d'une ressource candidate (stratégie 11) donne des performances acceptables comparées à l'adoption d'une stratégie unique. Par conséquent, ce constat montre que l'adoption d'une seule stratégie n'est pas adéquate dans une logique d'amélioration continue des performances et qu'il faut varier les stratégies : c'est ce que permettent les différents algorithmes proposés dans ce travail,
- a priori, les résultats mentionnés dans le tableau Tab. 46 peuvent paraître surprenants. En effet, l'adoption de quelques stratégies particulières (en l'occurrence la stratégie 7 et 9) donne les meilleures performances en terme de nombre de produits finis et ces performances concernent les deux modes de fonctionnement normal et perturbé. Une question se pose alors : pourquoi proposer les différents mécanismes d'apprentissage puisque l'adoption d'une stratégie donne les meilleurs résultats ? Les parties suivantes donnent des éléments de réponse à cette question,
- l'adoption d'un apprentissage mixte donne des performances médiocres. Pourtant ce mode décisionnel est *censé* permettre l'obtention d'excellents résultats. Nous verrons par la suite que ce point est lié au point précédent.

Première conclusion (C1) : pour l'ensemble des produits, les performances globales obtenues varient en fonction de la stratégie adoptée : cette variation permet de qualifier chaque stratégie (bonne, mauvaise, …). La question qui se pose alors : dans une optique d'amélioration continue des performances, est-ce qu'il faut utiliser uniquement les bonnes stratégies ?

Après avoir évalué la qualité de chaque stratégie et l'apport des différents modes d'apprentissage (apprentissage ressource, produit et mixte), la partie suivante étudie l'influence des principaux paramètres de notre modèle sur la performance globale.

1.2.2. Influence du facteur d'oubli

La notion du facteur d'oubli est centrale dans les mécanismes d'apprentissage que nous avons proposés. Il est évident alors de commencer par observer l'influence de ce paramètre. Pour ce faire, nous avons réalisé des simulations en faisant varier ce paramètre (les valeurs prises par λ_m varient entre 3 et 30). Notons par ailleurs que, pour l'ensemble de ces simulations, l'apprentissage des ressources est désactivé (chaque ressource adopte la stratégie 1 (FIFO)). Les résultats obtenus sont donnés dans le tableau Tab. 47 :

En examinant ces résultats numériques, deux constats essentiels peuvent être faits :

- les performances obtenues en terme de nombre de produits finis ne sont pas identiques : nous constatons que, pour des valeurs réduites du facteur d'oubli ($\lambda_m \leq 10$), les performances sont les meilleures. Ce constat est prévisible puisque ce sont les derniers produits qui reflètent une image réaliste de l'état global du système,
- lorsque le facteur d'oubli augmente, la performance du système évolue d'une manière non rationnelle (fluctuation, voir Figure 83). Une explication possible de cette variation consiste à dire qu'une grande valeur du facteur d'oubli donne la possibilité, pendant le processus décisionnel, de prendre en compte de 'bonnes matrices (M(emp(i)))' mais qui étaient obtenues dans un contexte différent de contexte actuel du SPBS. Par conséquent, les stratégies fournies par ces matrices donnent logiquement des mauvaises performances mais peuvent également

[78] En toute logique, ces quatre qualifications sont concordantes avec la définition des stratégies donnée dans Tab. 44.

donner de bonnes performances, d'où un *phénomène de compensation* qui peut expliquer en partie cette fluctuation.

λ_m	SP	AP	λ_m	SP	AP
3	1058	925	15	1055	871
4	1091	865	16	1049	876
5	1047	899	17	1056	855
6	1103	979	18	1057	908
7	1085	885	19	1078	871
8	1098	958	20	1046	927
9	1095	870	21	1036	863
10	1124	873	22	1034	897
11	1084	936	23	1028	924
12	1053	834	24	1047	844
13	1095	949	25	1063	891
14	1056	864	30	1018	811

Tab. 47 Nombre de produits finis obtenus en fonction du facteur d'oubli (λ_m) pris en compte

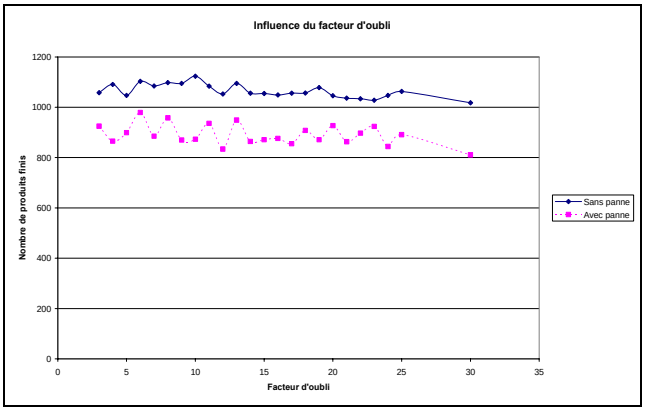


Figure 83- Évolution de la performance du système en fonction du facteur d'oubli (λ_m)

Deuxième conclusion (C2) : le facteur d'oubli a un impact important dans le processus d'apprentissage d'un produit. Plus la valeur de ce paramètre est réduite, plus les mécanismes d'apprentissage sont efficaces.

Pour la suite des simulations, nous avons fixé λ_m à 6. Notons en outre que l'application des mécanismes génétiques sur un nombre réduit de matrices relatives aux différents produits permet d'accélérer considérablement les simulations informatiques (réduire le temps CPU).

1.2.3. Influence du seuil de performance

Les mécanismes d'apprentissage par renforcement proposés dans ce travail intègrent une évaluation continue des différentes stratégies utilisées dans le processus décisionnel. Cette évaluation se base sur la notion de seuil de performance (s_m, voir chapitre III, § 4.3.2.3). Rappelons que la comparaison entre la performance réalisée, pour une tâche donnée, par une stratégie et ce seuil détermine la valeur qui sera donnée au signal de renforcement (récompense ou pénalité) et qui sera attribué à cette stratégie. Dans les tests réalisés, nous avons fait varier la valeur de ce seuil entre 0.1 et 10. Les résultats numériques obtenus sont donnés dans le tableau Tab. 48 :

En observant les résultats mentionnés ci-dessus, trois remarques principales peuvent être faites :

- pour une plage de valeurs (variant entre 0.1 et 0.9) du seuil de performance s_m, les performances obtenues sont identiques et médiocres,
- à partir d'une certaine valeur de s_m (s_m=1), les performances du système s'améliorent pour atteindre une valeur maximale (obtenue pour s_m=6),
- au delà de cette valeur, les performances commencent à décroître sensiblement.

En résumé, nous distinguons trois zones sur la Figure 84. Dans la première zone ($s_m \in [0.1, 0.9]$), les mécanismes d'évaluation sont incapables d'apprécier (ou plutôt apprécient mal) la qualité des stratégies adoptées (via leurs performances). Dans ce cas de figure, les stratégies sont continuellement récompensées et/ou pénalisées. Dans la deuxième zone ($s_m \in [1,6]$), les performances s'améliorent remarquablement. Cette amélioration est rendue possible grâce à l'extension progressive de l'intervalle [PMEmp-s_m, PMEmp+s_m] (voir chapitre III, § 4.3.2.3) qui

permet d'estimer convenablement la qualité des différentes stratégies. Enfin, dans la troisième zone (s_m>6), les performances décroissent. Pire encore, la différence entre les performances obtenues pour le mode de fonctionnement normal ainsi que le mode perturbé est réduite. Ceci peut être expliqué par le fait que les mécanismes d'évaluation perdent considérablement de leur pertinence. En effet, en élargissant 'trop' l'intervalle [PMEmp-s_m, PMEmp+s_m], il n'est plus possible de récompenser les bonnes stratégies ou de pénaliser les mauvaises.

s_m	SP	AP	s_m	SP	AP
0.1	908	765	2	1103	979
0.2	908	765	3	1158	980
0.3	908	765	4	1155	948
0.4	908	765	5	1414	948
0.5	908	765	6	1394	1293
0.6	908	765	7	1394	1191
0.7	908	765	8	1318	1234
0.8	908	765	9	1314	1230
0.9	908	765	10	1316	1238
1	1033	859			

Tab. 48 Nombre de produits finis obtenus en fonction du seuil de performance (s_m)

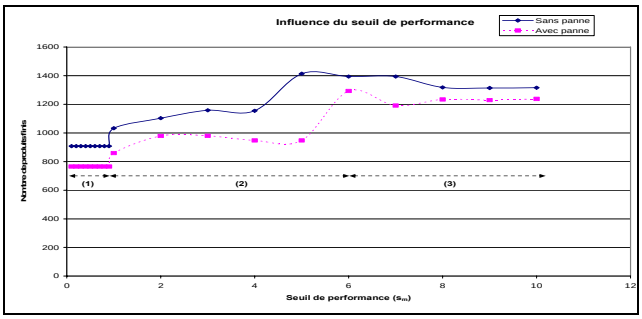


Figure 84- Influence du seuil de performance sur les performances globales (s_m)

Troisième conclusion (C3) : le seuil s_m joue un rôle central dans le succès ou l'échec des mécanismes de renforcement et par conséquent dans l'efficacité des mécanismes d'apprentissage. Il est primordial[79] *de déterminer la bonne valeur de ce seuil.*

Une fois le seuil de performance fixé (s_m=6), nous nous intéressons dans la partie suivante à l'impact du taux de mutation sur l'efficacité des mécanismes d'évaluation et d'apprentissage proposés.

1.2.4. Influence du taux de mutation

Dans les mécanismes génétiques, le taux de mutation est une probabilité d'un changement localisé d'un gène (d'où les valeurs de ce taux varient entre 0 et 1). Dans ce sens, nous avons réalisé des simulations en faisant varier le taux de mutation (σ_m) entre ces deux valeurs (avec à chaque fois un pas de 0.1). Les résultats obtenus sont donnés dans le tableau Tab. 49. Encore une fois, les résultats obtenus confirment le fait qu'il suffit de faire varier un paramètre (en l'occurrence le taux de mutation) pour obtenir des performances totalement différentes. Par ailleurs, les deux principaux constats méritent d'être mis en avant :

- la plus mauvaise performance (pour le mode fonctionnement normal et perturbé) est observée pour un taux de mutation nul. Ce résultat est attendu puisqu'en absence de mutation, les opérations de croisement ne sont pas suffisantes pour assurer la diversification des données, d'autant plus que le nombre de tâches (égal à 3 et qui représente également la dimension de la matrice M(emp(i))) est réduit. Dans ce cas, il existe un risque important d'adopter les mêmes stratégies durant toute la simulation,
- pour les deux modes de fonctionnement, la meilleure performance est obtenue en fixant σ_m à 0.1. Il est difficile de trouver une explication rationnelle à ce constat. Néanmoins, nous pensons que l'opération de mutation est nécessaire mais selon une proportion réduite. Ceci est confirmé par les performances obtenues pour des taux de mutation supérieurs à 0.1 (voir

[79] Nous sommes persuadés que chaque jeu de données (ou scénario) possède une valeur adéquate de s_m qu'il faut déterminer.

Figure 85). En effet, l'opérateur de mutation tel que nous l'avons défini consiste à introduire aléatoirement une stratégie indépendamment de son coefficient d'évaluation. Donc il est évident que l'introduction récurrente (taux de mutation élevé) des stratégies induit une baisse des performances globales du système.

σ_m	SP	AP
0	1247	980
0.1	1394	1293
0.2	1353	1135
0.3	1382	1175
0.4	1322	1046
0.5	1288	1143
0.6	1253	1064
0.7	1364	1054
0.8	1341	1121
0.9	1361	1080
1	1287	998

Tab. 49 Nombre de produits finis obtenus en fonction du taux de mutation (σ_m)

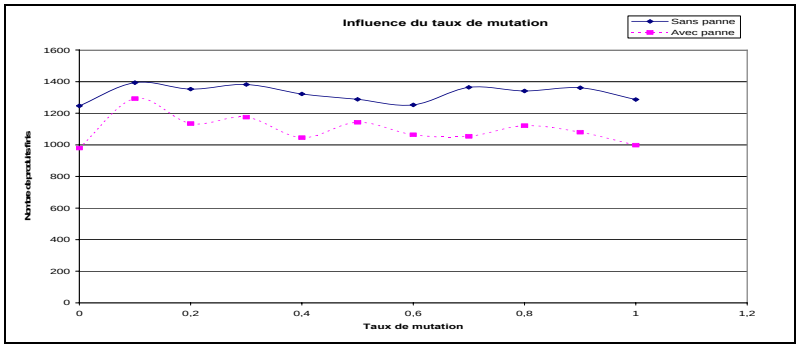


Figure 85- Évolution des performances globales en fonction du taux de mutation (σ_m)

Quatrième conclusion (C4) : Nous avons montré que l'opértion de mutation est nécessaire mais selon une probabilité réduite (σ_m=0.1). Ce constat concorde avec le taux de mutation communément pris en compte dans les algorithmes génétiques classiques (recherhce opérationnelle) [Sevaux, 04].

1.2.5. Influence du signal de renforcement

Le dernier paramètre dont nous avons jugé utile d'observer l'impact est le signal de renforcement (r_m). Pour ce faire, nous avons réalisé quatre catégories de tests. Dans la première catégorie, une stratégie est récompensée ou pénalisée avec le même signal. Dans la deuxième catégorie, la pénalisation des stratégies est plus importante que leur récompense ($(+r_m,0,-r'_m)$ avec $r'_m > r_m$). A contrario, la récompense des stratégies est plus importante que leur pénalisation ($(+r_m,0,-r'_m)$ avec $r'_m < r_m$) pour la troisième catégorie. Enfin, dans la dernière catégorie, le signal de renforcement prend une valeur aléatoire (qui varie entre 1 et 10) à chaque opération d'évaluation, voir tableau Tab. 50 :

r_m	SP	AP
1	1394	1293
2	1394	1293
3	1394	1293
4	1394	1293
5	1394	1293
6	1394	1293
7	1394	1293
8	1394	1293
9	1394	1293
10	1394	1293
(+2,0,-4)	1394	1293
(+4,0,-2)	1394	1293
Aléatoire	1317	1112

Tab. 50 Nombre de produits finis obtenus en fonction de r_m

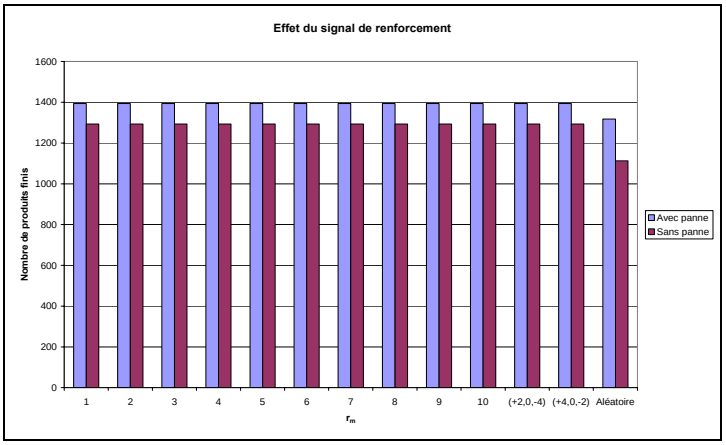


Figure 86- Évolution des performances globales en fonction du signal de renforcement (r_m)

Les trois premières catégories de tests donnent des performances identiques (voir Figure 86).

Ce constat démontre que tant que la récompense et la pénalité sont identiques, alors les performances obtenues sont égales. Par contre, si une variation est opérée sur r_m au cours des cycles d'évaluation successifs, alors nous observons une dégradation de ces performances. L'origine de cette dégradation peut s'expliquer par le fait qu'une stratégie peut être parfois peu récompensée et parfois trop pénalisée ou l'inverse.

Cinquième conclusion (C5) : le succès du processus d'évaluation des différentes stratégies est essentiellement lié à l'adoption d'un signal de renforcement identique tout au long de ce processus.

L'ensemble des ces cinq conclusions permettent de répondre à la question posée précédemment (voir § 1.2.10) : l'ajustement des différents paramètres permet d'optimiser le fonctionnement des algorithmes proposés dans ce travail et par conséquent l'amélioration des performances obtenues. Il est intéressant par ailleurs de réaliser une analyse plus approfondie relative au comportement des mécanismes d'apprentissage des produits. L'analyse que nous avons effectuée concerne le mode de fonctionnement normal et le mode perturbé et fait l'objet de la partie suivante. Les perturbations que nous avons simulées sont de deux types : une augmentation du temps opératoire (panne) et une amélioration du temps opératoire.

1.2.6. Analyses du comportement des mécanismes d'apprentissage

1.2.6.1 Comportement en fonctionnement normal et perturbé (dégradation du temps opératoire)

En premier lieu, nous avons observé l'évolution du temps d'attente total (égal à la somme des temps d'attente intermédiaires pour les 3 tâches) des différents produits finis. Cette évolution est observée en appliquant, premièrement, les mécanismes d'apprentissage et deuxièmement en appliquant la stratégie 7 (qui est une bonne stratégie). Nous déduisons de cette comparaison que :

- l'application de la stratégie 7 donne des temps d'attente constants : cette constance est imputée à l'orientation des produits vers les mêmes ressources (phénomène d'oscillations permanentes), voir Figure 87,
- le fonctionnement avec apprentissage ne permet pas cette constance pour les temps d'attente. En effet, nous remarquons sur la Figure 88 la présence de deux phases : la première phase est caractérisée par une grande variation des temps d'attente totaux. L'initialisation du processus décisionnel (choix aléatoire des stratégies) et des mécanismes d'apprentissage est à l'origine de cette variation. Pendant la deuxième phase, nous remarquons que l'évolution des temps d'attente est 'stabilisée'. Les mécanismes d'apprentissage ont permis cette stabilisation grâce au processus d'évaluation continue des stratégies adoptées.

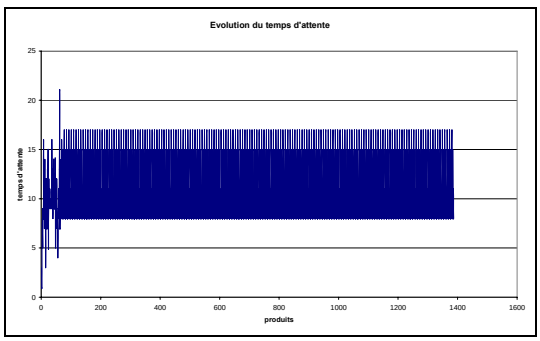


Figure 87- Évolution des temps d'attentes (S7, sans panne)

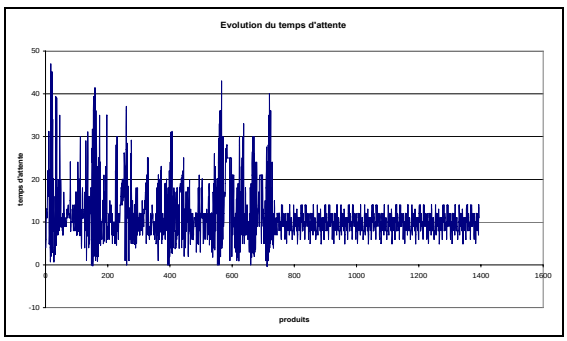


Figure 88- Évolution des temps d'attentes (apprentissage, sans panne)

En second lieu, nous avons examiné un ensemble d'indicateurs relatifs aux temps d'attente (valeur minimale, maximale et moyenne). Ces indicateurs ont été relevés pour un fonctionnement sans panne (SP) et avec panne (AP). Par ailleurs, la comparaison est faite entre un mode dynamique (mécanismes d'apprentissage) et un mode statique (adoption exclusive de la stratégie 7, voir Figure 89 et Figure 90).

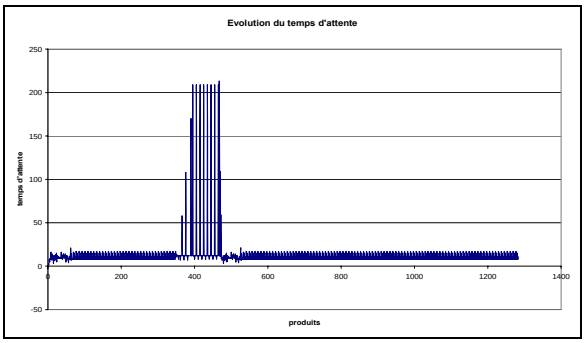


Figure 89- Évolution des temps d'attentes (S7, avec panne)

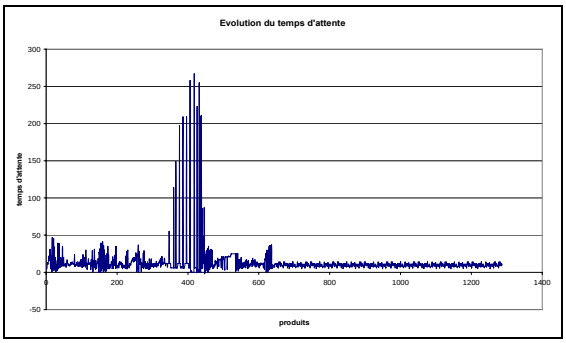


Figure 90- Évolution des temps d'attentes (Apprentissage, avec panne)

L'application de la stratégie 7 présente l'avantage de minimiser les temps d'attentes minimaux, maximaux et moyens, et ceci pour le mode de fonctionnement normal ou perturbé, voir tableau Tab. 51. Or l'application des mécanismes d'apprentissage permet d'optimiser le nombre de produits finis (1394 avec l'apprentissage produit contre 1387 avec la stratégie 7). Nous estimons que, pour un SPBS, l'objectif le plus important est la maximisation du nombre de produits finis : cet objectif est mieux atteint grâce à l'application des mécanismes d'apprentissage.

	temps d'attente minimal	temps d'attente maximal	temps d'attente moyen
apprentissage produit (SP)	0	47	11,49
apprentissage produit (AP)	0	267	13,38
stratégie 7 (sans panne)	1	21	10,33
stratégie 7 (avec panne)	1	209	12,26

Tab. 51 Variation du temps d'attente (avec ou sans panne)

En troisième lieu, nous avons observé la répartition de l'utilisation des onze stratégies en appliquant les mécanismes d'apprentissage, c'est à dire, combien de fois chaque stratégie a été utilisée pour la réalisation des trois tâches. Cette répartition a été mesurée pour le fonctionnement normal et perturbé. Les résultats obtenus sont mentionnés dans les tableaux Tab. 52 et Tab. 53 :

Stratégie	T1	T2	T3	(T1,T2,T3)
S 1	41	63	78	182
S 2	7	8	50	65
S 3	167	57	65	289
S 4	5	5	37	47
S 5	1136	56	62	1254
S 6	4	5	42	51
S 7	20	341	790	1151
S 8	1	6	43	50
S 9	4	843	134	981
S 10	3	7	35	45
S 11	6	3	58	67

Tab. 52 Répartition de l'utilisation des différentes stratégies (apprentissage, sans panne)

Stratégie	T1	T2	T3	(T1,T2,T3)
S 1	41	57	60	158
S 2	7	11	40	58
S 3	167	24	35	226
S 4	5	5	55	65
S 5	1029	56	58	1143
S 6	4	5	37	46
S 7	20	174	77	271
S 8	1	7	36	44
S 9	4	938	794	1736
S 10	3	7	31	41
S 11	6	3	64	73

Tab. 53 Répartition de l'utilisation des différentes stratégies (apprentissage, avec panne)

L'examen de ces résultats montre qu'il y a trois stratégies qui sont majoritairement utilisées : S5, S7 et S9. En effet, pour la réalisation des tâches T1, T2 et T3, les stratégies 5, 9 et 7 sont respectivement les plus utilisées (plus de 60% chacune, voir Tab. 54). Dans ce sens, nous remarquons qu'il s'agit, pour chaque tâche, d'une 'bonne stratégie'.

		T1	T2	T3
Stratégie utilisée (%)	Sans panne	S 5 (82%)	S 9 (61%)	S 7 (56%)
	Avec panne	S 5 (81%)	S 9 (73%)	S 9 (61%)

Tab. 54 Récapitulation des stratégies utilisées majoritairement pour les 3 tâches

En outre, le pourcentage de l'utilisation des stratégies restantes est minoritaire ne dépassant pas les 4% (voir Figure 91 jusqu'à Figure 98). Ces deux constats montrent que les mécanismes d'apprentissage que nous avons développés 'détectent' les bonnes stratégies sans pour autant éliminer complètement les 'mauvaises stratégies'. Nous sommes persuadés que ce 'brassage formalisé' de bonnes et de mauvaises stratégies est à l'origine du succès de l'apprentissage des produits.

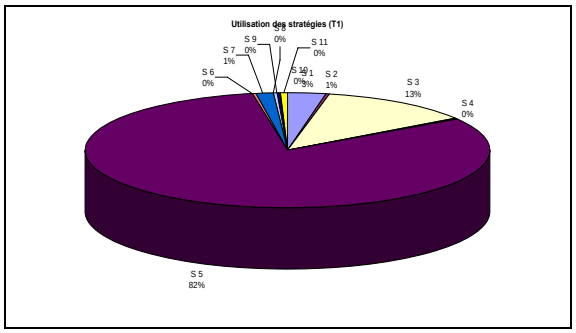


Figure 91- Utilisation (en pourcentage) des différentes stratégies (T1, sans panne)

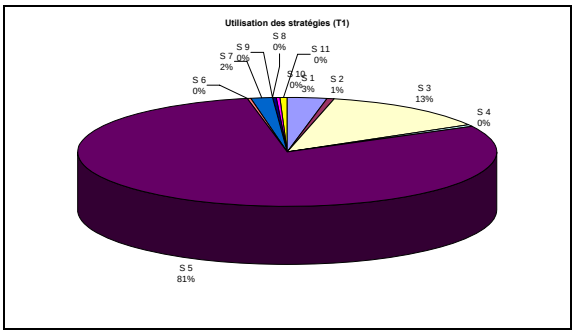


Figure 92- Utilisation (en pourcentage) des différentes stratégies (T1, avec panne)

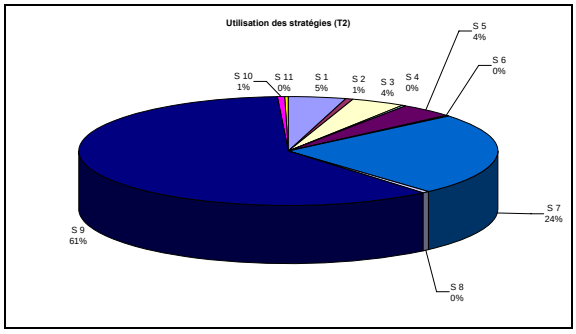


Figure 93- Utilisation (en pourcentage) des différentes stratégies (T2, sans panne)

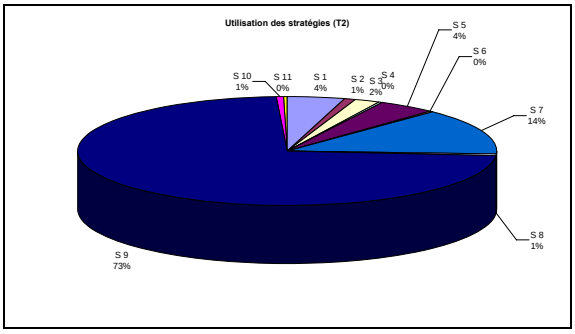


Figure 94- Utilisation (en pourcentage) des différentes stratégies (T2, avec panne)

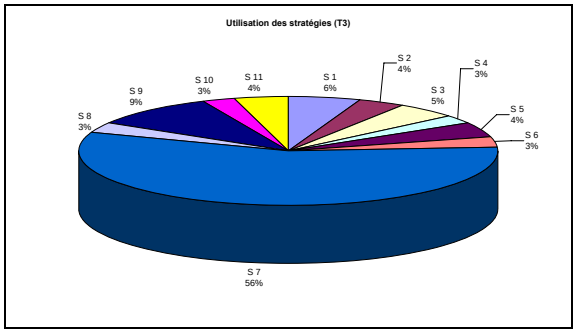


Figure 95- Utilisation (en pourcentage) des différentes stratégies (T3, sans panne)

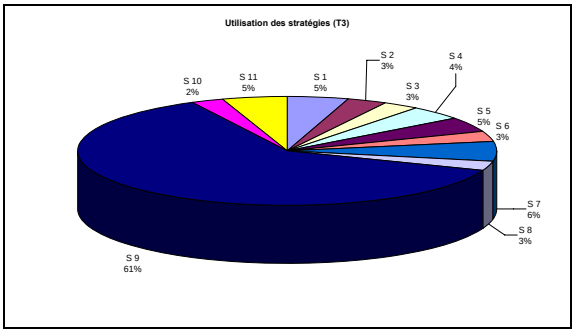


Figure 96- Utilisation (en pourcentage) des différentes stratégies (T3, avec panne)

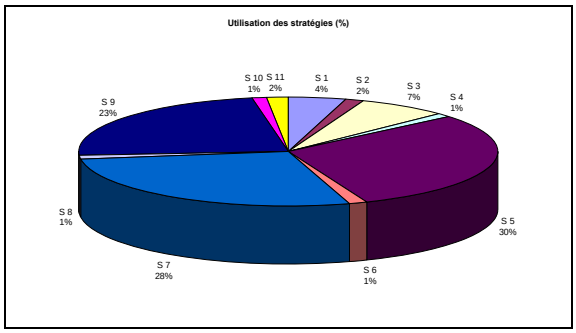


Figure 97- Utilisation (en pourcentage) des différentes stratégies (T1,T2,T3, sans panne)

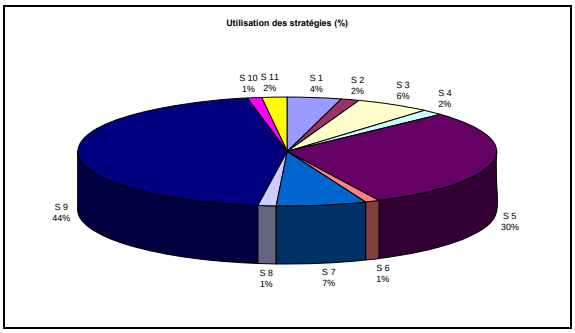


Figure 98- Utilisation (en pourcentage) des différentes stratégies (T1,T2,T3, avec panne)

Dans ce même cadre, il est intéressant d'observer le nombre d'utilisations de la ressource R2 pendant son fonctionnement perturbé. Rappelons d'abord que cette ressource est capable d'exécuter les tâches T2 et T3. Pour ce faire, une comparaison est effectuée en appliquant successivement les mécanismes d'apprentissage, la stratégie 7 et la stratégie 9. Les résultats obtenus montrent que, pour la réalisation de la tâche T2, R2 est moins utilisée lorsque nous adoptons la stratégie S9 : Cette conclusion est conforme avec la définition de cette stratégie (voir Tab. 44). Or, pour la réalisation de la tâche T3 (qui doit être réalisée juste après T2), les mécanismes d'apprentissage permettent de réduire le nombre d'utilisations de la ressource perturbée (voir tableau Tab. 55) : les produits acquiert une connaissance suite à la réalisation de T2 sur R2 et évitent de choisir la même ressource perturbée pour réaliser T3. C'est un constat très intéressant dans la mesure où il montre que les mécanismes d'apprentissage permettent 'd'éviter la ressource en fonctionnement perturbé'.

		apprentissage	S 7	S 9
nombre d'utilisations de la ressource R2	T2	178	147	3
	T3	697	737	871

Tab. 55 Nombre d'utilisations de la ressource en panne (R(2))

1.2.6.2 Comportement en fonctionnement perturbé (amélioration du temps opératoire)

Nous avons également observé le comportement des mécanismes d'apprentissage lorsque le temps opératoire d'une ressource est diminué (amélioration). Pour ce faire, nous avons envisagé les quatre scénarii suivants (la perturbation dure 400 UTs) :

- (1) : une seule ressource est perturbée (la ressource R2 est améliorée, respectivement R3 est améliorée),
- (2) : deux ressources sont perturbées identiquement (R2 et R3),
- (3) : deux ressources sont perturbées (R2 et R3), mais R2 est mieux améliorée (respectivement plus dégradées) que R3,
- (4) : deux ressources sont perturbées (R2 et R3), mais R3 mieux améliorée (respectivement plus dégradées) que R2,

Les résultats obtenus sont mentionnés dans le tableau Tab. 56

Type de perturbation	Ressource(s) concernée(s)	Stratégie 7	Stratégie 9	Apprentissage
perturbation=amélioration du temps opératoire	R2 (1)	1433	1445	1467
	R3(1)	1417	1422	1468
	R2 et R3 (2)	1428	1429	1476

	R2 et R3 (3)	1440	1443	1491
	R2 et R3 (4)	1411	1397	1456
Perturbation=dégradation du temps opératoire	R2 et R3 (2)	1258	1250	1150
	R2 et R3 (3)	1235	1231	1169
	R2 et R3 (4)	1190	1180	1192

Tab. 56 Apport de l'apprentissage sur plusieurs types de perturbations

Les bonnes performances obtenues par les mécanismes d'apprentissage proviennent de l'augmentation de l'utilisation de la ou les ressources perturbées (améliorées). Les nombres d'utilisations de ces ressources (R2 et R3), donnés dans les tableaux Tab. 57 et Tab. 58, confirment ce constat :

	R2		
	Tâche 2	Tâche 3	Nombre de produits finis
S7	227	927	1433
S9	100	1073	1445
Apprentissage	293	851	1467

Tab. 57 Nombre d'utilisations de la ressource R2 (amélioration)

	R3		
	Tâche 1	Tâche 3	Nombre de produits finis
S7	243	559	1417
S9	410	468	1422
Apprentissage	28	698	1468

Tab. 58 Nombre d'utilisations de la ressource R3 (amélioration)

Dans la partie suivante, nous nous intéressons à l'observation de l'impact des différents paramètres relatifs au modèle d'apprentissage des ressources. Outre les paramètres examinés dans le cas des produits, nous observons aussi l'influence de la taille des cycles de décisions sur le rendement des mécanismes d'apprentissage des ressources.

1.3. Étude de l'influence des paramètres liés au processus décisionnel des ressources

1.3.1. Importance des stratégies

Afin d'examiner l'importance de chaque stratégie utilisée par l'ensemble des ressources, nous avons observé les performances réalisées en appliquant chaque stratégie séparément. Concernant les produits, leurs mécanismes d'apprentissage sont désactivés et c'est la stratégie 7 (aller sur la ressource la moins chargée) qui est adoptée par tous les produits. Dans ce cas, trois catégories de simulations ont été effectuées :

- premièrement, application des onze stratégies une par une,
- deuxièmement, activation des mécanismes d'apprentissage des ressources (AR),
- troisièmement, adoption des stratégies aléatoirement (Ran, ce qui est différent de l'adoption de la stratégie 11), voir tableau Tab. 59,

Les résultats obtenus pour ces deux modes mettent en avant trois constats principaux :

- les stratégies 1 et 9 permettent d'obtenir les meilleures performances. Pour ces deux stratégies, il s'agit de traiter en priorité les produits qui arrivent en premier dans la file d'attente (voir la définition de ces stratégies, tableau Tab. 43). Intuitivement, ces deux stratégies permettent une

fluidité du flux des produits qui passent sur les ressources ce qui explique leurs bonnes performances,

- les stratégies 2, 8 et 10 apparaissent comme celles qui donnent les mauvaises performances du système (voir Figure 99). Cet inconvénient provient du fait que ces stratégies favorisent les produits qui entrent en dernier dans le système,
- les mécanismes d'apprentissage des ressources donnent des performances satisfaisantes mais inférieures à celle obtenues grâce aux stratégies 1 et 9. La question qui se pose alors : s'agit-il d'un problème identique à celui rencontré avec les produits (à savoir un problème lié à l'ajustement des paramètres) ou tout simplement d'un problème de non efficacité des mécanismes d'apprentissage. Les parties suivantes amèneront des éléments de réponse à cette question.

mode décisionnel	SP	AP
Stratégie 1	1387	1283
Stratégie 2	874	830
Stratégie 3	998	956
Stratégie 4	1326	1225
Stratégie 5	1326	1108
Stratégie 6	998	960
Stratégie 7	1026	982
Stratégie 8	949	935
Stratégie 9	1387	1283
Stratégie 10	915	915
Stratégie 11	1185	1107
AR	1202	1109
Ran	1089	1009

Tab. 59 Impact des différentes stratégies adoptées par les ressources

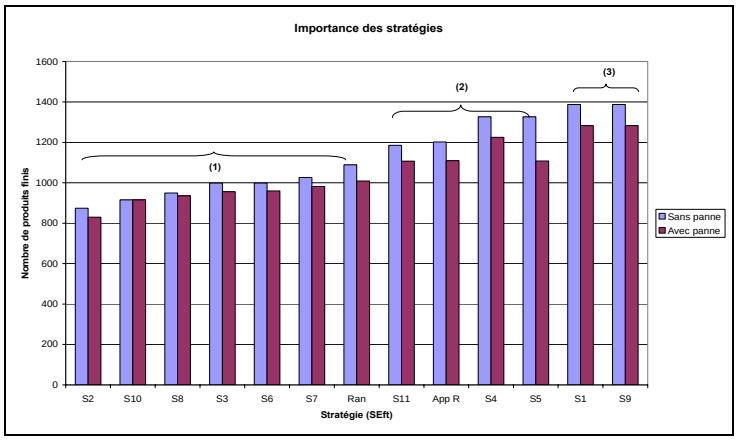


Figure 99- Variation des performances selon la stratégie adoptée

Sixième conclusion (C6) : pour l'ensemble des ressource, les performances globales obtenues varient en fonction de la stratégie adoptée : cette variation permet de qualifier chaque stratégie (bonne, mauvaise, …). La question qui se pose alors : dans une optique d'amélioration continue des performances, est-ce qu'il faut éliminer les mauvaises stratégies ?

1.3.2. Influence du facteur d'oubli

En faisant varier le facteur d'oubli, nous avons pu observer clairement l'effet de ce paramètre sur les performances obtenues. Les résultats obtenus confirment la conclusion C2 relative aux produits. En effet, nous constatons que :

- généralement, les performances obtenues décroissent avec l'augmentation de la valeur du facteur d'oubli,
- la meilleure performance est obtenue pour λ_f=7. Pour la suite des simulations, nous fixons ce paramètre à 7.
- l'écart entre les performances obtenues en mode normal et perturbé (voir Figure 100) est réduit comparé à celui observé dans le cas des produits (voir Figure 83).

Septième conclusion (C7) : la performance globale obtenue, dans le cas de l'application des mécanismes d'apprentissage des ressources, varie en fonction de la valeur prise par le facteur d'oubli : plus la valeur de ce paramètre est réduite, plus ces mécanismes sont efficace.

λ_f	SP	AP	λ_f	SP	AP
3	1294	1182	15	1234	1192
4	1280	1209	16	1189	1118
5	1261	1173	17	1256	1126
6	1175	1127	18	1250	1144
7	1367	1225	19	1188	1108
8	1299	1185	20	1202	1109
9	1239	1149	21	1232	1147
10	1216	1142	22	1227	1142
11	1201	1134	23	1216	1123
12	1216	1154	24	1263	1153
13	1226	1112	25	1228	1132
14	1224	1129	30	1179	1101

Tab. 60 Influence du facteur d'oubli sur la performance globale du système

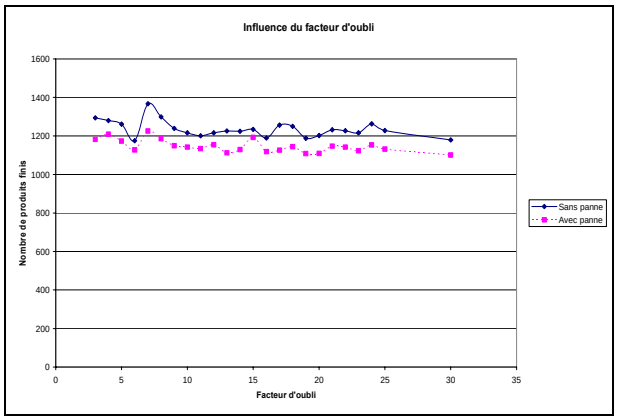


Figure 100- Évolution de la performance du système en fonction du facteur d'oubli

1.3.3. Influence du taux de mutation

En appliquant les mécanismes d'apprentissage, nous avons fait varier le taux de mutation (σ_f) entre 0 et 1, voir tableau Tab. 61. Les résultats obtenus confirment ceux observés pour l'apprentissage des produits. En effet, les meilleures performances sont obtenues pour σ_f=0.1 tant pour le mode de fonctionnement normal que pour le mode perturbé. L'évolution des performances globales du SPBS en fonction de ce paramètre est assez régulière (voir Figure 101) :

σ_f	SP	AP
0	1053	997
0.1	1367	1225
0.2	1334	1201
0.3	1321	1145
0.4	1332	1106
0.5	1333	1167
0.6	1225	1133
0.7	1338	1133
0.8	1283	1218
0.9	1340	1159
1	1323	1162

Tab. 61 Variation des performances obtenues en fonction deσ_f

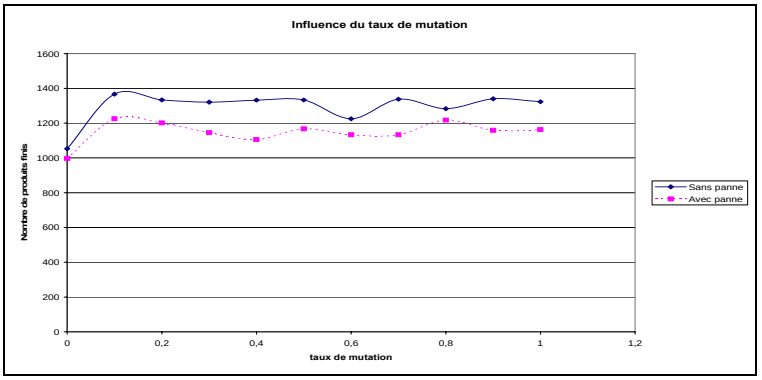


Figure 101- Évolution des performances globales en fonction de σ_f

Huitième conclusion (C8) : L'application de l'opérateur de mutation est nécessaire dans une logique de diversification de données. Les simulations effectuées montrent que l'absence de cet opérateur réduit les performances globales d'un SPBS.

1.3.4. Influence du seuil de performance

Rappelons que pendant le processus décisionnel des ressources, les cycles de décisions prises par chaque ressource sont évalués continuellement. Cette évaluation se base sur la définition d'un seuil de performance (s_f). Dans les tests de simulation que nous avons faits, ce seuil prend des valeurs comprises entre 0.1 et 12. Les performances obtenues appréciées en terme de nombre de produits finis sont mentionnées dans le tableau Tab. 62. L'analyse de ce tableau fait ressortir les deux constats suivants :

- pour des valeurs réduites du seuil de performance ($s_f \leq 4$), les performances sont assez médiocres et ne varient pratiquement pas,

- les meilleures performances sont réalisées pour $s_f \geq 5$ en fonctionnement normal et perturbé, voir Figure 102 :

Le premier point peut s'expliquer par le fait que les performances obtenues peuvent varier d'un cycle de décisions à un autre. Donc, définir un intervalle réduit ([PMEft(k,NbD,λ_f)-s_f , PMEft(k,NbD,λ_f)+s_f]) peut ne pas récompenser toutes les bonnes décisions (ou au contraire peut ne pas pénaliser toutes les mauvaises décisions).

s_f	SP	AP	s_f	SP	AP
0.1	1069	941	3	972	919
0.2	1109	1021	4	976	911
0.3	1027	981	5	1367	1225
0.4	1027	981	6	1367	1093
0.5	1032	984	7	1360	1158
0.6	1039	978	8	1365	1159
0.7	1039	978	9	1354	1105
0.8	1031	983	10	1367	1190
0.9	1002	975	11	1310	1225
1	1086	982	12	1367	1223
2	952	903			

Tab. 62 Variation du nombre de produits finis en fonction du seuil de performance

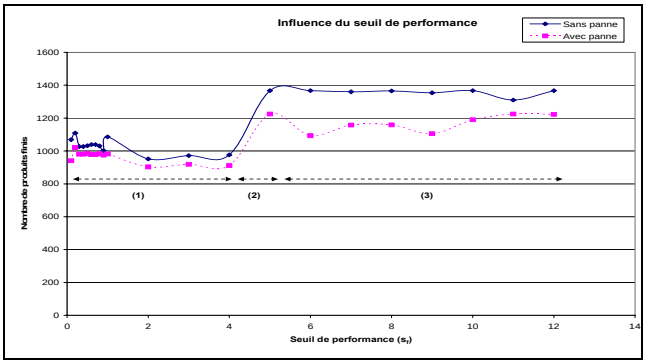


Figure 102- Influence du seuil de performance sur les performances globales

Neuvième conclusion (C9) : Afin de pouvoir évaluer correctement les bonnes ou les mauvaises décisions, le seuil de performance doit être fixé de manière à définir un intervalle ([PMEft(k,NbD,λ_f)-s_f PMEft(k,NbD,λ_f)+s_f]) : ceci peut être réalisé en effectuant des tests préliminaires (simulation).

1.3.5. Influence du signal de renforcement

Nous avons étudié par ailleurs l'influence du signal de renforcement (r_f) sur les performances globales obtenues. Les résultats numériques sont mentionnés dans le tableau Tab. 63. Ces résultats obtenus vont dans le même sens que la conclusion C5 à savoir qu'il faut pénaliser ou récompenser les stratégies avec un signal identique, voir Figure 103 :

r_f	SP	AP
1	1367	1225
2	1367	1225
3	1367	1225
4	1367	1225
5	1367	1225
6	1367	1225
7	1367	1225
8	1367	1225
9	1367	1225
10	1367	1225
(+3,0,-5)	1367	1225
(+5,0,-3)	1367	1225
Aléatoire	1301	1129

Tab. 63 Variation du nombre de produits finis en fonction du signal de renforcement

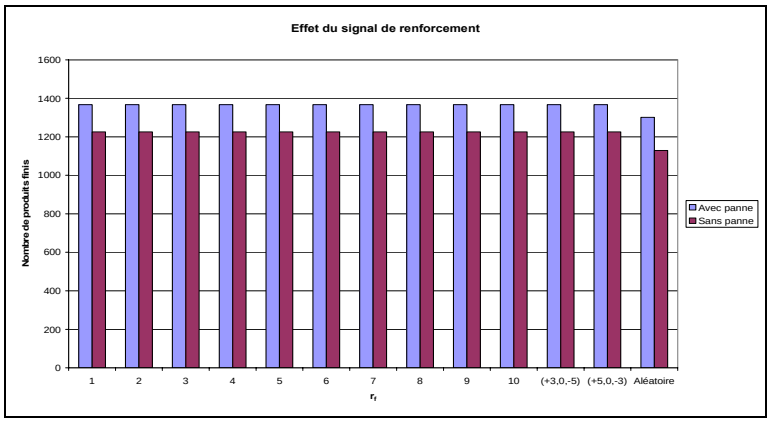


Figure 103- Influence du signal de renforcement sur les performances globales

1.3.6. Influence de la taille des cycles de décisions

Outre l'étude de l'influence des différents paramètres donnée précédemment, nous avons jugé essentiel, dans le cas du processus décisionnel des ressources, d'observer l'effet de la taille des cycles de décisions (NbD) sur l'efficacité des mécanismes d'apprentissage. Logiquement, le choix du nombre de décisions par cycle doit tenir compte des deux points suivants :

- un cycle composé d'un nombre réduit de décisions aura pour conséquence de limiter l'effet des mécanismes génétiques (mutation et croisement),
- un cycle composé d'un grand nombre de décisions entraîne trois inconvénients : prise en compte de données très anciennes, un renouvellement lent des ensembles des cycles qui servent aux mécanismes génétiques et le ralentissement de la simulation (trop de données utilisées).

Il s'agit alors d'utiliser une taille qui satisfait un compromis entre les deux points cités précédemment. Dans les simulations que nous avons réalisées, la taille des cycles est identique pour toutes les ressources (cette hypothèse facilite la mise en œuvre informatique). Dans ce sens, nous avons varié cette taille entre 3 et 10 et les performances obtenues en mode normal et perturbé sont données dans le tableau Tab. 64.

Nous constatons que la taille du cycle des décisions prises a une influence nette sur le rendement des mécanismes décisionnels que nous avons mis en œuvre. En effet, un nombre très limité ou assez élevé de décisions par cycle a tendance à dégrader les performances globales du système, voir Figure 104.

NbD	SP	AP
3	1344	1073
4	1253	1147
5	1367	1225
6	1236	1191
7	1246	1123
8	1062	1002
9	1073	1004
10	1092	1026

Tab. 64 Variation du nombre de produits finis en fonction de la taille du cycle des décisions

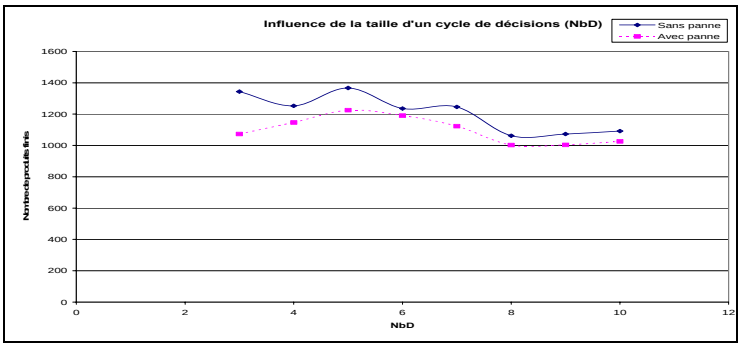


Figure 104- Influence de la taille du cycle de décisions sur les performances globales

En outre, et pour observer plus clairement l'effet du nombre de décisions par cycle, nous avons relevé, pour chaque ressource, le nombre d'utilisations de chaque stratégie (voir tableau Tab. 65). Cette observation est réalisée en faisant varier NbD entre 3 et 10. Les données obtenues confirment deux constats essentiels : pour un nombre de décisions par cycle inférieur à 7, la première stratégie (FIFO qui appliquée séparément, donne les meilleures performances) est principalement utilisée. Ceci s'explique par le fait que les mécanismes d'apprentissage détectent rapidement l'efficacité de cette stratégie. Le deuxième constat montre que lorsque le nombre de décisions par cycle augmente, nous remarquons que les autres stratégies sont plus utilisées. Nous pensons que cette utilisation plus concrète provient des opérateurs génétiques (croisement, mutation) qui permettent une diversification accrue des stratégies.

Stratégie										
S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11

Ressource R1	2366	11	4	4	8	3	2	6	2	2	3	3	taille cycle
	2432	5	5	3	2	4	2	4	2	3	3	4	
	2414	10	5	4	6	4	3	3	3	5	4	5	
	2303	16	5	4	3	1	5	7	7	5	7	6	
	2322	10	11	8	4	4	7	6	6	7	6	7	
	230	7	7	229	5	451	228	6	4	451	225	8	
	397	393	6	204	206	9	7	7	395	199	6	9	
	183	5	361	360	186	360	6	6	6	363	15	10	
	Stratégie												
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11		
Ressource R2	5	5	1	3	4	2	190	4	193	195	193	3	taille cycle
	982	3	6	4	2	4	1	5	4	2	3	4	
	586	4	396	7	9	1	5	3	3	1	6	5	
	225	223	7	115	4	3	5	7	5	110	3	6	
	215	109	319	6	12	5	4	113	3	3	6	7	
	91	86	1	98	98	93	9	97	6	90	94	8	
	84	88	5	81	87	86	4	169	91	91	8	9	
	78	80	73	14	155	5	5	12	152	80	157	10	
	Stratégie												
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11		
Ressource R3	142	141	4	3	3	140	142	6	4	4	2	3	taille cycle
	190	191	2	2	2	1	191	4	4	3	3	4	
	125	3	3	5	128	4	1	128	126	5	128	5	
	317	5	4	3	213	7	5	6	6	3	108	6	
	78	81	158	8	83	8	82	2	5	84	3	7	
	163	102	122	10	29	26	28	5	5	55	74	8	
	189	70	61	10	135	3	70	5	9	67	4	9	
	170	2	67	68	69	9	62	63	68	65	8	10	

Tab. 65 Répartition de l'utilisation des stratégies par les 3 ressources

L'application des cycles comportant un nombre de décisions différent montre que le processus décisionnel des ressources permet de détecter les bonnes stratégies. Dans notre modèle, c'est la stratégie S1 (une très bonne stratégie) qui est la plus utilisée, voir Figure 105.

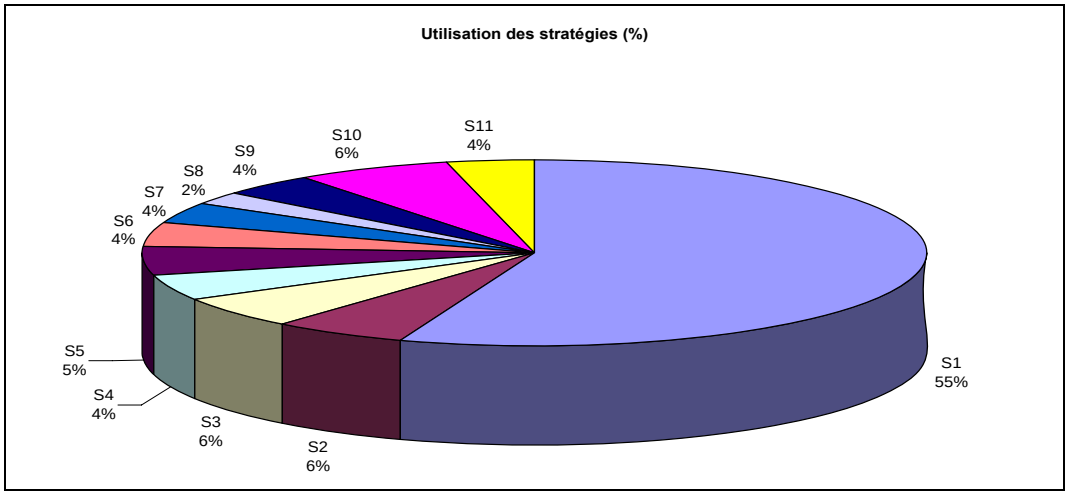


Figure 105- Répartition (en pourcentage) de l'utilisation des stratégies par les ressources

Dixième conclusion (C10) : nous constatons que les résultats obtenus grâce à l'apprentissage des ressources sont satisfaisants et ceci grâce à l'ajustement des différents paramètres du modèle. Au travers des simulations, il s'est avéré que la stratégie FIFO (la plus simple) est la plus utilisée et elle donne les meilleures performances.

1.3.7. Influence du nombre de produits présents simultannément dans le SPBS

Nous avons aussi observé l'influence du nombre de produits (NPS) qui sont présents simultanément dans le SPBS (jusqu'à ici il est fixé à 10). Ce paramètre est une donnée globale liée au fonctionnement du SPBS. Cette observation est réalisée en comparant les résultats obtenus par apprentissage et en appliquant la stratégie 1 pour chaque ressource et la stratégie 7 pour les produits. Par ailleurs, cette comparaison est effectuée en mode de fonctionnement normal (voir Tab. 66 et Figure 106) et perturbé (voir Tab. 67 et Figure 107).

NPS	S1-S7	Apprentissage
5	980	990
6	1021	1286
7	1238	1310
8	1297	1318
9	1248	1312
10	1387	1394
11	1362	1225
12	1395	1252
13	1370	1212
14	1361	1282
15	1362	1267
20	1459	1160
25	1447	1197
30	1417	1113
35	1446	1173
40	1443	1232

Tab. 66 Impact du nombre de produits présents simultanément dans un SPBS (sans panne)

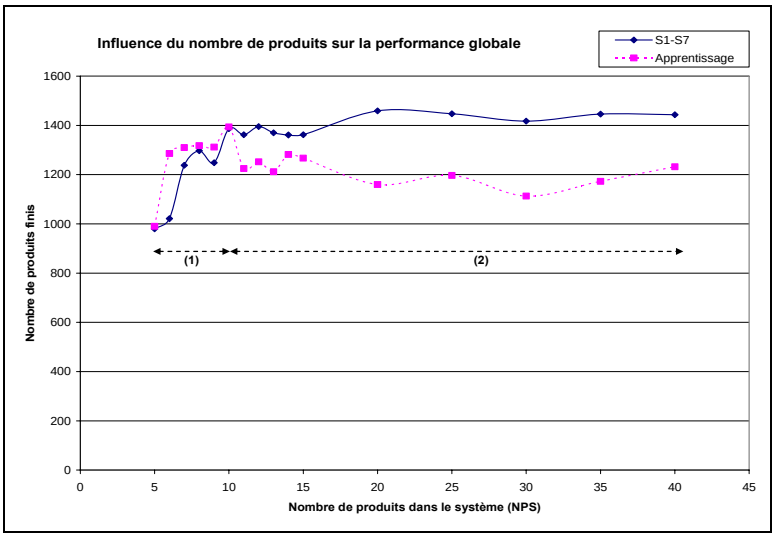


Figure 106- Variation du nombre de produits finis en fonction du nombre de produits présents dans le système (sans panne)

NPS	S1-S7	Apprentissage
5	788	958
6	977	1199
7	1169	1238
8	1216	1231
9	1174	1233
10	1283	1287
11	1261	1014
12	1278	1007
13	1270	1077
14	1249	1023
15	1249	974
20	1300	919
25	1263	777
30	1225	729
35	1215	816
40	1194	898

Tab. 67 Impact du nombre de produits présents simultanément dans un SPBS (avec panne)

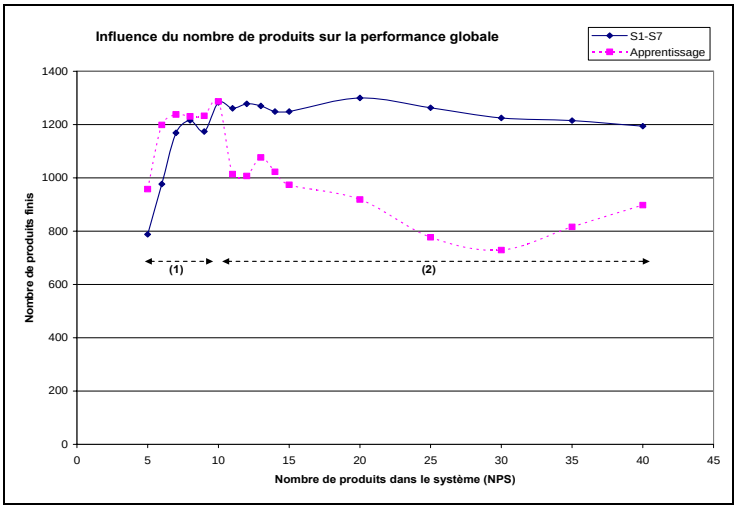


Figure 107- Variation du nombre de produits finis en fonction du nombre de produits présents dans le système (avec panne)

Les résultats obtenus montrent que l'augmentation du nombre de produits présents simultanément dans un SPBS réduit l'effet des mécanismes d'apprentissage. En effet, au-delà de NPS>10, les résultats obtenus par les mécanismes d'apprentissage sont médiocres pour les deux modes d'apprentissage : c'est un inconvénient de notre approche. Un cause possible de cette dégradation des performances réside dans la variation 'intensive' des stratégies adoptées : cette

variation peut avoir comme conséquence l'existence de plusieurs produits créés mais pas encore finis à la fin de la simulation. Néanmoins, celle-ci peut être intéressantes dans le cas d'un SPBS dans lequel les files d'attente ont une taille finie car pour NPS<10, les mécanismes que nous avons développés donnent les meilleurs résultats.

1.4. Comparaison entre les modes d'apprentissage

Nous avons, à ce stade, identifié les 'bonnes stratégies' pour les entités actives. Il apparaît judicieux de faire une étude comparative en combinant ces bonnes stratégies avec les mécanismes d'apprentissage des produits et des ressources. Dans ce cadre, nous avons envisagé 9 modes (combinaisons) possibles, voir Tab. 68 :

Processus décisionnel d'un produit	Processus décisionnel d'une ressource	nombre de produits finis (sans panne)	nombre de produits finis (avec panne)
SEmp(7)	SEft(1)	1387	1283
	SEft(9)	1387	1283
	Apprentissage ressource	1367	1225
SEmp(9)	SEft(1)	1373	1289
	SEft(9)	1330	1252
	Apprentissage ressource	1353	1205
Apprentissage produit	SEft(1)	1394	1293
	SEft(9)	1303	1127
	Apprentissage ressource	941	796

Tab. 68 Tableau comparatif des différents modes décisionnels

L'analyse du tableau ci-dessus, permet de d'établir les conclusions suivantes :

- un apprentissage hybride ne convient pas : ceci est dû à notre avis au fait qu'une autonomie totale est conflictuelle : si chaque type entité active optimise son fonctionnement, les effets de chaque type d'apprentissage peuvent s'annuler mutuellement,
- les mécanismes d'apprentissage proposés (pour les entités fixes de traitements et les entités mobiles produits) donnent d'excellents résultats (notamment pour les entités mobiles produits) à condition de les combiner avec les bonnes stratégies et d'ajuster les différents paramètres qui définissent le modèle d'apprentissage.

Cette étude relative à l'impact des différents paramètres détermine leur importance dans les mécanismes décisionnels proposés, voir Tab. 69 :

	paramètres	important	peu ou pas important
paramètres liés aux ressources	facteur d'oubli (λ_f)	*	
	seuil de performance (s_f)	*	
	signal de renforcement (r_f)		*
	taux de mutation (σ_f)		*
	taille des cycles de décisions (NbD)	*	
paramètres liés aux produits	facteur d'oubli (λ_m)	*	
	seuil de performance (s_m)	*	
	signal de renforcement (r_m)		*
	taux de mutation (σ_m)		*
paramètres liés au SPBS	nombre de produit dans le SPBS (NPS)	*	

Tab. 69 Tableau récapitulatif des différents paramètres et de leur importance

2. Simulation d'un exemple étendu

L'exemple, que nous avons traité est simple (3 ressources, 3 tâches), mais représente l'avantage d'être complet. Néanmoins, il est indispensable de traiter un exemple plus étendu (en augmentant le nombre de ressources et/ou de tâches) pour observer les performances obtenues. Pour ce faire, nous avons considéré un exemple de 6 ressources, dans lequel chaque produit est associé à une gamme composée de 6 tâches. Dans cet exemple, chaque ressource est capable d'exécuter trois tâches. Les temps opératoires et de transfert sont mentionnés respectivement dans les tableaux Tab. 70 et Tab. 71 :

	T1	T2	T3	T4	T5	T6
R1	1	4	3	-	-	-
R2	-	5	2	6	-	-
R3	-	-	1	5	2	-
R4	-	-	-	4	3	6
R5	2	6	-	-	-	4
R6	3	-	-	-	1	5

Tab. 70 Temps de opératoires (6 ressources, 6 tâches)

→	R1	R2	R3	R4	R5	R6	S
E	1	4	4	4	2	3	-
R1	1	2	4	5	3	2	1
R2	2	1	3	4	2	5	1
R3	4	3	1	2	5	4	1
R4	5	4	2	1	6	3	1
R5	3	2	5	6	1	6	1
R6	2	5	4	3	6	1	1

Tab. 71 Temps de transfert (6 ressources)

2.1. Etude comparative

Les premières simulations ont été effectuées en tenant compte des données mentionnées dans le tableau Tab. 72 :

entité active	produit	ressource
facteur d'oubli	λ_m=5	λ_f=5 (taille d'un cycle NbD=5)
taux de mutation	σ_m=0.1	σ_f=0.1
seuil de performance	s_m=4	s_f=5
signal de renforcement	$(+r_m,0,-r_m)=(+1,0,-1)$	$(+r_f,0,-r_f)=(+1,0,-1)$
nombre de produits (NPS)	30	

Tab. 72 Données utilisées pour les tests de simulation relatif à l'exemple étendu

Comme dans le cas du premier exemple étudié, nous comparons entre les résultats donnés par les 'bonnes stratégies' et ceux donnés par les mécanismes décisionnels des entités actives.

2.1.1. Comparaison des résultats obtenus

La meilleure performance est obtenue pour une application, conjointe, des stratégies 9 relatives aux produits et aux ressources, voir Tab. 73.

Processus décisionnel d'un produit	Processus décisionnel d'une ressource	nombre de produits finis (sans panne)
SEmp(7)	SEft(1)	833
	SEft(9)	832
	Apprentissage ressource	783
SEmp(9)	SEft(1)	851
	SEft(9)	852
	Apprentissage ressource	791
Apprentissage produit	SEft(1)	733
	SEft(9)	739
	Apprentissage ressource	632

Tab. 73 Performances obtenues par les différents modes décisionnels

2.1.2. Ajustement des différents paramètres

Les résultats mentionnés ci-dessus confirment ceux obtenus pour le premier exemple et notamment l'inadéquation d'un apprentissage hybride dans une logique d'amélioration continue des performances. Par ailleurs, nous avons étudiés l'impacts des différents paramètres qui définissents nos mécanismes décisionnels, les résultats obtenus sont donnés dans Tab. 74 et Tab. 75 :

paramètres relatifs aux produits	valeur	nombre de produits finis (SP)	
facteur d'oubli (λ_m)	4	751	
	5	762	↓
	6	757	
	7	723	
	8	712	
seuil de performance (s_m)	5	762	
	6	781	
	7	794	↓
	8	790	
	9	779	
	10	753	
taux de mutation (σ_m)	0	632	
	0.1	867	↓
	0.2	824	
	0.9	795	
	1	780	
signal de renforcement (r_m)	1	867	↓
	2	867	
	3	867	
	4	867	

Tab. 74 Influence des différents paramètres liés aux produits sur les performances obtenues

paramètres relatifs aux ressources	valeur du paramètre	nombre de produits finis (SP)	
facteur d'oubli (λ_f)	4	740	
	5	762	
	6	780	↓
	7	755	
	8	749	
seuil de performance (s_f)	6	743	
	7	787	
	8	803	↓
	9	796	
	10	756	
taux de mutation (σ_f)	0	695	
	0.1	855	↓
	0.2	841	
	0.9	766	
	1	740	
signal de renforcement (r_f)	1	855	↓
	2	855	
	3	855	
	4	855	
nombre de décisions par cycle (NbD)	4	822	
	5	855	↓
	6	843	
	7	812	
	8	796	

Tab. 75 Influence des différents paramètres liés aux ressources sur les performances obtenues

Ces résultats montrent la nécessité d'un ajustement des valeurs prises par les différents

paramètres pour une meilleure efficacité des différents algorithmes proposés dans notre travail.

3. Positionnement de notre approche de simulation

Nous distinguons quatre approches qui peuvent servir à l'évaluation des performances d'une approche de pilotage hétérarchique :

- simuler le système physique et le système de pilotage : il s'agit de simuler des scénarii et d'observer le comportement de ces deux systèmes, c'est l'approche adoptée dans notre travail (voir Figure 108-(a)),
- concevoir un système de pilotage et le coupler directement avec le système réel (voir Figure 108-(b)). Des applications à l'échelle industrielle existent dans ce cadre, voir [Lažanský et al., 01],
- une approche qui repose sur une distinction claire entre l'émulation du système physique, et la représentation du système décisionnel, [Klein et Thomas, 06] (voir Figure 108-(c)),
- la dernière approche consiste à associer la partie physique et la partie information (par exemple [McFarlane et al., 03] cité dans [Morel et Grabot, 03][80]), voir Figure 108-(d). Nous pensons que notre approche peut être appliquée dans ce cadre grâce aux nouvelles technologies (électronique, communication, capteurs, …).

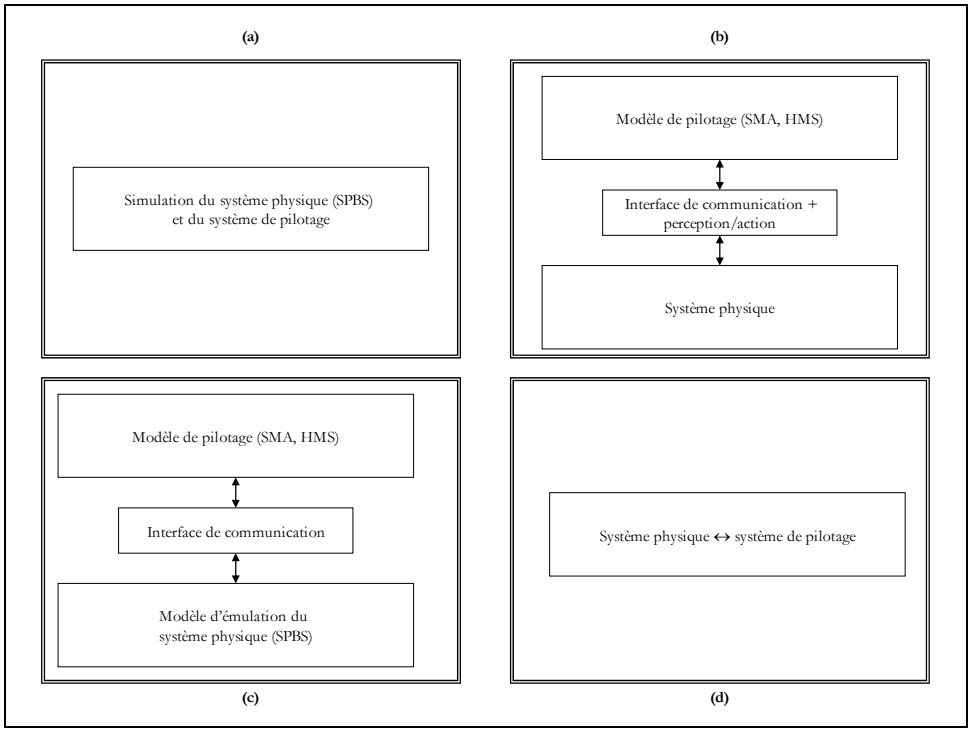


Figure 108- Simulation, émulation et pilotage des SPBS

4. Conclusion

Dans ce chapitre, nous avons présenté un ensemble de simulations qui ont confirmé l'intérêt

[80] Ces implémentations restent d'une complexité relativement faible, mais permettent néanmoins de donner une bonne appréciation des possibilités offertes en terme de flexibilité et de stabilité des systèmes holoniques face à des perturbations [Gouyon, 04].

de notre approche :

- nous avons observé l'importance des différents paramètres liés aux modèles décisionnels des entités actives via un ensemble de tests préliminaires. Cette étape est nécessaire pour le succès des mécanismes décisionnels que nous avons proposés,
- nous avons observé le comportement du système en mode de fonctionnement normal et perturbé : les résultats sont satisfaisants par rapport à ceux obtenus en appliquant des règles de décisions statiques,
- l'apprentissage des entités mobiles produits donne les meilleurs résultats que l'apprentissage des entités fixes de traitements. La combinaison de ces deux modes d'apprentissage n'est pas appropriée dans une optique d'amélioration continue des performances d'un SPBS.

CONCLUSION GENERALE ET PERSPECTIVES

1. Conclusion générale

1.1. Bilan

La problématique de l'amélioration continue des performances des Systèmes de Production de Biens et de Services a été abordée dans ce mémoire à la fois selon une approche conceptuelle (modèles) et une approche expérimentale (simulations).

Nous avons commencé, dans le premier chapitre, par placer notre travail dans un cadre général d'agilité. Ce positionnement était le précurseur d'une double caractérisation de l'agilité : une première caractérisation systémique définie par le triplet (agilité structurelle, agilité opérationnelle, agilité évolutionniste) et une deuxième caractérisation typologique qui classifie l'agilité en trois catégories selon la nature du SPBS étudié (classe I, classe II, classe III).

Par la suite, notre intérêt a été porté sur une agilité de type <évolutionniste, classe III>. Cette orientation nous a amené à réserver le second volet du premier chapitre à un état de l'art consacré à la présentation des principales contributions qui traitent partiellement ou totalement cette problématique. Cet état de l'art a mis l'accent sur l'importance des notions tels que l'autonomie ou encore l'apprentissage dans une optique d'amélioration continue des performances.

Grâce aux conclusions issues du premier chapitre, nous avons pu définir un ensemble de spécifications relatives à un système de pilotage hétérarchique capable d'assurer l'amélioration continue des performances. Une première tâche a consisté à proposer un modèle systémique pour identifier un SPBS et qui est caractérisé par quatre types d'entités (de quoi un SPBS est-il composé ?).

Par ailleurs, la distinction entre des entités actives et des entités passives ainsi que le processus décisionnel dynamique des entités actives basé sur l'adoption des stratégies font partie des spécifications fondamentales présentées dans le chapitre deux. En outre, nous avons formalisé les mécanismes d'apprentissage pour les entités fixes de traitements et pour les entités mobiles produits. Ce formalisme associe le quadruplet <<source,état>,objet,moment> à chaque processus d'apprentissage d'une entité active.

Notre contribution a été présentée dans le chapitre trois. Son originalité provient d'une adéquation judicieuse de trois techniques issues de trois domaines différents : le pilotage hétérarchique (décentralisation locale des capacités décisionnelles), les approches génétiques (sélection naturelle et diversification des données) et l'apprentissage par renforcement (évaluation a posteriori et continue des stratégies adoptées). Notre modèle de pilotage hétérarchique évolutif combine ces trois techniques pour assurer l'amélioration continue des performances des SPBS.

La mise en œuvre de nos propositions a fait l'objet du chapitre quatre. Une modélisation grâce au formalisme UML était nécessaire pour décrire les différentes interactions qui existent dans les modèles proposés. Essentiellement, nous avons présenté les principaux diagrammes (classes, activités et séquences) qui permettent de décrire le fonctionnement du simulateur que nous avons développé.

Le dernier chapitre été consacré à la validation de nos propositions. Dans une première phase, nous avons réalisé une étude complète sur l'influence des différents paramètres relatifs à nos modèles. Les résultats obtenus ont montré la nécessité d'un ajustement préalable de ces paramètres pour la réussite des mécanismes décisionnels. Dans la deuxième phase du chapitre cinq, nous avons travaillé sur des exemples de SPBS plus complets afin d'estimer l'adaptation de nos

propositions à des scénarii divers et variés. Les résultats obtenus ont montré la pertinence de notre contribution.

1.2. Apports et limites

Plusieurs propositions présentées dans ce travail méritent d'être mises en avant :

Tout d'abord, le double référentiel (systémique et typologique) relatif à l'agilité que nous avons proposé constitue un cadre générique de définition du paradigme agilité des SPBS. Dans le même sens, nous avons établi un positionnement entre agilité, performance et amélioration continue des performances.

Ensuite, la modélisation systémique à base d'entité d'un SPBS présentée dans le chapitre deux est doublement générique :

- elle offre la possibilité de modéliser plusieurs domaines divers et variés (voir le tableau Tab. 76 dans lequel sont illustré plusieurs exemples) :

domaine		entité…					Amélioration continue des performances…
		fixe de traitement (eft)	fixe de stockage (efs)	mobile produit (emp)	mobile de transfert (emt)		
	Productique (ateliers)	machines	stocks intermédiaires	pièces	chariot filoguidé	gérer les pannes, éviter l'arrêt complet	
	Informatique (réseaux)	routeur, serveur	carte mémoire	trame, fichiers,…	onde hertzienne, radio,…	éviter la saturation du réseau	
	Robotique (industrie automobile)	robot manipulateur	stocks intermédiaires	pièces mécaniques	AGV	éviter les collisions, optimiser les chemins	
	Services (distribution du courrier)	machine d'aiguillage	dépôt, hangar	colis, courrier	chariots, tapis roulants, …	réduire les délais de distribution et une meilleure traçabilité	

Tab. 76 Applicabilité de la modélisation proposée

- elle permet une décomposition hiérarchique d'un SPBS. Autrement dit, une entité vue séparément (par exemple une société filiale d'une firme internationale) peut être décomposée et modélisée à son tour avec le même formalisme, voir Figure 109 :

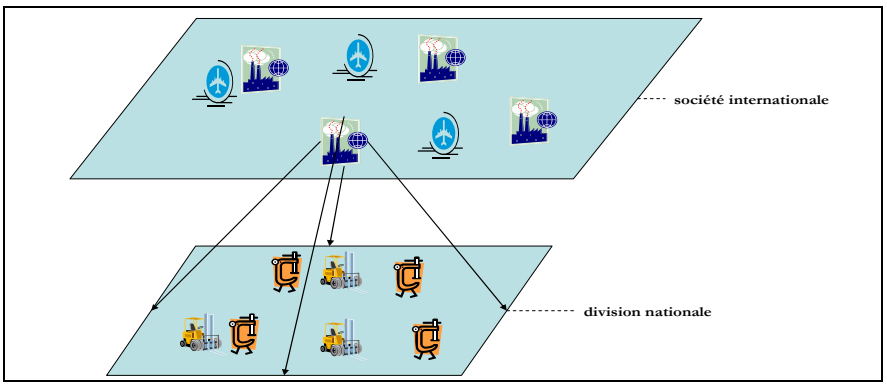


Figure 109- Généricité de la modélisation systémique proposée

Enfin, la formalisation de l'apprentissage (qui est une forme d'intelligence) des entités actives ainsi que la proposition d'un ensemble de mécanismes représentent une originalité en productique. Autant, cette thématique est largement étudiée dans d'autres disciplines (notamment en robotique), autant la notion de l'intelligence en productique est relativement récente. Notre travail apporte une 'pierre à l'édifice des SPBS intelligents'.

Ainsi globalement, les résultats obtenus grâce aux différentes simulations montrent l'intérêt de notre approche dans une optique d'amélioration continue des performances des SPBS :

- en terme de poursuite, une recherche continue des meilleures stratégies qui permettent d'optimiser le temps de passage d'un produit dans le système,
- en terme de régulation, les mécanismes décisionnels et d'apprentissage permettent, via un processus d'évaluation continue des stratégies, d'éviter la ou les ressources en pannes.

Néanmoins, nos propositions montrent deux limites :

Premièrement, la nécessité d'ajuster les paramètres (facteurs d'oubli, taux de mutation, signal de renforcement, …) des différents modèles. Or c'est un inconvénient identifié et commun à d'autres disciplines surtout celles utilisant des résolutions rapprochées. Toutefois, cette étape peut être considérablement réduite. En effet, l'impact de ces paramètres, considérés séparément, est connu dans le cas d'autres disciplines (algorithmes génétiques, apprentissage par renforcement ou l'automatique (facteur d'oubli)). Ces connaissances peuvent être très utiles pour notre approche.

Deuxièmement, la notion du coût est doublement absente dans nos modèles :

- le critère 'coût' n'a pas été pris en compte pour les entités fixes de traitements et pour les entités mobiles produits. Or ce facteur est primordial actuellement pour les SPBS,
- les coûts éventuels d'une mise en pratique de nos propositions n'ont pas été étudiés dans ce travail. Il est évident que notre approche n'est pas adéquate pour les systèmes de type chaînes d'embouteillage par exemple (des milliers de produits ainsi que des flux linéaires). Par contre, nous pensons que notre approche peut être appliquée dans le cas des SPBS qui nécessitent un minimum de pilotage 'intelligent' (industrie automobile par exemple).

2. Perspectives

Notre travail ouvre de nombreuses perspectives tant sur les aspects recherche qu'applicatif :

Sur le plan théorique, les perspectives suivantes peuvent être envisagées :

- l'application des mécanismes génétiques été très intéressante surtout en ce qui concerne la diversification des données. Or dans notre travail, nous nous sommes inspirés de cette technique que nous avons adoptée dans le cadre du pilotage hétérarchique des SPBS. Il sera judicieux d'appliquer les algorithmes génétiques tels qu'ils sont communément utilisés en recherche opérationnelle. En ce sens, sevaux [Sevaux, 04] distingue deux versions possibles d'algorithmes génétiques (population replacement et incremental replacement). Par ailleurs, d'autres techniques (recherche tabou, recuit simulé, …) méritent d'être intégrées dans notre approche,
- la comparaison que nous avons réalisée entre notre approche et une optimisation statique (centralisée) a montré l'intérêt de nos propositions. Néanmoins, c'était une comparaison 'simpliste'. Un développement théorique issu de la recherche opérationnelle sera une piste à

explorer pour pouvoir comparer. Certes, un 'optimiseur centralisé' (par exemple les outils Ilog) donnera des performances optimales mais nous sommes persuadés que ça sera au prix fort d'une explosion combinatoire inévitable. Dans ce sens, l'axe de recherche en ordonnancement 'communication delays' (voir par exemple [Hoogeveen et Woeginger, 01], [Hanen et Munier, 01] ou [Guinand et al., 04]) qui est assez développé constitue une importante source de benchmarking à exploiter,

- dans notre travail, les stratégies sont adoptées séparément dans le processus décisionnel. Une pondération de ces stratégies ou une agrégation floue constitue une perspective de notre travail de recherche. Par ailleurs, un processus décisionnel en adoptant des 'étapes' (donc statique) peut être facilement transposable. Les mêmes principes de sélection et d'évaluation restent valables,
- nous avons observé dans le chapitre cinq l'influence du signal de renforcement. Cette observation peut être complétée par une pondération de ce signal (par exemple 'mieux' récompenser les bonnes stratégies et 'bien' pénaliser les mauvaises),
- une ouverture sur d'autres disciplines sera intéressante. En effet, la notion d'intelligence et d'apprentissage est assez répandue dans d'autres domaines. Cette ouverture peut concerner aussi la mise en œuvre. Dans ce sens, il sera intéressant d'appliquer nos propositions avec les systèmes multi agents ou les systèmes holoniques (nous avons eu l'occasion de coopérer avec une équipe de recherche en SMA, voir [Bousbia et al., 05]). Grâce à ce travail en collaboration, nous avons pu mesurer l'importance d'une coopération inter disciplinaire,
- la multitude des paramètres dont l'influence doit être étudié pose le problème du nombre de tests primaires qui devient non maîtrisable. Cet inconvénient peut être contourné grâce aux plans d'expériences qui représentent un outil mathématique puissant largement utilisé en industrie.

Sur le plan applicatif, nous pouvons étendre nos expérimentations pour inclure les points suivants :

- l'effet des paramètres génétiques de nos modèles méritent d'être étudiés plus finement (par exemple l'influence du point et de la longueur de croisement ou encore le point de mutation),
- la formalisation de l'apprentissage d'une entité active a permis de recenser 42 modes d'apprentissage possible. Or, dans le volet validation de notre travail, nous avons expérimenté un seul mode d'apprentissage pour chaque type d'entité active. La validation d'autres modes peut faire l'objet de travaux ultérieurs,
- dans ce mémoire, nous avons supposé que tous les produits réalisent la même gamme de tâches. Cette hypothèse n'est pas réductrice vis-à-vis de la généricité des mécanismes d'apprentissage proposés et par conséquent la généricité de notre approche. En effet, nous pouvons envisager, un 'apprentissage par famille de produits' dans le cas où nous avons plusieurs gammes à réaliser (produits personnalisés [Gouyon, 04]). Dans ce cas, chaque famille de produits appliquera les mécanismes d'apprentissage que nous avons développés. Des simulations pourraient confirmer la faisabilité et surtout l'efficacité de cette transposition,
- nous pouvons également tester d'autres modes d'entrée des produits : varier la taille d'un lot introduit dans un SPBS, changer la cadence de cette entrée et étudier l'influence de ce paramètre,
- évaluation, au fur et à mesure, des décisions prises : au lieu d'attendre l'exécution des NbT tâches (pour les emp) ou la prises des NbD décisions (pour les eft) pour évaluer les stratégies adoptées, une alternative consiste à évaluer chaque stratégie juste après son adoption. Une

augmenattion de la réactivité du système du pilotage peut être observée,

- nous pensons que les valeurs prises par les temps opératoires et de transfert ont un impact sur les performances obtenues grâce aux mécanismes d'apprentissage proposés dans ce travail : il suffit d'adopter des stratégies qui orientent les produits vers des ressources lointaines ou qui exécutent une tâche en mettant beaucoup de temps pour que la performance globale (nombre de produits finis) chute considérablement. Dans ce cas, nous nous éloignons d'une logique d'optimisation, donc l'amélioration continue des performances n'a plus de sens (voir par exemple les tableaux Tab. 41 et Tab. 42 : pour les temps opératoires, la différence entre le plus petit et plus grand temps opératoire atteint 5 UTs et pour les temps opératoires, elle atteint 9 UTs).
- une application des différents concepts et propositions présentées dans ce mémoire peut être effectuée sur un cas réel (par exemple dans le cadre de l'Atelier Inter-établissements de Production (AIP) de Valenciennes). Ce type de plate-forme offre un cadre adéquat à d'éventuelles applications,
- la dernière perspective concerne les perturbations auxquelles un SPBS est sujet. Deux points essentiels pourraient faire l'objet d'autres expérimentations :
  - dans notre contribution, l'effet de ces perturbations est 'atténué' grâce à leur détection par les mécanismes d'évaluation continus des décisions prises. Une autre alternative consiste à déclencher un processus de vérification chaque fois qu'une emp se trouve sur une file d'attente. Il s'agit dans ce cas de vérifier la disponibilité de l'eft correspondante. En effet, si une eft tombe en panne, le processus décisionnel principal est réactivé et une autre décision sera prise : l'eft en panne sera éliminée (provisoirement) de l'ensemble des eft candidates et l'emp sera orientée vers une autre eft. C'est une piste intéressante à exploiter, néanmoins une question peut se poser : est-il judicieux de relancer un autre processus décisionnel une seconde fois ? En effet, le départ vers une autre eft risque de produire un effet plus néfaste que si l'emp en question est restée sur l'eft en panne. Ce risque résulte de la nécessité d'effectuer une deuxième fois un temps de transfert ainsi qu'un autre temps d'attente sur la nouvelle eft destination, voir Figure 110 :
  - dans notre travail, seules les perturbations 'discontinues' ont été prises en compte (augmentation ou diminution constantes et finies du temps opératoire). Or il est possible dans un SPBS d'avoir des entités fixes de traitements qui tombent en panne progressivement (dérive). Ce cas de figure mérite d'être étudié.

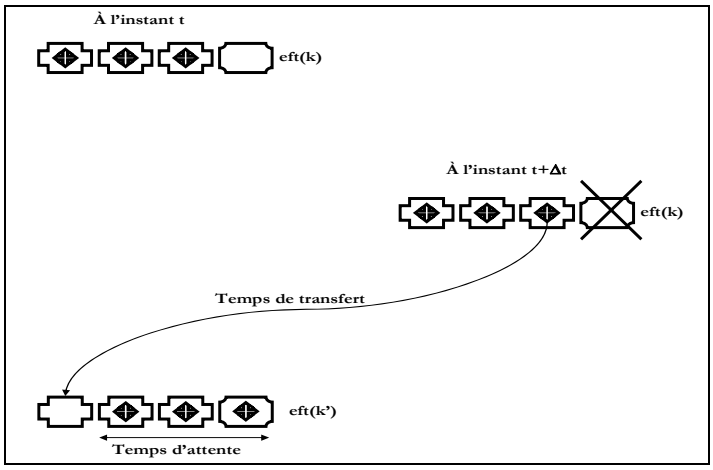


Figure 110- Redirection d'une emp en cas d'une panne survenue sur une eft

REFERENCES BIBLIOGRAPHIQUES

[Huyet, 04] Anne-Lise Huyet 'Extraction de connaissances pertinentes sur le comportement des systèmes de production : une approche conjointe par optimisation évolutionniste via simulation et apprentissage'. Thèse de doctorat, Université Blaise Pascal, Clermont II, 2004.

[Agarwal et al., 05] Ashish Agarwal, Ravi Shankar and M.K. Tiwari 'Modeling the metrics of lean, agile and leagile supply chain: An ANP-based approach' European Journal of Operational Research, February 2005.

[Aydin et Öztemel, 00] M. E. Aydin et E. Öztemel 'Dynamic job-shop scheduling using reinforcement learning agents' in Robotics and Autonomous Systems, volume 33, 2000, pages 169-178.

[Baïna et Morel, 06] Salah Baïna et Gérard Morel 'Product centric holons for synchronisation and interoperability in manufacturing environments'. Proceedings of IFAC INCOM 06 Symposium, April 7th-9th, Saint Etienne, France, 2006.

[Baker, 98] A. D. Baker, A survey of factory control algorithms that can be implemented in a multi-agent heterarchy: dispatching, scheduling, and pull, Journal of Manufacturing Systems, vol. 17, n°4, 1998, pp. 297-320.

[Bescos et al., 95] P.L. Bescos, P. Dobler, C. Mendoza et G. Naulleau 'Contrôle de gestion et management', Éditions Montchrestien, Collection Entreprendre, Guide des techniques et de la décision, Paris, 3ème édition, 1995.

[Beslon, 95] Guillaume Beslon 'Contrôle sensori-moteur par réseaux neuromimétiques modulaires : Approche pour le Pilotage Réactif en Atelier Flexible'. Thèse de doctorat, INSA de Lyon, 1995.

[Bessant et al., 01] John Bessant, Sarah Caffyn and Maeve Gallagher 'An evolutionary model of continuous improvement behaviour'. Technovation, Volume 21, Issue 2, February 2001, Pages 67-77.

[Blaha et Rumbaugh, 05] Michael Blaha et James Rumbaugh 'Modélisation et conception orientées objet avec UML 2'. 2ème édition, éditions Pearson Education.

[Bongaerts et al., 00] Luc Bongaerts, Laszlo Monostori, Duncan McFarlane et Botond Kadar 'Hierarchy in distributed shop floor control' in Computer in Industry N°43 (2000) pages 123-137.

[Booch et al., 00] G. Booch, J. Rumbaugh and I. Jacobson 'Le guide l'utilisateur UML'. Éditions Eyrolles Paris 2000.

[Bousbia et Trentesaux, 06] Salah Bousbia et Damien Trentesaux 'Amélioration continue des performances des

systèmes de production de biens et de services: proposition d'un système de pilotage hétérarchique évolutif par apprentissage'. GDR MACS, Valenciennes, 16 et 17 Novembre 2006.

[Bousbia et al., 05] Salah Bousbia, Abdouroihamane Anli, Damien Trentesaux et Emmanuelle Grislin 'Agile scheduling of flexible manufacturing systems of production'. In 17ème Congrès Mondial IMACS Calcul Scientifique, Mathématiques Appliquées et Simulation, Paris, France 11-15 Juillet 2005.

[Bousbia et Trentesaux, 04a] Salah Bousbia et Damien Trentesaux 'Towards an intelligent heterarchical manufacturing control system for continuous improvement of manufacturing systems performances' in the 5th International Conference On Integrated Design And Manufacturing in Mechanical Engineering, Université De Bath, Angleterre, Avril 2004.

[Bousbia et Trentesaux, 04b] Salah Bousbia et Damien Trentesaux 'Proposition d'un système de pilotage hétérarchique pour l'amélioration continue des performances: approche basée sur l'apprentissage'. GDR MACS, Aix en Provence, 21 et 22 Octobre 2004.

[Bousbia et Trentesaux, 2002] Salah Bousbia et Damien Trentesaux 'Self-organization in distributed manufacturing control: state-of-the-art and future trends' in IEEE International conference on Systems, Man et Cybernetics, El Kamel, Mellouli et Borne (eds), CD-Rom, ISBN: 2-9512309-4-X, Hammamet, Tunisie, Octobre 2002.

[Brezocnik et Balic, 01] Miran Brezocnik and Joze Balic 'A genetic-based approach to simulation of self-organizing assembly' in Robotics and Computer-Integrated Manufacturing, Volume 17, Issues 1-2, February 2001, Pages 113-120.

[Brooks, 91] R. A. Brooks 'The Role of Learning in Autonomous Robots' in Proceedings of the Fourth Annual Workshop on Computational Learning Theory (COLT '91), Santa Cruz, CA, Morgan Kaufmann Publishers, 1991, pages 5-10.

[Brun-Picard et al., 93] Daniel Brun-Picard, A. Ferrarini, M.L. Nervi, L. Couvreur «Proposition d'une architecture décentralisée autonome et ouverte sur le pilotage d'atelier flexible de fabrication'. Actes du 4ème Congrès International de Génie Industriel, Marseille, 15-17 Décembre 1993, Vol. 2 p. 395-403.

[Brun-Picard et Sousa, 99] Daniel Brun-Picard and João S. S. Sousa 'Design of machines and robots endowed with a permanent learning ability'. In Control Engineering Practice, Volume 7, Issue 4, April 1999, Pages 565-571.

[Cagliano et al., 04] Raffaella Cagliano, Federico Caniato and Gianluca Spina 'Lean, Agile and traditional supply: how do they impact manufacturing performance?' In Journal of Purchasing and Supply Management, Volume 10, Issues 4-5, July-September 2004, Pages 151-164.

[Cantamessa, 97] M. Cantamessa, Agent-based modelling and management of manufacturing systems, Computers in Industry, vol. 34, 1997, pp. 173-186.

[Charpentier et al., 01] Patrick Charpentier, Frédéric Chaxel, André Thomas et Emmanuel Muhl 'Quelques approches de pilotage distribué et leur formalisation'. Journal Européen des Systèmes Automatisés, Hermès N°35 2001.pages885-904.

[Chi et Turban, 95] R.T. Chi et E. Turban, Distributed intelligent executive information systems, Decision Support Systems, vol. 14, 1995, pp. 117-130.

[Choi, 95] Ty Choi 'Conceptualizing continuous improvement: Implications for organizational change'. In Omega, Volume 23, Issue 6, December 1995, Pages 607-624.

[David et Alla, 89] Réné David et Hassan Alla 'Du Grafcet aux réseaux de Petri', Hermès, Paris, 1989.

[Deneux, 02] Dominique Deneux 'Méthodes et modèles pour la conception concourante'. Habilitation à diriger des recherches, Université de Valenciennes et du Hainaut-Cambrésis, 24 Janvier 2002.

[Dilts et al., 91] D.M. Dilts, N.P. Boyd et H.H. Whorms 'The evolution of control architectures for automated manufacturing systems'. Journal of Manufacturing Systems, vol. 10, n° 1, 1991, pp. 79-93.

[Dorigo et al., 91] M. Dorigo, V. Maniezzo, et A. Colorni. Positive feedback as a search strategy. Technical Report 91-016, Politecnico di Milano, Italy, 1991.

[Duffie et Prabhu, 96] N.A. Duffie et V. Prabhu 'Heterarchical control of highly distributed manufacturing systems' in International Journal of Computer Integrated Manufacturing ISSN 0951-192X, pages 270-281.Volume 9, Number 4 (1996).

[Dupas, 04] Rémy Dupas 'Amélioration de performance des systèmes de production : apport des algorithmes évolutionnistes aux problèmes d'ordonnancement cycliques et flexibles'. Habilitation à diriger des recherches, L'Université d'Artois, 10 décembre 2004.

[Elkins et al., 04] Debra A. Elkins, Ningjian Huang and Jeffrey M. Alden 'Agile manufacturing systems in the automotive industry' In International Journal of Production Economics, Volume 91, Issue 3, 18 October 2004, Pages 201-214.

[Erschler et al., 97] J. Erschler, M.J. Huguet et G. de Terssac, Décision distribuée en gestion de production : exploitation et régulation de l'autonomie, in Concepts et outils pour les systèmes de production, coord. J.C. Hennet, éd. Cepadues, Toulouse, 1997, pp. 109-131 (ISBN 2-85428-437-2).

[Ferber, 95] Jaques Ferber 'Les systèmes multi-agents, vers une intelligence collective'. InterEditions, Paris, 1995.

[Frayret et al., 01] Jean-Marc Frayret, Sophie D'Amours, Benoit Montreuil and Louis Cloutier 'A network approach to operate agile manufacturing systems' In International Journal of Production Economics, Volume 74, Issues 1-3, December 2001, Pages 239-259.

[Giachettia et al., 03] Ronald E. Giachetti, Luis D. Martinez, Oscar A. Sáenz and Chin-Sheng Chen 'Analysis of the structural measures of flexibility and agility using a measurement theoretical framework'. In Int. J. Production Economics 86 (2003) 47–62.

[Giard, 03] Vincent Giard 'Gestion de la production', éditions Economica, 3ème édition, 2003.

[Glorennec, 03] Pierre-Yves Glorennec 'Apprentissage par renforcement : Application aux systèmes d'inférence floue'. Commande floue 2 : de l'approximation à l'apprentissage (Traité IC2, série systèmes automatisés) sous la direction de Laurent Foulloy, Sylvie Galichet, André Titli. Édition Hermès, 2003.

[Goldberg, 89] D. E. Goldberg. Genetic Algorithms in Search, Optimisation, and Machine Learning. Addison-Wesley Publishing Company, Reading, MA, 1989.

[Gouyon et al., 04] David Gouyon D., Simão J. M., Alkassem K. and Gérard Morel 'Work in progress for product driven manufacturing automation'. Proceedings of IFAC INCOM Symposium, April 7th-9th, Salvador de Bahia, Brazil.

[Gouyon, 04] David Gouyon 'Contrôle par le produit des systèmes d'exécution de la production : Apport des techniques de synthèse'. Thèse doctorat à l'Université Henri Poincaré Nancy, 6 décembre 2004.

[Guinand et al., 04] Frédéric Guinand, Aziz Moukrim and Eric Sanlaville 'Sensitivity analysis of tree scheduling on two machines with communication delays'. In Parallel Computing,

Volume 30, Issue 1, January 2004, Pages 103-120.

[Gunasekaran, 99] A. Gunasekaran 'Agile manufacturing: A framework for research and development' in International Journal of Production Economics, 62 (1999), pages 87-105.

[Hales et Chakravorty, 06] Douglas N. Hales and Satya S. Chakravorty 'Implementation of Deming's style of quality management: An action research study in a plastics company'. International Journal of Production Economics, Volume 103, Issue 1, September 2006, Pages 131-148.

[Hanen et Munier, 01] C. Hanen and A. Munier 'An approximation algorithm for scheduling dependent tasks on m processors with small communication delays'. In Discrete Applied Mathematics, Volume 108, Issue 3, 15 March 2001, Pages 239-257.

[He et al., 01] David He, Astghik Babayan and Andrew Kusiak 'Scheduling manufacturing systems in an agile environment' In Robotics and Computer-Integrated Manufacturing, Volume 17, Issues 1-2, February 2001, Pages 87-97.

[Holland, 75] J.H. Holland 'Adaptation in natural and artificial systems'. Technical report, University of Michigan, Ann Arbor, 1975.

[Hoogeveen et Woeginger, 01] Han Hoogeveen and Gerhard J. Woeginger 'A very difficult scheduling problem with communication delays'. In Operations Research Letters, Volume 29, Issue 5, December 2001, Pages 241-245.

[Intel, 02] 'Electronic Mail : Performance Improving distributed application performance and customer satisfaction' Intel Information Technology White Paper, Enterprise Infrastructure Research, August 2002. Disponible sur www.intel.com.

[Ip et al., 04] W. H. Ip, K. L. Yung and Dingwei Wang 'A branch and bound algorithm for sub-contractor selection in agile manufacturing environment' In International Journal of Production Economics, Volume 87, Issue 2, 28 January 2004, Pages 195-205.

[Iung, 02] Benoît Iung 'Contribution à l'Automatisation des Systèmes Intelligents de Production : Interopérabilité des Processus de Contrôle, Maintenance et Gestion Technique'. Habilitation à diriger des recherches, Université Henri Poincaré, Nancy-I, 2002.

[Jacot, 90] J. H. Jacot 'A propos de l'évaluation économique des systèmes intégrés de production'. Dans Ecosip, Gestion Industrielle et Mesure Economique, Economica, Paris, 1990.

[Klein et Thomas, 06] Thomas Klein et André Thomas 'Développement d'un modèle de simulation pour l'évaluation des systèmes de pilotage distribués'. 6ème Conférence Francophone de MOdélisation et SIMulation, MOSIM'06, du 3 au 5 avril 2006, Rabat, Maroc.

[Koonce et al., 97] David A. Koonce, Cheng-Hung Fang and Shi-Chi Tsai 'A data mining tool for learning from manufacturing systems'. In Computers & Industrial Engineering, Volume 33, Issues 1-2, October 1997, Pages 27-30.

[Koza, 92] J.R. Koza. Genetic Programming. MIT Press, Cambridge, MA, 1992.

[Landau, 93] Ioan Doré Landau 'Identification et commande des systèmes'. Collection Traité des nouvelles technologies série automatique, 2ème Édition. Éditions Hermes 1993.

[Last et Kandel, 04] Mark Last and Abraham Kandel 'Discovering useful and understandable patterns in manufacturing data'. In Robotics and Autonomous Systems, Volume 49, Issues 3-4, 31 December 2004, Pages 137-152.

[Lažanský et al., 01] Jiří Lažanský, Olga Štěpánková, Vladimír Mařík and Michal Pěchouček 'Application of the multi-agent approach in production planning and modelling'.

Engineering Applications of Artificial Intelligence, Volume 14, Issue 3, June 2001, Pages 369-376.

[Le Moigne, 94] Jean Louis Le Moigne 'Théorie du système générale, théorie de la modélisation'. Presses Universitaires de France. Juin 1994.

[Le Quéré, 04] Yann Le Quéré 'Proposition d'un modèle pour l'ordonnancement et la planification réactive : application à la maintenance réactive». Thèse doctorat, Université de Valenciennes et du Hainaut-Cambrésis, 08 septembre 2004.

[Longchamp, 95] R. Longchamp 'Commande Numérique des systèmes dynamiques'. Presses polytechniques et Universitaires Romandes, 1995.

[Maione et Naso, 01] B. Maione et D. Naso 'Evolutionary adaptation of dispatching agents in heterarchical manufacturing systems' in International Journal of Production Research, volume 39, n°7, 2001, pages 1481-1503.

[Malvache et Prieur, 95] P. Malvache et P. Prieur 'Mastering Corporate Experience with the REX Method, Management of Industrial and Corporate Memory'. In Proceedings of the International Symposium on the Management of Industrial and Corporate Knowledge (ISMICK'95). Compiègne.

[Mařík et Lažanský, 00] Vladimír Mařík and Michal Pěchouček: HOLOMAS'00: Industrial Applications of Holonic and Multi-Agent Systems, In Marik V. Pechoucek M., editor(s), Workshop on Industrial Applications of Holonic and Multi-Agent Systems, 11th International Conference on Database and Expert Systems Applications, pages 213, London, UK, September 2000. IEEE Computer Society, ISBN 0-7695-0680-1.

[Mařík et Lažanský, 06] Vladimír Mařík and Jiří Lažanský 'Industrial applications of agent technologies'. Control Engineering Practice, 2006, In Press, Volume xx, Issue x, Pages xxx-xxx.

[Maturana et al., 99] F. Maturana, W.Shen et D.H. Norrie 'MetaMorph : An adaptive agent-based architecture for intelligent manufacturing' in International Journal of Production Research. Volume 37, Number 10 (1999).

[McCulloch, 45] W. S. McCulloch 'A Heterarchy of values determined by the topology of nervous nets'. Bull. math. biophys., vol. 7, 1945, pp. 89-93.

[McFarlane et al., 02] D. C. McFarlane, S. E. Sarma, J. L. Chirn, C. Y. Wong, K. Ashton 'The Intelligent Product in Manufacturing Control'. Journal of EAIA, July 2002.

[McFarlane et al., 03] Duncan McFarlane, Sanjay Sarma, Jin Lung Chirn, C. Y. Wong and Kevin Ashton 'Auto ID systems and intelligent manufacturing control'. Engineering Applications of Artificial Intelligence, Volume 16, Issue 4, June 2003, Pages 365-376.

[McLurkin et Smith, 04] James McLurkin and Jennifer Smith 'Distributed Algorithms for Dispersion in Indoor Environments using a Swarm of Autonomous Mobile Robots' Proceedings of DARS 2004, 7th International Symposium on Distributed Autonomous Robotic Systems, June 23-25, 2004, Toulouse, France.

[McLurkin, 04] James D. McLurkin 'Stupid Robot Tricks: A Behavior-Based Distributed Algorithm Library for Programming Swarms of Robots'. Master of Science in Electrical Engineering and Computer Science at the Massachusetts Institute of Technology May 2004.

[Meinadier, 98] J. P. Meinadier 'Ingénierie et intégration des systèmes', Hermès, Paris, 1998 (ISBN 2-86601-720-X).

[Mesarovic et al., 80] M. D. Mesarovic, D. Macko et Y. Takahara, théorie des systèmes hiérarchiques à niveaux multiples, Economica, Paris, 1980 (ISBN 2-7178-0269-X).

[Michalewicz, 99] Z. Michalewicz. Genetic Algorithms + Data Structures = Evolution Programs. Springer, Berlin, 1999.

[Micouin, 06] Patrice Micouin : 'Eléments de définition et de mise en œuvre de processus d'ingénierie de systèmes : Application à la construction automobile'. Thèse de doctorat, Ecole Nationale Supérieure d'Arts et Métiers, septembre 2006.

[Millot, 99] P. Millot, Systèmes homme-machine et automatique, Journées Doctorales d'Automatique, JDA'99, Nancy, pp. 1-24.

[Mintzberg, 82] H. Mintzberg, Structure et dynamique des organisations, éd. Organisation, Paris, 1982. (ISBN 2-7081-0463-2).

[Miyagi et Riascos, 05] P.E. Miyagi and L.A.M. Riascos 'Modeling and analysis of fault-tolerant systems for machining operations based on Petri nets'. In Control Engineering Practice, March 2005.

[Monostori, 99] László Monostori 'AI and machine learning techniques for managing complexity, changes and uncertainties in manufacturing' Engineering Applications of Artificial Intelligence, Volume 16, Issue 4, June 2003, Pages 277-291.

[Morel et al., 03] Gérard Morel, Hervé Panetto, Marek Zaremba and Frédérique Mayer 'Manufacturing Enterprise Control and Management System Engineering: paradigms and open issues'. Annual Reviews in Control, Volume 27, Issue 2, 2003, Pages 199-209.

[Morel et Grabot, 03] G. Morel and B. Grabot 'Editorial of special issue'. Engineering Applications of Artificial Intelligence Volume 16, Issue 4, Pages 271-275.

[Ohno, 88] Taiichi Ohno 'Toyota Production System: Beyond Large-Scale Production'. Productivity Press, 1988.

[Okino, 93] Okino N. 'Bionic manufacturing system' in Peklenik J, editor, Flexible Manufacturing system: past-present-future. Faculty of Mechanical Engineering. Ljubljana, Slovenia, 1993.73-95.

[Öztürk et al., 05] Atakan Öztürk, Sinan Kayalıgil and Nur E. Özdemirel 'Manufacturing lead time estimation using data mining'. In European Journal of Operational Research, Available online 16 June 2005.

[Paris, 04] Jean-Luc Paris 'Apport des algorithmes évolutionnistes et de l'apprentissage pour l'analyse et l'optimisation via simulation des systèmes de production'. Habilitation à diriger des recherches, Université Blaise Pascal, 9 Novembre 2004.

[Parunak et Brueckner, 01] H. Van Dyke Parunak & Seven Brueckner 'Entropy and Self-Organization in Multi-Agents Systems' in Proceeding of the International Conference on Autonomous Agents pages 124-130. May 28-June 1, 2001, Montreal, Canada.

[Parunak, 85] H. V. D. Parunak 'Manufacturing experienced with contract net'. Proceedings of Distributed Artificial Intelligence Workshop, pp. 67-91, 1985.

[Peeters et al., 01] Patrick Peeters, Hendrik Van Brussel, Paul Valckenaers, Jo Wyns, Luc Bongaerts, Martin Kollingbaum and Tapio Heikkilä 'Pheromone based emergent shop floor control system for flexible flow shops'. In Artificial Intelligence in Engineering, Volume 15, Issue 4, October 2001, Pages 343-352.

[Penalva, 06] Jean Michèl Penalva 'Intelligence collective'. Les Presses des Mines de Paris, 371 pages, ISBN: 2911762711.

[Penalva, 97] Jean Michèl Penalva 'La modélisation par les systèmes en situations complexes'.

Thèse de doctorat, Université de Paris Sud, 1997.

[Pujo et al., 99] P. Pujo, N. Broissin, S. Meyer et J.C. Bertrand 'Pilotage décentralisé des systèmes de production'. Congrès Génie Industriel, Montréal, 1999, pp. 1975-1981.

[Pujo et Ounnar, 01] P. Pujo et F. Ounnar 'Proposition d'un pilotage décentralisé auto-organisé et rapproché pour système automatisé flexible – application à un hub de transbordement robotisé fer/fer'. Journal Européen des Systèmes Automatisés, vol. 35, n° 7-8, 2001, pp. 905-932.

[Rabelo et al., 99] R. J. Rabelo, L. M. Camarinha-Matos and H. Afsarmanesh 'Multi-agent-based agile scheduling' In Robotics and Autonomous Systems, Volume 27, Issues 1-2, 30 April 1999, Pages 15-28.

[Ramadge et Wonham, 87] Ramadge P.J. and Wonham W.M. 'Supervisory control of a class of discrete event processes'. SIAM Journal of Control and Optimization, vol. 25, n°1, 1987.

[Rasmussen, 83] J. Rasmussen, Skill, Rules and knowledge: signals, signs and symbols and other distinctions in human performance models, IEEE SMC, vol. 13, n° 3, 1983, pp. 257-266.

[Roy et Bouyssou, 93] Bernard Roy et Denis Bouyssou 'Aide multicritère à la décision : Méthodes et cas'. Éditions Economica 1993.

[Saad et al., 97] A. Saad, G. Biswas, and K. Kawamura 'Performance Evaluation of Contract Net-Based Heterarchical Scheduling for Flexible Manufacturing Systems'. In International Journal of Intelligent Automation and Soft Computing, special issue on Intelligent Manufacturing and Shop Floor Control, vol. 3, no. 3, pp. 233-252, AutoSoft Press, 1997.

[Sallez et al., 04] Y. Sallez, D. Trentesaux, T. Berger et C. Tahon 'Product-based and resource-based heterarchical approaches for dynamic FMS scheduling'. In Computers & Industrial Engineering 46 (2004) 611–623.

[Sebag et Schoenauer, 96] Michèle Sebag et Marc Schoenauer 'Contrôle d'un algorithme génétique'. In Revue d'Intelligence Artificielle, Volume 10, pages 389-428.

[Selen et Ashayeri, 01] Willem J. Selen et Jalal Ashayeri 'Manufacturing cell performance improvement: A simulation study' in Robotics and Computer Integrated Manufacturing, volume 17, pages 169-176, (2001).

[Sénéchal, 04] Olivier Sénéchal 'Pilotage des systèmes de production vers la performance globale'. Habilitation à diriger des recherches, Université de Valenciennes et du Hainaut-Cambrésis, 2 septembre 2004.

[Sevaux, 04] Marc Sevaux 'Méta heuristiques : Stratégies pour l'optimisation de la production de biens et de services'. Habilitation à diriger des recherches, Université de Valenciennes et du Hainaut-Cambrésis, 1er Juillet 2004.

[Sharifi et Zhang, 99] H. Sharifi et Z. Zhang 'A methodology for achieving agility in manufacturing organizations: An introduction'. In the Int. J. Production Economics vol. 62, pages 7-22, 1999.

[Shen et al., 98] W. Shen, F. Maturana et D. H. Norrie 'Learning in Agent-Based Manufacturing Systems'. In Proceedings of the AAAI's Special Interest Group in Manufacturing Workshop on Artificial Intelligence and Manufacturing: State of the Art and State of Practice, Albuquerque, New Mexico, 31 Août-2 Septembre 1998, pages 177-183.

[Shen et Norrie, 99] W. Shen et D.H. Norrie, Agent-Based Systems for Intelligent Manufacturing: A State-ofthe-Art Survey. Knowledge and Information Systems, an International

Journal, vol. 1, n° 2, 1999, pp. 129-156.

[Simão et al., 06] Jean Marcelo Simão, Paulo Cézar Stadzisz and Gérard Morel 'Manufacturing execution systems for customized production'. Journal of Materials Processing Technology, Volume 179, Issues 1-3, 20 October 2006, Pages 268-275.

[Simon, 77] H. A. Simon., The new science of management decision, Prentice-Hall, New-Jersey, USA, 1977.

[Smart et Kaelbling, 02] William D. Smart and Leslie Pack Kaelbling 'Effective Reinforcement Learning for Mobile Robots' in International Conference on Robotics and Automation, May 11-15, 2002

[Smith, 04] Shana Shiang-Fong Smith 'Using multiple genetic operators to reduce premature convergence in genetic assembly planning'. In Computers in Industry Volume 54, Issue 1, May 2004, Pages 35-49

[Sødal, 06] Sigbjørn Sødal 'Entry and exit decisions based on a discount factor approach'. Journal of Economic Dynamics and Control, Volume 30, Issue 11, November 2006, Pages 1963-1986.

[Swarm, 05] Projet européen Swarm bots URL http: //www.swarm-bots.org/

[Tchako, 94] J.F. Noubissie-Tchako 'Contribution à la conception d'un système de pilotage distribué pour les systèmes automatisés de production'. Thèse de Doctorat, Université de Valenciennes et du Hainaut-Cambrésis, 1er décembre 1994.

[Tharumarajah et al., 96] A. Tharumarajah, A.J. Wells et L. Nemes 'Comparison of the bionic, fractal and holonic manufacturing systems concepts'. International Journal of Computer Integrated Manufacturing, vol. 9, n° 3, 1996, pp. 217-226.

[Thiel, 93] D. Thiel, Enquête et analyse des décisions dans les systèmes de production, APII, vol. 27, n°2, 1993, pp. 167-188.

[Tranvouez et al., 99] E. Tranvouez, B. Espinasse et A. Ferrarini, Résolution coopérative et distribuée de problèmes : une application multi-agents au ré-ordonnancement d'atelier, congrès GI, Montréal, 1999, pp.1543-1552.

[Trentesaux et al., 97] D. Trentesaux, N. Moray et C. Tahon, Integration of the Human Operator into Responsive Discrete Production Management Systems, 7ème int. Conf. EURO 1997, (Bruges, Belgique, mars 1997), p. 66.

[Trentesaux, 02] Damien Trentesaux 'Pilotage hétérarchique des systèmes de production'. Habilitation à diriger des recherches, Université de Valenciennes et du Hainaut-Cambrésis, 19 décembre 2002.

[Trentesaux, 96] D. Trentesaux 'Conception d'un système de pilotage distribué, supervisé et multicritère pour les systèmes automatisés de production'. Thèse de Doctorat en Sciences, Institut National Polytechnique de Grenoble, 24 janvier 1996.

[Ueda, 01] Kanji Ueda 'Emergent Synthesis research overview' in Artificial Intelligence in Engineering, Volume 15, Issue 4, October 2001, Pages 319-320.

[Vaario et al., 97] Jari Vaario, Nobutada Fujii, Dirk Scheffter, Michael Mezger and Kanji Ueda 'Factory Animation by Self-Organization Principles' in Proceedings of International Conference on Virtual Systems and MultiMedia (VSMM'97), September 10-13, 1997, Geneva, Switzerland, pages 235-242.

[Valckenaers et al., 98] Paul Valckenaers, Hendrik Van Brussel, Jo Wyns, Luc Bongaerts et Patrick Peeters 'Designing Holonic manufacturing systems» in Robotics and Computer Integrated

Manufacturing N°14 (1998) pages 455-464.

[Van Brussel et al., 98] H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts et P. Peeters 'Reference architecture for holonic manufacturing systems: PROSA'. Computers in industry, Elsevier, vol. 37, 1998, pp. 255-274.

[Vits et Gelders, 02] Jeroen Vits et Ludo Gelders 'Performance improvement theory' in International Journal of Production Economics, Volume 77, Issue 3, 11 June 2002, pages 285-298.

[Whitley et al., 93] D. Whitley, S. Dominic, R. Das, C. Anderson 'Genetic reinforcement learning for neurocontrol problems'. In Machine Learning, Vol. 13, pages 259-284, 1993.

[Wong et McFarlane, 06] Alex C.Y. Wong and Duncan McFarlane 'RFID and Product Intelligence'. Workshop on Ambient Interlligence Technologies to Enhance the Product Lifecycle, February 2006, Brussels.

[Yamamoto et al., 01] Masaru Yamamoto, Job de Haan and Gerben Lovink 'Production planning in Japan: rediscovering lost experiences or new insights?'. International Journal of Production Economics, Volume 71, Issues 1-3, 6 May 2001, Pages 101-109

[Yang et Li, 02] S.L. Yang et T.F. Li 'Agility evaluation of mass customization product manufacturing'. In the Journal of Materials Processing Technology 129 (2002) 640–644.

[Yusuf et al., 99] Y. Y. Yusuf, M. Sarhadi and A. Gunasekaran 'Agile manufacturing : The drivers, concepts and attributes'. In International Journal of Production Economics, Volume 62, Issues 1-2, 20 May 1999, Pages 33-43.

[Zaremba et Morel 03] Marek B. Zaremba et Gérard Morel 'Integration and control of intelligence in distributed manufacturing'. International Journal of Intelligent Manufacturing. Eds. A. Kusiak, Special Issue on Internet-Based Distributed Intelligent Manufacturing Systems (Ed. Z. Banaszak) Vol.14 n°1, pages 25-42, Janvier 2003.

[Zwingelstein, 95] G. Zwingelstein 'Diagnostic des défaillances : théorie et pratique pour les systèmes industriels'. Traité des nouvelles technologies, éditions. hermès, Paris, 1995.

ANNEXE 1

1. Agilité

D'une façon plus spécifique, Sharifi et Zhang [Sharifi et Zhang, 99] présentent l'agilité dans le domaine industriel comme un facteur déterminant pour obtenir les meilleures performances. Cette amélioration des performances a pour origine une réaction rapide aux changements et l'utilisation judicieuse de nouvelles stratégies et méthodes de management. Les auteurs s'accordent pour dire qu'actuellement le prix de revient d'un produit ne représente plus à lui seul l'élément qui détermine la performance d'une entreprise mais plutôt d'autres facteurs tels que la qualité, le délai de livraison et plus globalement la satisfaction du client. L'agilité représente un paradigme qui vise à atteindre cet objectif.

Avec une approche plus expérimentale, Cagliano et al. [Cagliano et al., 04] étudient empiriquement l'impact de l'agilité sur les performances d'un système de production de type chaîne logistique (données réelles provenant de plusieurs entreprises européennes). Leurs conclusions montrent d'une part que l'agilité donne une grande amélioration des performances surtout en terme de délai (lead time) et d'autre part qu'elle représente une approche viable.

Dans un but d'évaluation, Giachettia et al. [Giachettia et al., 03] proposent des indicateurs pour mesurer la flexibilité et l'agilité relatives aux aspects structurels et opérationnels des systèmes de production. Cette étude montre d'une part que l'on dispose de plusieurs indicateurs pour mesurer la flexibilité (flexibilité de routage, opérationnelle, de gamme, de volume) et ceci pour les différents types d'entités (produit, opération, machine, système de production). D'autre part, il existe peu d'indicateurs pour mesurer l'agilité. Dans ce sens, [Yang et Li, 02] proposent des indicateurs destinés à l'évaluation de l'agilité d'un type particulier des systèmes de production (mass customization ou production personnalisée). Ces indices sont déterminés avec une méthode s'appuyant sur la logique floue (extrêmement agile, agile, généralement agile, non agile et extrêmement non agile).

Dans [Gunasekaran, 99], Gunasekaran développe un cadre méthodologique définissant l'agilité caractérisé par quatre paramètres :

- les stratégies : elles couvrent les aspects organisationnels mis en place dans un SPBS (par exemple les chaînes logistiques, les entreprises virtuelles, …),
- les technologies : elles concernent les systèmes d'informations, les équipements de production employés pour assurer le fonctionnement d'un SPBS,
- le système de pilotage : ce paramètre inclut toutes les fonctions relatives au cycle de vie d'un

produit ou d'un service (la conception, la planification, l'ordonnancement, …),

- le facteur humain : ce dernier élément est relatif aux compétences des opérateurs humains et à leur acquisition (apprentissage) de nouveaux savoir-faire.

2. Aperçu des approches d'amélioration continue des performances des SPBS

2.1. Les systèmes de gestion de type Kanban

Les modèles des flux tendus ou flux tirés représentent une logique de production totalement différente par rapport à celle existante depuis la fin de la deuxième guerre mondiale. Cette logique est dictée par les besoins et les SPBS ne produisant que le strict nécessaire afin de réduire au maximum les opérations sans valeurs ajoutées. Cette approche repose sur des principes tels que la réduction des stocks ou encore l'utilisation de la totalité du potentiel productif des opérateurs. Pour cela, ces opérateurs doivent avoir une formation qualifiée de haut niveau pour assumer localement le fonctionnement de la chaîne de production. Une autre notion qui est importante consiste à l'application de la maintenance prédictive du système de production pendant le temps inoccupé (appelé aussi temps masqué). Cette action limite les rebuts et favorisent la qualité de la fabrication. L'inconvénient de ce type de production est illustré par le surdimensionnement de ces systèmes pour avoir la flexibilité et la réactivité.

La méthode Kanban[81] est la plus connue des méthodes par flux tirés. C'est une méthode décentralisée de gestion de flux basée sur des ordres de fabrication implicites. Les étiquettes sont traitées manuellement ce qui implique directement les salariés et contribue ainsi à leur intéressement [Giard, 03]. La méthode Kanban a été mise en place vers la fin des années 50 dans les usines automobiles Toyota. Il s'agit d'un un système d'information qui fonctionne entre deux postes de travail et qui limite la production du poste amont aux besoins exacts du poste aval. La méthode de gestion par kanban constitue une approche de mise en œuvre du concept de Juste À Temps. Les objectifs de ce concept sont : une gestion visuelle du flux des pièces, la maîtrise des encours en temps réel et la responsabilisation des acteurs sur la gestion des flux. Les résultats sont souvent impressionnants. Il n'est pas rare de constater un doublement de la productivité, une diminution sensible des niveaux de stocks et des aires de stockages intermédiaires tout en constatant une diminution des ruptures de stock. Schématiquement, le système Kanban fonctionne entre les postes de production avals et amonts :

- quand l'opérateur aval entame un conteneur, il libère alors le kanban de manutention fixé sur le conteneur et le dispose dans une boîte,
- le manutentionnaire ramasse le kanban de manutention et va au poste amont,
- au poste amont, il enlève le kanban de production du conteneur plein, le met dans une autre boîte et lui substitue le kanban de manutention,
- il ramène le conteneur plein avec le kanban de manutention au poste aval,
- quand l'opérateur du poste amont a rempli un conteneur, il regarde la boîte de kanbans de production. S'il y a un kanban, il l'enlève, le fixe à un conteneur vide et reprend la production. S'il n'y a pas de kanban, cela veut dire que les en-cours sont suffisants et il attend, voir Figure 111 :

[81] Le kanban est une simple fiche cartonnée que l'on fixe sur les bacs ou les conteneurs de pièces.

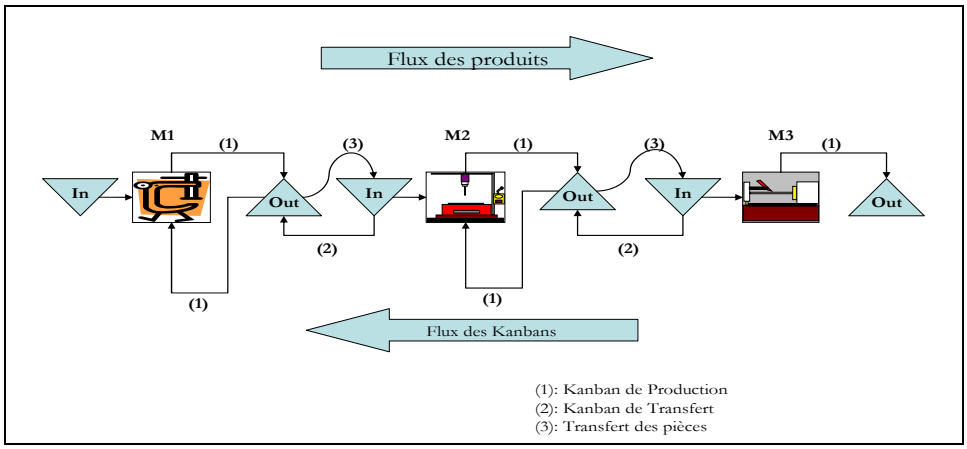


Figure 111- Schéma général d'un système de production à Kanbans

Par ailleurs, nous distinguons couramment trois types de kanbans :

- des kanbans de production dimensionnés par le poste goulet et par le cycle de production,
- des kanbans d'approvisionnement dimensionnés par le conditionnement du fournisseur, le coût de transport par lot et le délai négocié avec le fournisseur,
- des kanbans de transfert dans le cas où un poste alimenterait plusieurs postes en aval dont la taille des lots est différente. Ce type de kanban est dimensionné par les postes aval, pour des délais nuls.

2.2. Le Kaizen

Le mot Kaizen signifie une amélioration qui implique tous les acteurs des directeurs aux ouvriers [Bessant et al., 01]. Cette démarche se base sur des améliorations faites constamment et graduellement. Il s'agit par exemple de rendre le travail des opérateurs plus productif, moins fatiguant, plus efficace et sûr. Une autre démarche consiste à améliorer les équipements, notamment en changeant la disposition des unités de production ou encore la révision des procédures de travail. Le Kaizen se base sur le cycle de Deming [Hales et Chakravorty, 06] qui est caractérisé par les quatre étapes suivantes : Plan, Do, Check et Act (PDCA). La roue de Deming (son nom vient du statisticien Edwards Deming) est une illustration de la méthode qualité PDCA. Cette méthode comporte quatre étapes, chacune entraînant l'autre, et vise à établir un 'cercle vertueux', voir Figure 112 :

- plan : la première étape consiste à planifier la réalisation, étudier la situation actuelle et rassembler les différents programmes et informations. Il s'agit par exemple d'écrire le cahier des charges ou l'établissement d'un planning.
- do : ces différents programmes sont expérimentés à travers des essais.
- check : une fois l'étape précédente achevée, on entame l'étape Check (de l'anglais vérifier) qui consiste à contrôler que le travail (Do) correspond bien à ce qui était prévu (Plan) et s'il y a des améliorations à apporter. Cette étape utilise des moyens de contrôle divers, tels que les indicateurs de performance.
- act : cette étape consiste à rechercher des points d'améliorations.

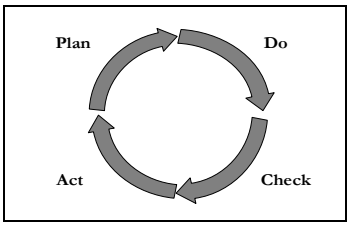


Figure 112- Les quatre étapes principales du Kaizen

2.3. Autres méthodes

Nous pouvons citer d'autres méthodes qui peuvent s'intégrer dans une logique d'amélioration continue des performances des SPBS, notamment les méthodes de marketing ou encore la méthode Rex.

La première approche consiste à appliquer des actions a posteriori auprès des clients via des enquêtes (voir [Intel, 02]). Selon les tendances recueillies et le degré de satisfaction, les industriels procèdent à des changements de fond (par exemple la composition) ou de forme (l'emballage ou la couleur). Ce type de procédure accompagne généralement le lancement d'une nouvelle gamme de produits (textiles, cosmétiques). Un autre exemple caractérisant cette démarche est celui des opérateurs téléphoniques ou les fournisseurs d'accès Internet qui s'efforcent de réduire le temps d'attente de leurs clients lors des appels vers la hotline. C'est une forme, parmi d'autres, d'amélioration continue des performances.

Dans une application de gestion des connaissances Rex, la première tâche est dédiée au recueil des connaissances (interviews de spécialistes et d'analyse de documents de références du domaine). Les connaissances acquises sont stockées dans un système informatique de gestion de documents qui utilise des modèles du domaine pour mettre en correspondance un ensemble d'éléments de connaissances et une question d'un utilisateur. La construction des modèles du domaine utilisés par le système de gestion de documents constitue la deuxième tâche à réaliser dans une application Rex. Pour construire les modèles, la méthode Rex préconise d'utiliser les thesaurus ou dictionnaires disponibles dans l'entreprise. Si de telles ressources n'existent pas, une lecture systématique des éléments de connaissance permet de structurer dans les modèles les relations entre les termes utilisés dans ces éléments de connaissance. Ces modèles permettront de spécialiser et généraliser les requêtes utilisateurs.

Nous pouvons également considérer d'autres paramètres de la performance comme facteurs effectifs ou potentiels d'amélioration continue : recherches et développement, management des connaissances, la réingénierie des processus (administratifs, opérationnels,…), la coopération, … Notons enfin que ces approches ne se situent pas au même niveau. En effet, elles peuvent être appliquées soit au niveau stratégique, tactique ou opérationnel (voir [Tchako, 94]).

ANNEXE 2

1. Apprentissage

1.1. Typologie de l'apprentissage

Plusieurs auteurs ([Beslon, 95], [Shen et al., 98], [Monostori, 99]) s'accordent pour distinguer trois types d'apprentissage : l'apprentissage supervisé, non supervisé et hybride.

- l'apprentissage supervisé (qualifié aussi d'apprentissage par des exemples) est caractérisé par la présence d'un 'professeur' qui possède une connaissance approfondie de l'environnement dans lequel évolue le système. Ce type d'apprentissage possède une inconvénient : en absence de professeur pour fournir les valeurs cibles, ce type d'apprentissage ne peut d'aucune façon apprendre de nouvelles stratégies pour de nouvelles situations qui ne sont pas couvertes par les exemples d'apprentissage. L'environnement dans lequel opère le système supervisé produit un stimulus qui est acheminé à la fois au professeur et au système. Grâce à ses connaissances intrinsèques, le professeur produit une sortie désirée pour ce stimulus. Cette réponse est ensuite comparée par soustraction avec la sortie du système pour produire un signal d'erreur qui est réinjecté dans le système pour modifier son comportement,
- l'apprentissage non supervisé (appelé aussi 'auto-organisé') est caractérisée par l'absence complète de professeur, c'est-à-dire que nous ne disposons pas d'un signal d'erreur, comme dans le cas supervisé. Dans ce type d'apprentissage nous ne disposons que d'un environnement qui fournit des stimuli et d'un système qui doit apprendre sans intervention externe,
- apprentissage hybride : ce type d'apprentissage est une combinaison d'un apprentissage supervisé et non supervisé,

1.2. Techniques d'apprentissage

Dans la littérature qui traite ce domaine de recherche, trois techniques principales d'apprentissage peuvent être distinguées :

1.2.1. Apprentissage à base de cas

L'apprentissage à base de cas s'intègre dans une technique plus globale de raisonnement à base de cas (ou Case Based Reasoning CBR). Pour résoudre un problème donné, un système CBR dispose d'une base de cas (ou de connaissances) dans laquelle chaque cas possède une description et une solution à un problème. Cette technique résout les problèmes en retrouvant des cas analogues dans sa base de connaissances et en les adaptant au cas considéré. La représentation des cas (les informations à stocker et leur forme) détermine l'efficacité et la rapidité de la recherche des

cas dans la base. Les différentes étapes d'un CBR sont données dans la Figure 113 :

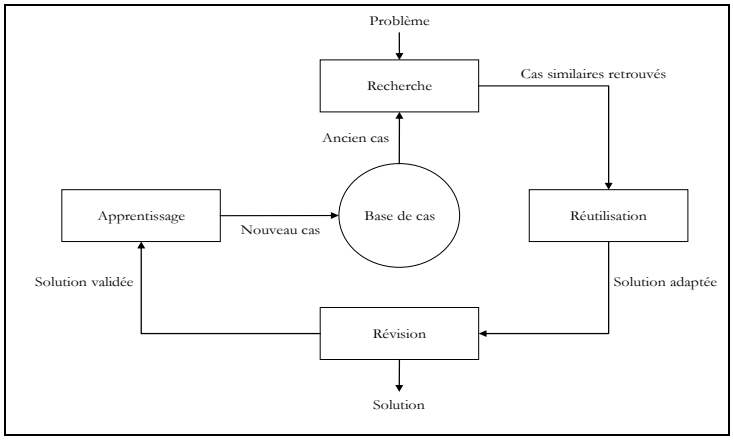


Figure 113- Schéma général d'un processus d'apprentissage à base de cas

Un système de raisonnement à base de cas est composé de quatre étapes principales :

- la recherche des cas similaires qui se décompose en deux phases :
  - l'étape de filtrage qui consiste à réduire au préalable le nombre de cas utilisés dans la recherche,
  - l'étape de sélection qui, à partir de l'ensemble de cas obtenus lors de l'étape de filtrage, construit un nouvel ensemble de cas similaires en utilisant des algorithmes (généralement des heuristique) pour permettre de mesurer la similarité entre le problème posé et les cas candidats.
- la réutilisation de cas et leur adaptation : dans les systèmes CBR simples, lorsqu'un cas similaire est retrouvé, la solution qu'il propose pour le problème courant est réutilisée directement. Or, cette façon de procéder est peu satisfaisante. En effet, il est rare qu'un un cas identique au problème soit trouvé, il est alors souvent nécessaire d'adapter les solutions préexistantes. L'adaptation consiste à construire une nouvelle solution à partir du problème courant et des cas similaires trouvés,
- la révision : la solution du problème générée par le système CBR est testée. Ce test fait souvent appel à un logiciel de simulation ou à un expert. Si cette évaluation est concluante, cette nouvelle expérience est retenue. Cependant si la solution n'est pas satisfaisante, il faut la réparer ou tout au moins expliquer les raisons de l'échec : c'est la phase de révision. Ainsi, lors des prochaines générations de solutions, le système ne répétera pas les mêmes erreurs,
- l'apprentissage : c'est la dernière étape du cycle du CBR. Cette phase consiste à une mémorisation du problème cible enrichi de sa solution. Le nouveau cas et sa solution validée vont être ajoutés à la base de cas.

1.2.2. Apprentissage par renforcement

L'apprentissage par renforcement repose sur un système prédéfini de récompense/pénalité. Une décision prise par une entité (entité physique ou abstraite) ayant donnée de bonnes satisfactions se voit récompensée et elle sera pénalisée dans le cas contraire. Glorennec [Glorennec, 03] définit l'apprentissage par renforcement (ou apprentissage avec un critique) comme l'émergence de comportements permettant d'atteindre un objectif sans autre information qu'un signal scalaire (appelé renforcement). Cette définition générale met en évidence deux caractéristiques importantes, voir Figure 114 :

- l'entité interagit avec son environnement et l'ensemble {entité, environnement} constitue un système dynamique,
- le signal de renforcement, qui est généralement perçu en termes de récompense ou pénalité, doit permettre à une entité de modifier son comportement.

La démarche générale d'un apprentissage par renforcement est la suivante :

1ère étape : l'entité est dans l'état x(t)
Elle choisit une des actions possibles dans cet état, a(t)
2ème étape : l'entité applique l'action, qui provoque
Le passage dans un nouvel état, x(t+1)
La réception du renforcement, r(t+1)
t←t +1
Retour à la deuxième étape ou fin si l'état atteint est un état terminal.

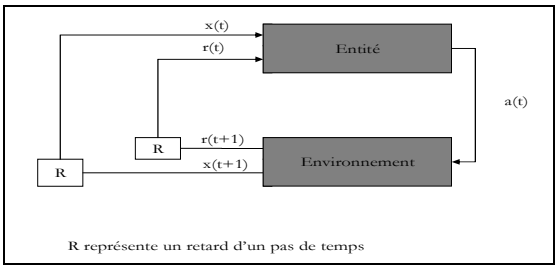


Figure 114- Les étapes d'un apprentissage par renforcement (d'après [Glorennec, 03])

Signalons enfin que ce type d'apprentissage est très utilisé en robotique (détermination des mouvements des robots, évitement des obstacles, … voir [Brooks, 91] ou [Smart et Kaelbling, 02]) et a donné des résultats intéressants.

1.2.3. Apprentissage par réseaux de neurones

Un réseau de neurones (Artificial Neural Network) est un modèle de calcul dont la conception est inspirée du fonctionnement de vrais neurones. Les réseaux de neurones sont généralement optimisés par des méthodes d'apprentissage de type statistique ou des méthodes de l'intelligence artificielle. Les paramètres les plus importants de ce modèle sont les coefficients synaptiques. Ce sont eux qui construisent le modèle de résolution en fonction des informations données au réseau. Il faut donc trouver un mécanisme qui permette de les calculer à partir des grandeurs que l'on peut acquérir du problème : c'est le principe fondamental de l'apprentissage [Whitley et al., 93]. Dans un modèle de réseaux de neurones formels, apprendre, c'est d'abord calculer les valeurs des coefficients synaptiques en fonction des exemples disponibles. Les premiers réseaux de neurones capables d'apprendre par expérience ont été proposés par Franck Rosenblatt en 1957 avec le modèle du perceptron.

Le processus d'apprentissage dans les réseaux se base sur des observations limitées pour tirer des généralisations plausibles. Ainsi, cette notion d'apprentissage recouvre deux points successifs :

- mémorisation : qui consiste à assimiler sous une forme dense des exemples éventuellement nombreux,
- généralisation : qui exprime la capacité, grâce aux exemples appris, de traiter des exemples distincts, encore non rencontrés, mais similaires.

ANNEXE 3

1. Formalisme de modélisation objet UML (Unified Modeling Langage)

1.1. Présentation

Les langages de modélisation orientés objets ont fait leur apparition entre le milieu des années soixante-dix et la fin des années quatre-vingt quand les spécialistes méthode, confrontés à un nouveau genre de langages de programmation orientés objet et à des applications de plus en plus complexes, ont commencé à expérimenter de nouvelles approches d'analyse et de conception. Plusieurs méthodes orientées objet sont alors apparues. La majorité des utilisateurs ne trouvaient pas de langage de modélisation qui réponde entièrement à leurs besoins. Basées sur l'expérience acquise, de nouvelles générations de ces méthodes sont apparues. Certaines d'entre elles se sont nettement détachées du lot. Plus spécialement sont apparues les méthodes Booch, OOSE (Object-Oriented Software Engineering) de Jacobson et OMT (Object Modeling Technique) de Rumbaugh. Chacune de ces méthodes constituait une méthode complète mais présentait aussi bien des avantages que des inconvénients. En effet, la méthode Booch était particulièrement expressive lors des phases de conception et de construction des projets, OOSE représentait un excellent outil pour les cas d'utilisation en matière de définition des exigences, d'analyse et de conception générale, et OMT était plus particulièrement utile à l'analyse et aux systèmes d'information contenant une grande quantité de données.

En tant que principaux auteurs des méthodes Booch, OOSE et OMT, Gary Booch, Ivar Jacobson et James Rumbaugh ont souhaité créer un langage de modélisation unifié. Les travaux ont officiellement commencé en octobre 1994. La première version de la méthode unifiée (Unified Method) fut publiée en octobre 1995. La première version d'UML est apparue en juin 1996.

UML est défini comme un langage standard conçu pour l'écriture des plans d'élaboration des logiciels. Il est utilisé pour visualiser, spécifier, construire, et documenter les artefacts d'un système à forte composante logicielle.

1.2. Les briques de base d'UML

La terminologie d'UML inclut trois sortes de briques : les éléments, les relations, et les diagrammes.

1.2.1. Les éléments d'UML

Il existe quatre types d'éléments dans UML :

- les éléments structurels : sont représentés par des noms dans les modèles UML. Ce sont les parties les plus statiques d'un modèle. Il existe sept types d'éléments structurels largement détaillés dans [Booch et al., 00] : les classes, les interfaces, les collaborations, les cas d'utilisation, les classes actives, les composants et les nœuds,
- les éléments comportementaux : représentent les parties dynamiques des modèles UML. Ce sont les verbes du modèle et ils représentent son comportement dans le temps et dans l'espace (par exemple les messages),
- les éléments de regroupement : représentent les parties organisationnelles des modèles UML. Ce sont les boîtes dans lesquelles un modèle peut être décomposé (par exemple les paquetages),
- les éléments d'annotation : représentent les parties explicatives des modèles UML. Ce sont les commentaires qui peuvent accompagner tout élément dans un modèle à des fins de description, d'explication et de remarque (par exemple les notes).

1.2.2. Les relation dans UML

Il existe quatre types de relations dans UML :

- la dépendance : est une relation sémantique entre deux éléments selon laquelle un changement apporté à l'un peut affecter la sémantique de l'autre,
- l'association : est une relation structurelle qui décrit un ensemble de liens. Un lien constitue une relation entre deux objets. L'agrégation est un type particulier d'association,
- la généralisation : est une relation de spécialisation/généralisation selon laquelle les attributs de l'élément spécifié (enfant) peuvent se substituer aux attributs de l'élément généralisé (parent). L'enfant partage la structure et le comportement du parent,
- la réalisation : est une relation sémantique entre classificateurs, selon laquelle un classificateur spécifie un contrat dont l'exécution est garantie par un autre classificateur.

1.2.3. Les diagrammes dans UML

Un diagramme est une représentation graphique d'un ensemble d'éléments qui constitue un système. UML comprend 9 diagrammes dont les principaux sont :

- diagramme de classes : l'objectif du diagramme de classes est de représenter la vue de conception statique d'un système. Le diagramme de classes inclut des classes, des attributs associés à ces classes, des relations de collaboration entre classes et des opérations qui manipulent ces classes,
- diagramme de cas d'utilisation : un cas d'utilisation est un ensemble de séquences d'actions réalisées par le système et produisant un résultat observable intéressant pour un acteur particulier. Chaque cas d'utilisation spécifie un comportement attendu du système. Il permet de décrire ce que le système devra faire, sans spécifier comment il le fera. L'objectif d'un diagramme de cas d'utilisation est de représenter la vue de conception dynamique d'un système. Il montre un ensemble de cas d'utilisation, d'acteurs, et leurs relations. Il permet de visualiser et spécifier le comportement des éléments du système avec les acteurs qui sont visibles de l'extérieur, de telle sorte que l'utilisateur puisse comprendre comment utiliser le système et que le développeur puisse l'implémenter (en s'appuyant sur d'autres informations plus spécifiques),

- diagramme de séquences : le diagramme de séquences est un diagramme d'interaction, qui met en œuvre le classement chronologique des messages communiqués entre les objets qui participent à l'activité de modélisation. Le diagramme de séquences est construit ainsi : les objets qui participent à l'interaction sont placés le long de l'axe des abscisses. Les messages sont envoyés et reçus par ces objets le long de l'axe des ordonnées, par ordre chronologique de haut vers le bas. Cette représentation donne au lecteur une indication claire du flot de contrôle dans le temps.

Dans la Figure 115, nous illustrons les principaux éléments, relations et diagrammes d'UML ([Booch et al., 00]).

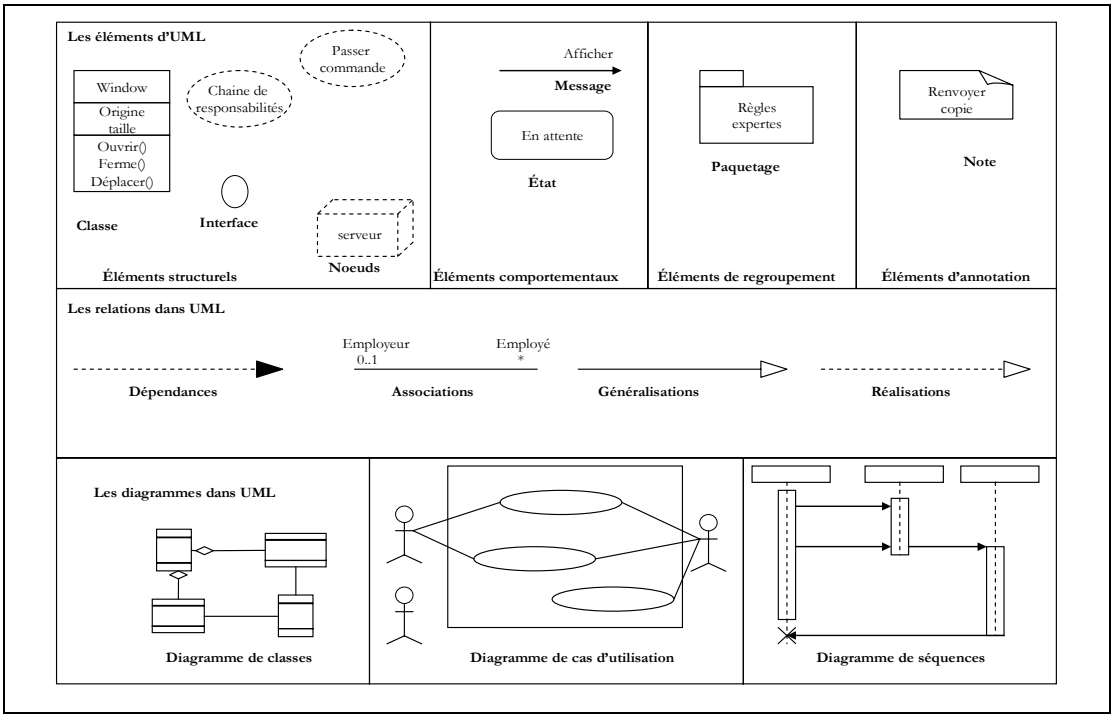


Figure 115- Les principales notations d'UML

ANNEXE 4

1. Description des principales opérations

1.1. classe product

- date du prochain événement significatif d'un produit

```
date product::nextevent(date current)
{
      //reste-t-il des tâches à réaliser ?
      if (empty())
            return infinity();
      //la tâche courante est prévue dans le futur
      if (current < front().arrival)
            return front().arrival;
      //la tâche courante est en cours de traitement ou terminée
            return front().start + front().done + front().todo;
}
```

- avance de l'exécution de la tâche courante

```
void product::step(duration x)
{
      front().todo -= x;
      front().done += x;
}
```

1.2. classe resource

- date du prochain événement significatif d'une ressource

```
date resource::nextevent()
{
      ifs = failures.upper_bound(currentdate);
      date fail = (ifs == failures.end()) ? infinity() : ifs->first;
      date tool = empty() ? infinity() : top().nextevent(currentdate);
      return min(fail, tool);
}
```

1.3. classe workshop

- date du prochain événement significatif de l'atelier

```
date workshop::nextevent()
{
//on cherche la date minimum de prochain événement de toutes les ressources
      date result = infinity();
//on parcourt la liste des ressources
```

```
      for (pos = reslist.begin(); pos != reslist.end(); ++pos)
      {
//on cherche la date la plus proche (min) d'un événement dans le futur (max)
      result = min(result, max(currentdate, pos->second->nextevent()));
}
```

Résumé

Titre 'Amélioration continue des performances des systèmes de production de biens et de services : proposition d'une architecture logique d'un système de pilotage hétérarchique évolutif par apprentissage'.

La complexité et la variabilité du contexte actuel des Systèmes de Production de Biens et de Services (SPBS) imposent une vision dynamique des performances (techniques, économiques,...) de ces systèmes, c'est-à-dire des performances continuellement améliorées. Ce besoin d'une vision dynamique des performances des SPBS est de plus en plus ressenti et exprimé par les acteurs de différents domaines de SPBS, s'inscrit dans le contexte plus global d'agilité et implique surtout de mettre en œuvre des systèmes de pilotage adaptés.

D'une part, dans un contexte fortement concurrentiel, l'agilité devient une caractéristique clef de la prospérité d'un SPBS et symbolise l'actuel objet de compétition entre les SPBS. D'autre part, la mise en place d'un système de pilotage représente un moyen adéquat pour garantir l'agilité d'un SPBS. En effet, la fonction pilotage consiste à décider dynamiquement des commandes pertinentes à donner à un système soumis à des perturbations pour atteindre un objectif donné décrit en termes de maîtrise de performances. La notion de maîtrise intègre non seulement celle de maintien d'un niveau de performance donné, mais également celle de progrès (évolution vers un niveau de performance souhaité ou amélioration continue). C'est sur cet objectif d'amélioration continue des performances des SPBS que nous nous focalisons dans cette thèse.

Cependant, les systèmes de pilotage actuels ne répondent pas efficacement à cette évolution permanente du contexte des SPBS et par conséquent l'objectif d'amélioration continue des performances reste difficilement atteignable. Plus précisément, les systèmes de pilotage sont généralement conçus pour répondre à un besoin spécifique et ce manque de généricité réduit fortement les propriétés d'agilité d'un SPBS. Nos travaux de recherche s'inscrivent dans un cadre qui porte sur le développement d'un système de pilotage pour l'amélioration continue des performances des SPBS. En ce sens, grâce aux Technologies de l'Information et de la Communication (TIC), l'accès aux informations est de plus en plus simple et efficace. Cet accès offre l'opportunité de concevoir des systèmes de pilotage hétérarchiques qui favorisent la décentralisation des capacités décisionnelles.

L'objectif de ce mémoire est de proposer une caractérisation du paradigme d'agilité des systèmes de production de biens et de services, un cadre de modélisation générique des systèmes de production de biens et de services et un système de pilotage hétérarchique évolutif par apprentissage pour l'amélioration continue des performances des SPBS.

Mots clefs : système de production de biens et de services, agilité, amélioration continue des performances, pilotage hétérarchique, stratégies, autonomie, mécanismes génétiques, apprentissage par renforcement, simulation.

Abstract

Title: 'Performances' continuous improvement of the Production Systems of Goods and Services: proposal of a logical architecture of an evolutionary heterarchical control system based on learning capacities'.

We are interested in this study in the performances' continuous improvement of the Production Systems of Goods and Services (PSGS) which organization and structure of control have significantly evolved these last years. In this work, we propose a systemic modelling of the PSGS. This modelling is based on the concept of entity and takes account of the structural and functional aspect of a PSGS. Indeed, a first distinction based on the structure of any PSGS system will give two classes of entities: fixed entities and mobile entities. A second characterisation based on the functional aspect of the entities brings out four subsets: fixed entities of treatment and storage (noted respectively fet(k) and fes(k)) and mobile entities to treat and of transfer (noted respectively mep(i) and met(j)). This modelling is generic in the sense that it makes it possible to represent a PSGS (machines, queues, parts, AGV) as well as other fields (networks, transport). In addition, we propose an original approach of control based on a total distribution of the decision-making capacities on the various entities which compose the system. This local decision-making is based on a set of strategies which takes into account the real state of the system. Moreover, capacities of learning are integrated in the control system. The learning's mechanisms consist on an evaluation a posteriori of the performances realized by the last entities which leave the system (respectively given by the last decisions' cycles) in order to determine the best strategy to adopt for the entering ones (respectively for the new decisions' cycle). The results obtained through a simulation using a discret-event showed the interest of our approach.

Keywords: production systems of goods and services, agility, entities based modelling, continuous performances' improvement, heterarchical control, autonomy, reinforcement learning, evolutionist algorithms and simulation.