50376
1996
472

UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE

N° d'ordre : 1841

**Thèse**

présentée en vue de l'obtention du grade de

**DOCTEUR DE L'UNIVERSITE**

Spécialité

Automatique et Informatique Industrielle

Présentée par

**Jalel AKAICHI**

# Systèmes Automatisés de Production à Intelligence Distribuée

## Des Stratégies de Répartition Basées sur une Approche de Classification

Soutenue publiquement le 6 Novembre 1996 devant la commision d'examen

### Membres du jury

| | |
|---|---|
| **M. Bayart** | Professeur, École Universitaire Des Ingénieurs de Lille, **Codirectrice** |
| **E. Craye** | Professeur, École Centrale de Lille, **Président** |
| **M. Moalla** | Professeur, Faculté des Sciences de Tunis, **Examinateur** |
| **R. Soenen** | Professeur, Université de Valenciennes, **Rapporteur** |
| **F. Simonot-Lion** | Maître de conférence, Institut National Polytechnique de Lorraine, **Rapporteur** |
| **M. Staroswiecki** | Professeur, Université des Sciences et Technologie de Lille, **Codirecteur** |
| **J. P. Thomesse** | Professeur, Institut National Polytechnique de Lorraine, **Rapporteur** |

SCD LILLE 1
D 030 230287 4

# Avant-Propos

Le travail présenté dans ce mémoire a été effectué au sein de l'équipe "Instrumentation intelligente" du Laboratoire d'Automatique et d'Informatique Industrielle de Lille (L.A.I.L. URA CNRS 1440). Il a été réalisé dans le cadre d'un contrat (N° 92.P.0239) avec le Ministère de la Recherche et de l'Enseignement Supérieur.

Je remercie vivement :

Monsieur **M. Staroswiecki**, Professeur à l'Ecole Universitaire Des Ingénieurs de Lille et directeur-adjoint du L.A.I.L., pour avoir mis à ma disposition tous les moyens pour mener à bien ce travail. Ses conseils, ses connaissances et son soutien moral étaient déterminants pour la réalisation de ce travail.

Mademoiselle **M. Bayart**, Professeur à l'Ecole Universitaire Des Ingénieurs de Lille et directeur de l'Atelier Régional de Micro-Informatique, pour son aide scientifique et morale considérable. Outre ses conseils, ses qualités humaines étaient décisives pour mener ce travail à son terme. Je la remercie également pour son amitié.

Monsieur **E. Craye**, Professeur à l'Ecole Centrale de Lille, pour l'honneur qu'il me fait, en présidant le jury de ma thèse, et pour avoir accepté d'examiner ce travail.

Monsieur **J. P. Thomesse**, pour l'honneur qu'il m'accorde en rapportant mon travail. Ses critiques, ses conseils étaient d'une grande importance pour la réalisation de cette thèse.

Monsieur **R. Soenen**, pour l'honneur qu'il me fait en rapportant mon travail.

Madame **F. Simonot-Lion**, pour l'honneur qu'elle me fait en rapportant ce travail, et pour ses précieux conseils, ses critiques constructives et son amitié.

Monsieur **M. Moalla**, pour l'honneur qu'il me fait en examinant ce travail, et pour l'intérêt qu'il a porté pour mes recherches pendant les discussions qu'on a eu à Tunis.

Monsieur **K. Mellouli**, mon honorable professeur tout au long des études précédant la thèse. Ses encouragements, son soutien m'ont beaucoup aidé pour progresser dans mes recherches.

Les membres du groupe **MRES 2033** pour l'échange d'idées fructueux qu'on a eu ensemble. Je remercie particulièrement Madame F. Simonot-Lion, Monsieur J. P. Thomesse et Monsieur Luis Vega.

Ma famille, et mes amis.

Tout le personnel et les membres du L.A.I.L., de l'A.R.E.M.I. et de l'I.S.G. Je remercie particulièrement **P**aul, **S**ylvain, **C**hris, **M**ichel, **M**alika, **C**laudie.

Mon intérêt réside dans le futur car
je me prépare à y passer le reste de
ma vie.

**C. F. Kettering**

# Présentation générale

*1 Contexte du travail* ..... *8*
*2 Problématique* ..... *8*
*3 Plan de la thèse* ..... *10*

## 1 Contexte du travail

Le travail présenté dans cette thèse, s'est déroulé dans le cadre d'un projet soutenu par le Ministère de Recherche et de l'Enseignement Supérieur, qui s'intitule **"Impact des réseaux de terrains et de l'instrumentation intelligente dans la conception des systèmes automatisés de production"** et dont les résultats et les perspectives figurent dans le rapport de fin de contrat (convention N° 92.P.0239). Ont participé à ce projet des industriels et des laboratoires de recherche dont la liste se trouve en annexe. Notre travail s'est concentré sur la modélisation des systèmes automatisés de production à intelligence distribuée et la proposition de méthodes de répartition basées essentiellement sur une approche de classification.

## 2 Problématique

Les systèmes de production tendent vers une automatisation de plus en plus poussée. Cette dernière est la conséquence logique d'une connaissance de plus en plus approfondie et détaillée des fonctions du système, des besoins croissants de productivité, de qualité et de sécurité, ainsi que de la progression très rapide de la technologie micro-électronique. Le nombre de fonctions automatisées ou à automatiser devient de plus en plus important. L'implantation de ces nombreuses fonctions sur une architecture matérielle (comprenant entre autres des capteurs et actionneurs "intelligents", reliés dans la plupart des cas par des réseaux de terrain), devient une tâche ardue et délicate, et modifie les méthodes de conception. Les difficultés sont généralement dues :

- à l'absence d'une modélisation claire et concise des traitements assurant la réalisation des fonctions du système automatisé de production,

- à l'inexistence de stratégies de répartition de ces traitements, tenant compte de leurs caractéristiques, ainsi que de l'impact de leur projection sur les composantes matérielles qui vont les supporter. Ces dernières souffrent souvent de limites de natures diverses : capacité mémoire, capacité de l'ordonnanceur, puissance du processeur, ...

Dans la littérature, sont présentées essentiellement des stratégies qui concernent l'informatique parallèle et distribuée [Che 90], [Udi 90], [Tal 91]. Ces stratégies définissent généralement un ensemble de concepts fondamentaux tels que :

- les entités élémentaires de programmes ou d'application à répartir, appelées souvent processus,
- les fonctions d'évaluation des solutions réparties, appelées fonctions objectifs ou coûts,
- et finalement les algorithmes de répartition des entités élémentaires.

Ces stratégies nous ont paru insuffisantes et/ou inadaptées à notre contexte pour deux raisons fondamentales :

- la première est relative à la modélisation du système automatisé de production. En effet, nous pensons qu'il est primordial de caractériser les concepts relatifs aux entités élémentaires de répartition, aux relations entre ces entités, à la définition des critères et des contraintes de répartition,
- la deuxième est relative à la mesure des performances des solutions de répartition. En effet, il est important de proposer une approche adaptée au contexte du système automatisé de production, permettant d'établir une fonction coût tenant compte des critères définis, qui prendra en charge la mesure des performances des solutions en exploitant les connaissances que l'on a sur les entités élémentaires.

L'objectif de ce travail est de trouver des solutions aux deux problèmes cités ci-dessus, ainsi que de proposer des nouvelles stratégies de répartition. Notre intérêt s'est porté essentiellement sur l'utilisation des méthodes de classification hiérarchique. Cette

classe de méthodes apparaît adéquate à la structuration hiérarchique des fonctions du système d'automatisation. En plus cette approche est nouvelle dans notre contexte et n'a pas fait l'objet de travaux de recherches antérieurs, ce qui rend sa modélisation et son implémentation attractives.

## 3 Plan de la thèse

Le chapitre I propose une description des fonctions, services et traitements du système d'automatisation, qui représente une composante principale du système automatisé de production. Il présente également des concepts relatifs à l'instrumentation intelligente. Nous insisterons dans ce chapitre sur des composants particuliers appelés instruments intelligents (actionneurs et capteurs intelligents).

Le chapitre II s'intéresse à la définition et à la modélisation de l'architecture fonctionnelle du système d'automatisation. Dans une première phase, on propose un modèle de structuration. Dans une deuxième phase une nouvelle approche de modélisation des traitements, appelée modélisation par atome de distribution, est décrite. Celle ci expose en détail le concept d'entité élémentaire appelée atome de distribution. Ce dernier terme remplacera le terme traitement élémentaire, entité élémentaire ou processus.

Dans le chapitre III une étude détaillée relative aux stratégies de répartition sera présentée. Cette étude englobera la manière d'établir les fonctions d'évaluation des solutions réparties, les critères et les contraintes de répartition ainsi que quelques algorithmes de répartition de la littérature. Nous présentons également une classification des stratégies existantes.

Le chapitre IV concerne essentiellement la répartition du système d'automatisation en adoptant une approche de classification. Nous entamerons cette tâche par la modélisation du problème de répartition. Ensuite deux algorithmes heuristiques seront proposés. Ces derniers seront combinés pour donner naissance à un troisième dont

l'optimalité locale sera démontrée. Ces algorithmes ont été implémentés en C, et testés sur des exemples d'architectures fonctionnelles.

L'un des hommes tend l'arc de la programmation fonctionnelle, l'autre brandit une équation. Ils ne rapporteront peut-être pas le même gibier, mais ils traversent le fjord sur le même bateau, celui de la logique.

**Embarquons**

# Chapitre I

# Les traitements du système automatisé de production

*Introduction* ..... *14*
*I.1 Le système automatisé de production* ..... *14*
*I.2 Architectures d'un système automatisé de production* ..... *16*
*I.3 Les machines de l'architecture matérielle* ..... *17*
*I.3.1 Les instruments intelligents* ..... *18*
*I.3.1.1 Les capteurs intelligents* ..... *20*
*I.3.1.2 Les actionneurs intelligents* ..... *22*
*I.3.2 Instruments intelligents : architecture matérielle ou fonctionnelle* ..... *23*
*I.4 Structuration des fonctions du système d'automatisation* ..... *24*
*I.5 Les traitements du système d'automatisation* ..... *31*
*I.5.1 Les aides à la conduite* ..... *32*
*I.5.1.1 Les traitements de surveillance* ..... *33*
*I.5.1.2 Les traitements de pilotage* ..... *33*
*I.5.1.3 Elaboration d'informations complémentaires* ..... *34*
*I.5.2 Les aides à la maintenance* ..... *34*
*I.5.2.1 Les traitements de surveillance* ..... *35*
*I.5.2.2 Les traitements d'ordonnancement* ..... *35*
*I.5.2.3 Elaboration d'informations complémentaires* ..... *36*
*I.5.3 Les aides à la gestion* ..... *36*
*I.5.3.1 Les aides à la gestion de production* ..... *36*
*I.5.3.2 Les traitements de gestion technique* ..... *37*
*I.6 Conclusion* ..... *38*

## Introduction

Le développement de capteurs et actionneurs "intelligents" associé à celui des réseaux de terrain modifie la structure d'un système automatisé de production. Celui-ci, profitant des nouvelles possibilités de mémorisation, de traitements et d'interconnexion, évolue vers une nouvelle structure appelée système automatisé de production à intelligence distribuée. Cette nouvelle structure permet la réalisation de nouvelles fonctions jusque là non automatisées.

Dans ce chapitre, nous présenterons les différentes composantes du système automatisé de production. Puis nous procéderons à la définition et à la structuration de ces fonctions à partir de différents modèles. La dernière étape résulte des deux premières et concerne la description informelle des traitements nécessaires à la réalisation des fonctions du système automatisé de production.

## I.1 Le système automatisé de production

La littérature abonde de définitions relatives aux systèmes automatisés de production [Des 91], [Del 89], ... Malgré la diversité des approches qui les sous-tendent, elles s'accordent généralement sur le fait qu'un système automatisé de production est constitué essentiellement de trois composantes :

- des hommes appelés aussi opérateurs, ou utilisateurs, qui exploitent le système,
- le processus de production appelé aussi le processus physique, qui constitue le support physique du procédé. Le processus physique comprend un certain nombre d'équipements (cuves, tuyaux, réacteurs, machines, centres d'usinage, ... ) qui permettent la mise en oeuvre du procédé de fabrication. Ce dernier représente la définition du produit à obtenir grâce aux transformations effectuées par le processus.

- un système de contrôle commande appelé aussi système d'automatisation ou système informatique temps réel, qui permet le contrôle et la gestion des transformations réalisées dans le processus de production.

La figure 1.1 présente les composantes principales du système automatisé de production.

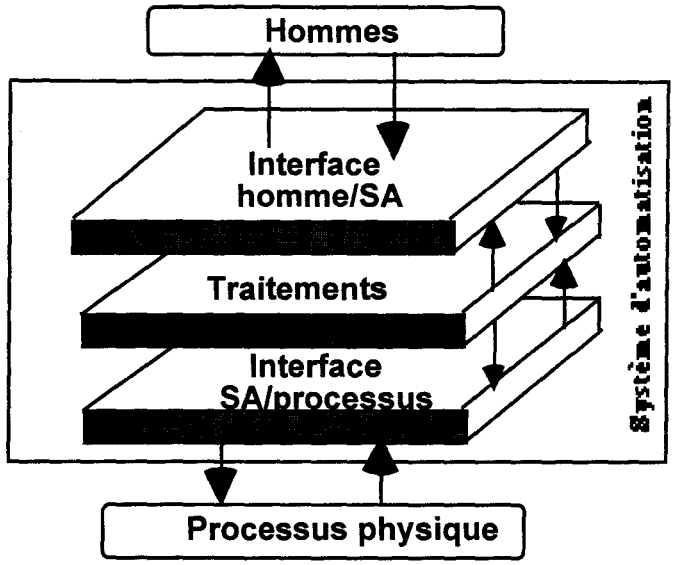



**Figure 1.1**

*Système automatisé de production*

SA : système d'automatisation

Le système d'automatisation communique avec les hommes à travers l'interface homme/SA, (des terminaux par exemple), ainsi qu'avec le processus à travers l'interface SA/processus supportée par les capteurs et les actionneurs.

La partie *centrale* du système d'automatisation qui regroupe *les traitements* réalise les services assurés par le système.

Le développement des instruments intelligents [Bay 95] et des réseaux de communications [Tho 90], conduit à une modification de l'architecture du système automatisé de production en offrant en particulier des possibilités de distribution de traitements et de communication donnant naissance aux Systèmes Automatisés de

Production à Intelligence Distribuée (voir figure 1.2). Parallèlement, les missions confiées aux systèmes d'automatisation, qui concernaient essentiellement la commande et la régulation, se sont élargies à des fonctions de surveillance et de supervision, d'aide à la gestion de production, à la gestion technique, à la maintenance, à la mise en place, à l'évolution, et au démantèlement du système.

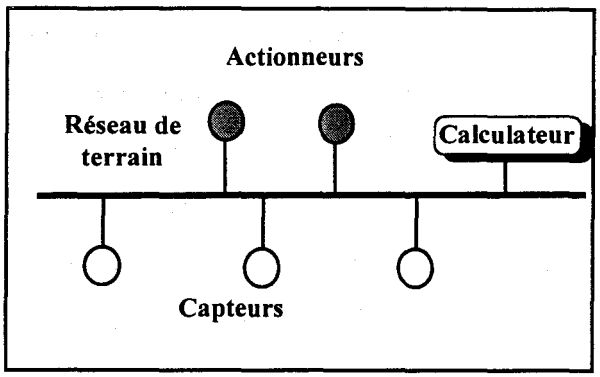



**Figure 1.2**

*Exemple d'architecture de système automatisé de production à intelligence distribuée*

Ces évolutions ont grandement accru la complexité de l'activité de conception. Le projet [Bay 95] a permis une formalisation de cette étape de conception, en définissant en particulier trois architectures du système automatisé de production.

## I.2 Architectures d'un système automatisé de production

**L'architecture fonctionnelle** d'un système d'automatisation est un modèle abstrait formalisé de la structure et du comportement externe des activités du système d'automatisation. C'est une description de la solution envisagée pour répondre aux exigences du cahier des charges. L'architecture fonctionnelle est un résultat de l'étape de spécification.

**L'architecture matérielle** du système automatisé de production à intelligence distribuée (SAPID) est constituée essentiellement [Sim 95] : d'un ensemble organisé de machines munies de systèmes d'exploitation, d'un ensemble de moyens de communication connectant ces machines, et de la définition des connexions des

machines sur les moyens de communication. Elle est le résultat d'activités de choix de matériels (machines, réseaux, exécutifs et protocoles), de dimensionnement de ces matériels et de définition des points de connexion. La caractérisation d'une architecture matérielle donnée est indispensable (coûts, capacités, performances, ...) pour vérifier ses propriétés et, ainsi, la valider.

**L'architecture opérationnelle** désignera le résultat d'une projection de l'architecture fonctionnelle sur une architecture matérielle.

L'architecture opérationnelle validée et optimisée est le résultat de l'étape de conception. Il s'agit de la "meilleure" architecture opérationnelle au sens d'un ou plusieurs critères. Elle est validée dans le sens où elle est conforme au cahier des charges, c'est à dire qu'elle respecte toutes les contraintes énoncées.

Nous détaillons dans ce qui suit l'architecture matérielle, et en particulier les instruments intelligents. L'architecture fonctionnelle fera l'objet du second chapitre.

## I.3 Les machines de l'architecture matérielle

Les machines de l'architecture matérielle du système d'automatisation sont vues comme des points de connexion. Elles jouent un rôle important puisqu'elles sont destinées à supporter l'architecture fonctionnelle du système d'automatisation, en plus des fonctions fournis par le constructeur. Chacune des machines (ou équipements autonomes) [Deb 93] est caractérisée principalement à partir des éléments suivants :

- ses composantes internes de communication, qui lui permettront de communiquer avec son environnement à travers les moyens de communication sur lesquels elle est connectée (annexe 2),
- son unité de traitement contenant un ou plusieurs processeurs, dont le rôle principal est d'exécuter un sous-ensemble de traitements implantés dans la machine. Ce sous-ensemble permet de définir le rôle fonctionnel de la machine,
- son système d'exploitation, qui soustrait le matériel aux regards de l'utilisateur, et qui permet de gérer et d'accorder l'usage des ressources, en évitant les

conflits d'accès entre les atomes de distribution. Il assure deux fonctions principales :

- la fourniture d'un ensemble de services en présentant aux utilisateurs une interface mieux adaptée à leurs besoins que celle de la machine physique; cette interface joue le rôle d'une machine virtuelle qui fournit un ensemble de fonctions pour la gestion et la communication d'information, et pour la réalisation des logiciels d'application, l'exécution des programmes qui composent ces derniers, l'aide à leur mise au point, le traitement de certaines défaillances, ...
- la gestion des ressources en assurant leur partage entre l'ensemble des demandeurs des services. Le système d'exploitation assure entre autres la gestion des ressources physiques, c'est à dire l'allocation des mémoires principales, des mémoires secondaires, des organes d'entrée-sortie, et d'autres services divers tels que la facturation des ressources, statistiques d'utilisation, mesure des performances, ...

Les instruments intelligents font partie des machines que l'on trouve dans l'architecture matérielle du système automatisé de production. Dans le paragraphe suivant, nous présentons les instruments intelligents : capteurs et actionneurs. Ces derniers seront décrits d'un point de vue matériel et fonctionnel.

### I.3.1 Les instruments intelligents

Pendant la dernière décennie, la révolution micro-électronique a donné un élan considérable à l'émergence de nouveaux types de machines qui intègrent en plus de leurs tâches habituelles, des mécanismes de traitement, de mémorisation et de communication de l'information. Ces mécanismes pouvant être implantés dans un microprocesseur, dans un micro-contrôleur, voire dans un simple circuit spécifique, permettent non seulement la miniaturisation, l'autonomie, l'optimisation, l'intégration de fonctions auparavant réalisées sous une forme analogique, mais aussi la création de fonctions entièrement nouvelles. Les capteurs et actionneurs intelligents sont l'un des résultats importants de cette mutation technologique.

Dans la boucle *mesure*, *décision*, *action* (MDA) [Sta 94b], la partie *décision* a été la première à profiter des nouvelles possibilités de traitement de l'information. En effet, les automates programmables, les régulateurs numériques, les systèmes numériques de contrôle commande ont complètement transformé la tâche des opérateurs.

La partie *mesure* a suivi, avec l'amélioration des performances des capteurs (grâce à des procédures de traitement de signal et de validation des données) et l'intégration de nouvelles fonctionnalités aux transmetteurs, destinées à faciliter l'exploitation de la partie *mesure* par la partie *décision.*

Enfin, la partie *action* bénéficie actuellement des techniques digitales, ce qui permet, comme pour les capteurs, à la fois d'améliorer les caractéristiques propres des actionneurs et de les doter de nouvelles fonctionnalités, plaçant ainsi l'intelligence et la prise de décision au niveau du "terrain", c'est à dire du processus piloté [Rob 93], [Bay 92a].

Les instruments intelligents (capteurs et actionneurs, ...) sont les machines les plus fréquentes dans l'architecture matérielle du système d'automatisation. Ils bénéficient de ressources permettant la réalisation de traitements locaux. Ces possibilités nouvelles vont leur permettre non seulement de remplir leurs fonctions initiales (la délivrance des mesures par les capteurs; la réalisation d'actions sur le processus physique par un actionneur), mais également d'améliorer leurs performances. Cette amélioration permet de répondre à de nouvelles exigences, grâce à l'implantation de traitements spécifiques.

Ainsi, les algorithmes de traitement de signal permettent de linéariser, de valider et par conséquent de fiabiliser la mesure fournie par un capteur, d'améliorer sa précision tout en étendant son domaine d'utilisation (grâce à des modèles de prise en compte des grandeurs d'influence); les algorithmes de commande avancée permettent d'améliorer la

précision de positionnement et le temps de réponse d'un actionneur, et d'accroître sa sûreté de fonctionnement, ...

La plupart des capteurs et actionneurs modernes, peuvent supporter des traitements, c'est une des raisons essentielles, qui leur a permis d'acquérir le statut de capteurs et d'actionneurs intelligents. Nous présentons dans ce qui suit l'architecture matérielle de ces instruments, ainsi qu'un ensemble de fonctions assurées par ces derniers.

### I.3.1.1 Les capteurs intelligents

Dans la boucle Mesure-Décision-Action, la décision est prise en fonction d'une image de l'état du processus physique fournie par les capteurs. Plus cette image sera proche de l'état réel du processus, mieux adaptées seront les décisions d'actions.

Pour améliorer la qualité de l'image que l'on a de l'état du processus, deux voies non exclusives sont a priori envisageables [Geh 94] : l'amélioration de la qualité du signal élaboré par chaque capteur, et l'augmentation du nombre de capteurs connectés. Le capteur intelligent est une solution permettant de prendre en compte les deux possibilités.

En effet, en associant au capteur classique une unité de traitement numérique, par exemple sous la forme d'un microprocesseur, on rend possible la réalisation de fonctions auparavant effectuées sous forme analogique (par exemple : le filtrage) mais on permet également l'intégration de fonctions entièrement nouvelles (par exemple : compensation des non linéarités, prise en compte des grandeurs d'influence). Ceci permet au capteur de remplir sa fonction initiale : délivrer des mesures des grandeurs relatives au processus physique, avec des performances accrues. Il contribue ainsi, en partie, à l'amélioration de la qualité de la représentation de l'état du système.

Ainsi, le capteur participe à l'évolution des systèmes de production vers plus de productivité et de sécurité grâce à l'amélioration de la qualité des signaux générés et la

prise en charge d'une partie des fonctions de décision. Dès lors, un capteur intelligent dispose de fonctionnalités nouvelles à savoir (voir figure 1.3) :

- la possibilité d'échanger des informations avec les autres composantes du système automatisé de production (les autres machines par exemple), ce qui lui permet d'acquérir un premier niveau d'intelligence,
- l'exécution et la gestion des atomes de distribution qui lui ont été affectés. Cette fonctionnalité permet d'obtenir, de la part du capteur intelligent, des services différents dans les différentes phases du cycle de vie du système d'automatisation,
- la surveillance locale. Ainsi certains atomes de distribution implantés dans le capteur intelligent ont à leur charge : la validation technologique (validation des conditions dans lesquelles les mesures ont été effectuées : nominales ou non) et la validation fonctionnelle (vérification de la plausibilité des mesures obtenues : comparaison à des seuils, vérification de la cohérence de plusieurs mesures, ...),
- l'élaboration et la gestion d'une partie de la base de données du système d'automatisation au niveau local; ...

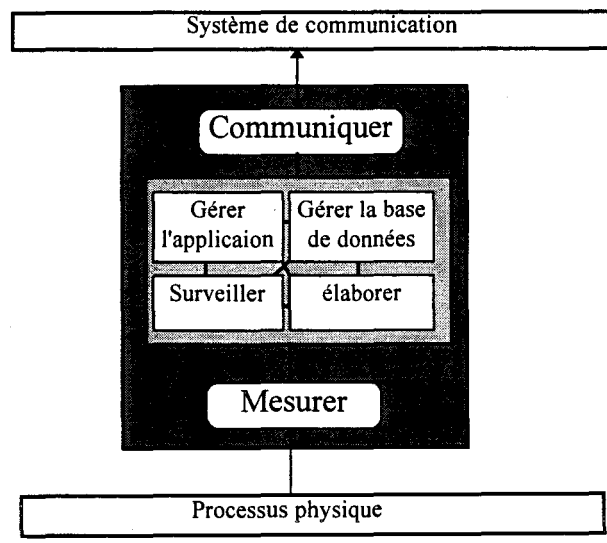



**Figure 1.3**

*Structure fonctionnelle d'un capteur intelligent*

### I.3.1.2 Les actionneurs intelligents

Un actionneur est conçu pour agir sur le processus physique, en modulant certaines des variables qui caractérisent son évolution et/ou son état. Ainsi, une pompe peut être utilisée pour moduler un débit de fluide, en transformant une puissance électrique (courant, tension) en une puissance hydraulique (débit, pression) ; un moteur modulera une vitesse, en transformant une puissance électrique (courant, tension) en puissance mécanique (vitesse de rotation, couple) ; un vérin modulera une position, en transformant une puissance pneumatique (débit, pression) en une puissance mécanique (vitesse linéaire, force).

Un actionneur comporte, en général, des capteurs utilisés pour obtenir des informations sur son état. Cette instrumentation permet de fournir des informations qui peuvent, entre autres, servir pour le processus physique.

Un actionneur intelligent est le résultat de l'association d'un actionneur conventionnel et d'une intelligence locale; celle-ci est caractérisée par :

- la capacité de l'actionneur intelligent à échanger des informations avec son environnement (autres instruments intelligents en particulier) à travers un système de communication (voir figure 1.4),

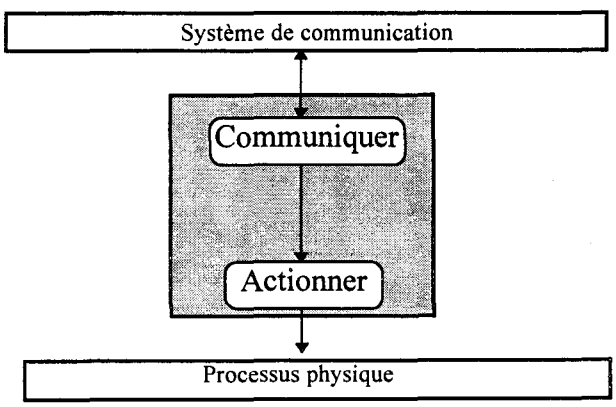



**Figure 1.4**

*Actionneur en boucle ouverte communicant*

- la capacité de l'actionneur intelligent à traiter l'information. En effet ses possibilités de traitement local de l'information, lui permettent de réaliser en

plus de sa vocation initiale (agir sur le processus physique), un sous-ensemble des traitements du système d'automatisation. En particulier, les traitements relatifs à la surveillance, l'élaboration de l'information et la gestion de la base de données globales du système, ... (voir figure 1.5)

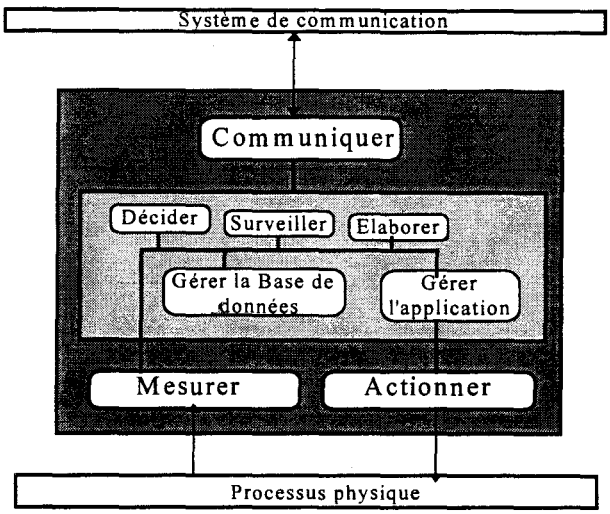



**Figure 1.5**

*Structure fonctionnelle d'un actionneur intelligent en boucle fermée*

## I.3.2 Instruments intelligents : architecture matérielle ou fonctionnelle

On distingue principalement deux types d'instruments intelligents :

- les instruments "ouverts", auxquels il est possible de confier une partie spécifique de l'application. Ceux-ci sont programmables (téléchargement des programmes par des moyens de communication spécialisés ou configuration préliminaire en mémoire morte). Ils constituent des éléments de l'architecture matérielle susceptibles de supporter des atomes de distribution de l'architecture fonctionnelle du système d'automatisation,
- les instruments "fermés", qui ne proposent que les services définis par leur constructeur; parmi leurs caractéristiques matérielles, seules leurs caractéristiques globales et celles relatives à leur interfaçage avec le reste du système (entrées/sorties, communication, alimentation, ...) présentent un intérêt en termes de caractérisation de l'architecture matérielle. Les atomes de

distribution qui vont résider sur ce type d'instruments seront supposés affectés avant le recours aux stratégies de distribution. Les instruments de ce type peuvent être considérés comme faisant partie des éléments de l'architecture opérationnelle du système.

## I.4 Structuration des fonctions du système d'automatisation

La définition des services offerts par le système d'automatisation passe par une structuration de ses fonctions.

De nombreux travaux s'attachent à une telle modélisation et différents modèles ont été étudiés pour mettre en évidence les fonctions d'un système d'automatisation [Bay 92a], [Del 89], [Ver 89], [Sim 92]. Selon l'objectif de ces modèles, certains se limitent à la description des activités du système pendant sa phase d'exploitation, d'autres décrivent les activités du système à partir de sa phase de conception jusqu'à sa phase de démantèlement.

Dans ce premier chapitre, nous détaillons les fonctions d'un système d'automatisation ainsi que leurs interactions, elles constituent son architecture fonctionnelle.

Le modèle 3 axes fait apparaître plusieurs catégories de fonctions telles que : la conduite, la sécurité, la maintenance, le suivi.

- la conduite est destinée à contrôler le comportement du processus de production pour atteindre les objectifs exprimés en termes de qualité et de productivité,
- la sécurité (ou sûreté) est destinée à assurer la non-occurence de défaillances qui pourraient mettre en danger les hommes, l'environnement, le processus physique, les produits,
- la maintenance est destinée à assurer la disponibilité du processus de fabrication,
- le suivi est destiné à recueillir et à synthétiser les informations relatives à l'état du processus physique et des produits.

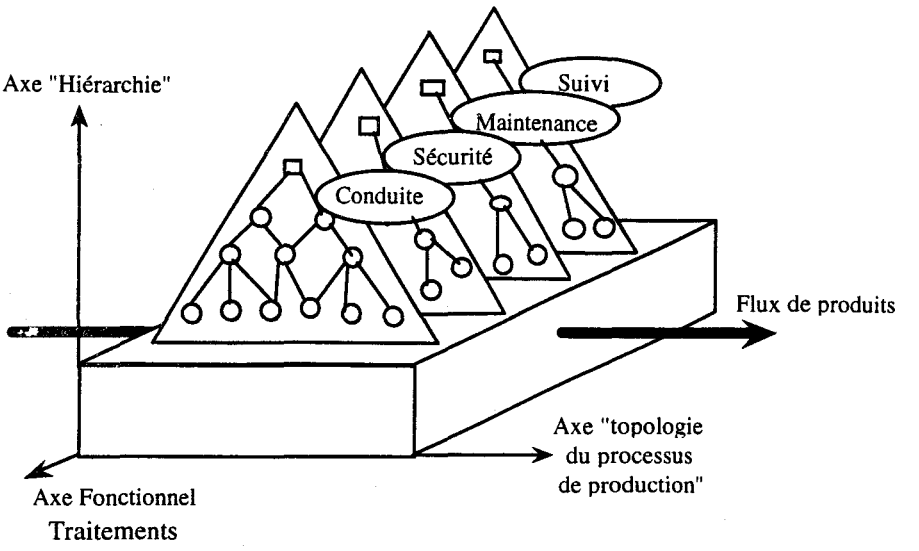



**Figure 1.6**

*Les différents niveaux du système d'automatisation (d'après [Del 89] )*

Le premier axe est l'axe hiérarchique selon lequel on peut décomposer une fonction en un certain nombre de niveaux hiérarchisés (le nombre de niveaux dépend de l'application) ; à un niveau donné, une entité, qui offre des services accessibles du niveau supérieur, s'appuie sur les niveaux inférieurs pour effectuer les requêtes qui lui viennent du niveau supérieur.

Le second axe représente le processus physique, soit équipement par équipement, comme c'est souvent le cas des systèmes manufacturiers, soit par ensembles de matériels ou d'équipements qui ont une signification particulièrement homogène comme dans les processus continus (par exemple : transport de fluide, gestion d'énergie, ...).

Le troisième axe représente les fonctions mises en oeuvre dans le système automatisé de production.

Une extension du modèle 3 axes à une architecture CIM (Computer Integrated Manufacturing) faisant apparaître les différents niveaux de communication entre les fonctions du système a été proposée dans [Ver 89]. La figure 1.7 présente cette extension.

fonctions du système a été proposée dans [Ver 89]. La figure 1.7 présente cette extension.

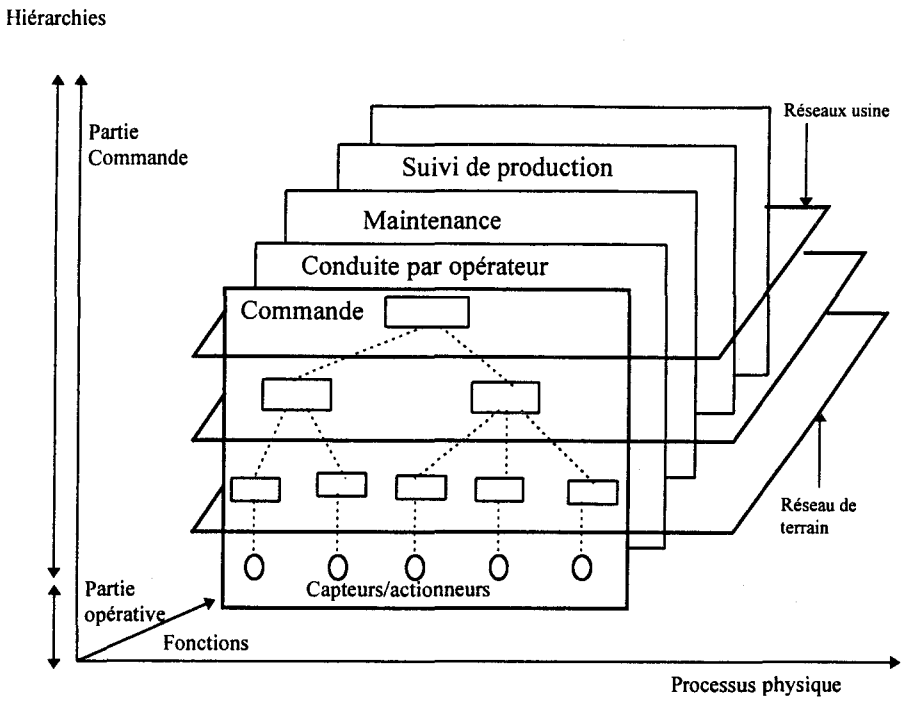



**Figure 1.7**

*Modèle 3 axes [Ver 89]*

Le modèle 3 axes précédent repose sur une partition entre partie opérative et partie commande qui rend difficile [Dum 96] :

- la différenciation entre le fonctionnel et l'organique, ainsi la partition entre le monde physique et le monde de l'information n'apparaît pas,
- la mise en valeur de l'unicité du système automatisé de production.

Pour résoudre ces problèmes, J. J. Duméry et al. [Dum 96] propose un cadre de modélisation (voir figure 1.8) reposant sur deux principes :

- un principe de matérialité énonçant que "la réalité est unique et porte toutes ses vues". La réalité correspond à l'implémentation sous ses formes matérielle et logicielle. Les aspects sont des interprétations fonctionnelles de cette implémentation.
- un principe d'abstraction/encapsulation : une hiérarchie d'abstraction propre à chaque aspect permet à un niveau donné, de ne laisser apparaître du niveau

inférieur que les éléments nécessaires à la compréhension de ce niveau (vue réduite).

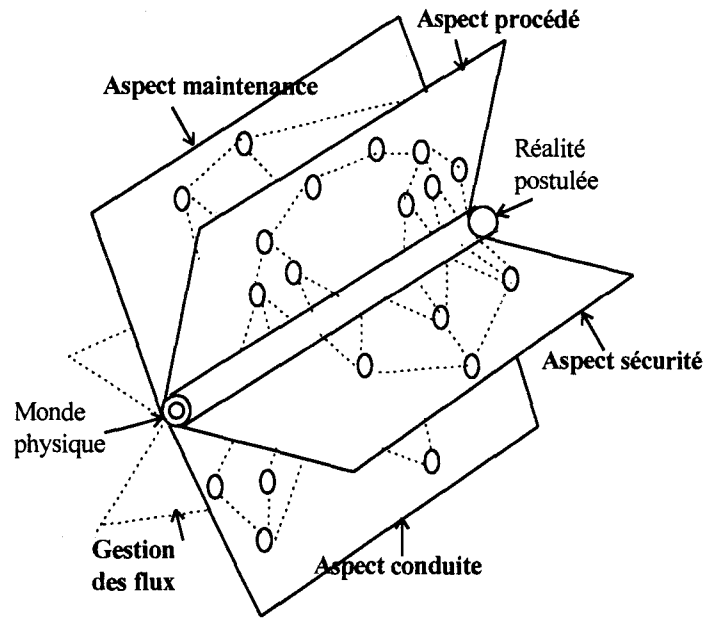



**Figure 1.8**

*Cadre de modélisation [Dum 96]*

Cette modélisation met en évidence que toute les fonctions du système s'appliquent à un même processus physique.

Dans les modèles présentés ci-dessus, les fonctions du système d'automatisation sont analysées dans le contexte de sa phase de vie la plus importante à savoir la phase d'exploitation. Cependant la vie du système d'automatisation ne se limite pas à cette phase d'utilisation ou d'exploitation. En effet, le système d'automatisation passe par d'autres phases de vie. L'ensemble de ces étapes (de la conception au démantèlement) constituent son cycle de vie, et décrivent son évolution dans le temps.

Les activités du cycle de vie, ainsi que les flux circulant entre ces différentes activités sont décrits dans [Bay 92b] et seront exposés dans ce paragraphe. La figure 1.9 illustre le cycle de vie du système automatisé de production. On y distingue les activités suivantes :

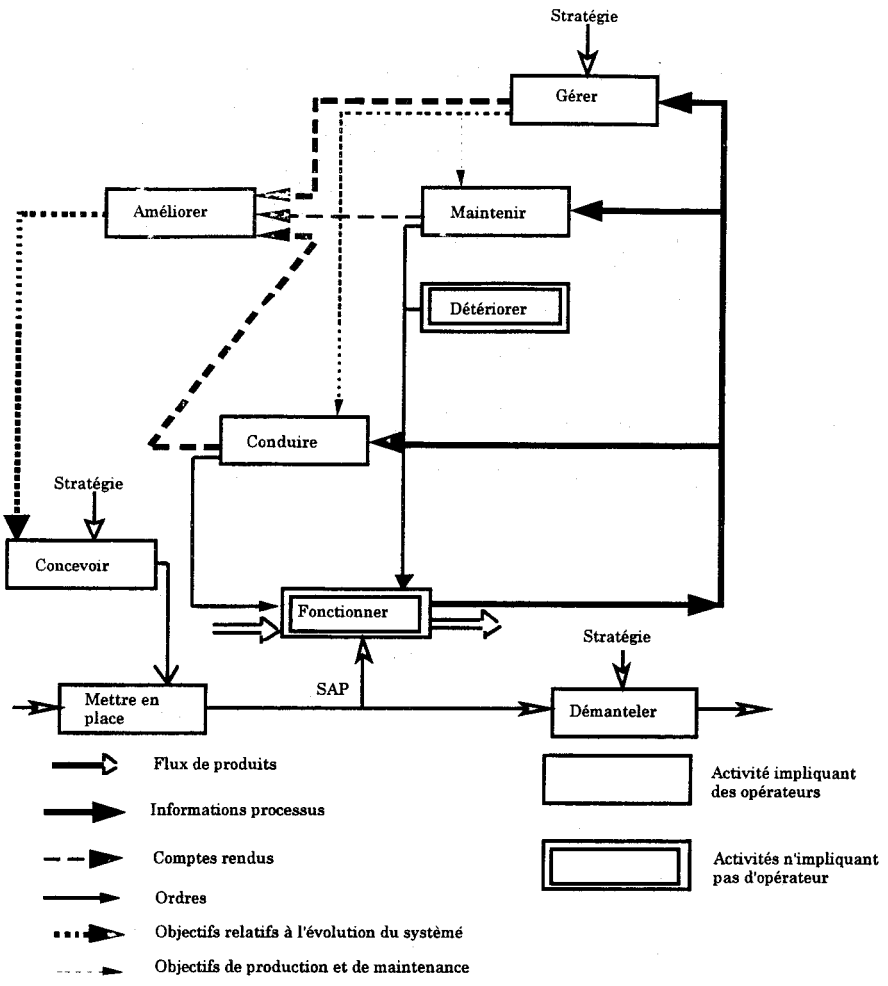



**Figure 1.9**

*Activités et cycle de vie [Bay 92b]*

- **Activité de conception :** au cours de cette phase, sont conçus les constituants du processus physique, ainsi que ceux du système d'automatisation. Cette conception couvre les aspects matériel, logiciel, la définition des procédures d'exploitation, de gestion, ... Il peut s'agir de la conception initiale du système aussi bien que de la conception de certaines améliorations d'un système déjà existant. Dans ce sens, l'activité de conception est activée par des contraintes de type stratégie (que faire, dans quels délais, à quel coût, avec quelles performances, ...) ainsi que par la sortie de l'activité d'amélioration, qui analyse les performances du système existant et décide de son évolution. La sortie de l'activité de conception déclenche l'activité de mise en place.

- **Activité de mise en place :** cette activité couvre la réalisation, l'assemblage, l'interconnexion des différents équipements. A travers un certain nombre de procédures de mise en service, elle aboutit à la fourniture du support des activités de production. Son entrée est constituée par les différents équipements, pièces détachées, fournitures dont la mise en oeuvre permettra la réalisation du système automatisé de production ou de ses évolutions. Celui-ci en constitue la sortie, qui est le support des activités de production proprement dites (activité de fonctionnement).

  L'activité de mise en place est pilotée par les résultats de l'activité de conception : plans, nomenclatures, notices de montage, procédures d'essai, de mise en service, ...

- **Activité de fonctionnement :** cette activité constitue l'objectif pour lequel le système a été conçu. Son support est le système automatisé de production, qui traite un flux de produits (produits bruts, matières premières en entrée, produits finis en sortie). Cette activité consomme des informations provenant de l'activité de conduite (pilotage du processus de production) et produit des informations destinées aux activités de conduite ("fermeture" de la boucle de pilotage), de maintenance, de gestion. Elle est influencée dans son exécution, par les sorties des activités de détérioration et de maintenance.
- **Activité de conduite :** cette activité génère les commandes qui pilotent l'activité de fonctionnement. Elle s'exécute sous le contrôle de l'activité de gestion, qui lui fournit ses objectifs ainsi qu'un certain nombre de contraintes (procédures d'exploitation par exemple). Elle est le fait des opérateurs de conduite, qui produisent par ailleurs les comptes rendus de conduite utilisés, entre autres, par l'activité d'amélioration.
- **Activité de détérioration** : la dégradation du système, le plus souvent liée à son utilisation (usure, fatigue, ...), peut aussi se produire de façon tout à fait imprévisible (accident, panne, ...). L'existence de l'activité de dégradation impose la mise en place d'une activité de maintenance. La sortie de l'activité de détérioration est une contrainte pour l'activité Fonctionner. En effet, le

processus physique et son système d'automatisation subissent les défauts et pannes en résultant et voient ainsi altérées leurs fonctions.

- **Activité de maintenance** : dans cette activité sont reconstituées, restaurées, les capacités du système de production. Les conditions de passage de l'activité exploitation à l'activité maintenance sont réalisées en boucle ouverte, indépendamment de l'état réel du processus (maintenance systématique) ou en boucle fermée, lorsque l'état du processus le justifie (maintenance préventive conditionnelle, ou maintenance corrective). C'est pourquoi l'activité de maintenance traite, en entrée, des données en provenance de l'activité de fonctionnement. L'activité de maintenance comme celle de détérioration influencent l'activité de fonctionnement. Elle est, comme l'activité de conduite, sous le contrôle de l'activité de gestion, qui définit les stratégies et les procédures de maintenance et gère les inévitables conflits entre ces deux activités. Les opérateurs de maintenance produisent, comme leurs collègues de la conduite, des comptes-rendus utilisés par les activités d'amélioration et de gestion.
- **Activité de gestion** : cette activité est de type organisationnel : elle fixe les objectifs et les contraintes relatives à la gestion des produits et à celle du processus.
- **Activité d'amélioration** : cette activité conduit à faire évoluer le processus ou ses procédures d'exploitation. L'amélioration peut être le résultat de modifications matérielles (remplacement d'un équipement par un autre, plus performant), logicielles (implantation d'une version plus élaborée) ou d'une modification dans les procédures de gestion (simplification d'un circuit d'information, ...).

  L'activité d'amélioration traite, en entrée, les comptes rendus issus des différents services qui concourent à l'exploitation et au maintien en état du processus de production et de son système d'automatisation. Elle fournit, en sortie, des contraintes à l'activité de conception (cahiers des charges).
- **Activité de démantèlement** : cette activité constitue le pendant de l'activité de mise en place. Elle conduit à la disparition du système de production, par désassemblage de ses constituants.

L'objectif de cette description dans [Bay 92a] était d'avoir une vue complète des activités effectuées tout au long du cycle de vie du système d'automatisation, de manière à mettre en évidence les traitements, nécessaires à la réalisation de ces aides, qui pouvaient être implantés dans les instruments intelligents.

## I.5 Les traitements du système d'automatisation

L'objectif commun des modèles présentés est de faire apparaître les différentes fonctions confiées au système d'automatisation. Celui-ci doit contrôler et commander le processus par des traitements adéquats, en fonction des tâches qui lui ont été confiées. Il ne peut accéder au processus que via des capteurs et des actionneurs qui constituent l'interface système d'automatisation/processus.

L'analyse des différentes activités présentées ci-dessus permet de déterminer l'ensemble des traitements à implanter. Ces traitements matériels et/ou logiciels seront supportés par un ensemble d'équipements de contrôle commande, et assistés par des opérateurs humains.

Ce sont ces traitements, qui constituent l'architecture fonctionnelle du système automatisé de production, qui seront distribués sur les différents équipements. Nous décrirons ci-dessous ces traitements, en nous limitant à ceux relatifs à la phase d'exploitation du système d'automatisation. Dans celle-ci on distingue trois grandes classes de traitements : les traitements d'aide à la conduite, d'aide à la maintenance et d'aide à la gestion.

### I.5.1 Les aides à la conduite

La fonction de conduite doit être en mesure [Geh 94] :

- d'identifier à tout instant le mode de marche du système qui résulte de la combinaison d'un mode d'utilisation souhaité par l'opérateur ou l'automatisme (configuration, automatique, manuel, ...) et d'un état jugé

normal, dégradé ou en panne selon les disponibilités des ressources matérielles,

- d'évaluer, la possibilité d'atteindre, dans le mode de marche courant, les objectifs de production fixés sous les contraintes de sécurité imposées pour les hommes, l'environnement et les ressources matérielles,
- d'évaluer, le cas échéant, la nécessité ou l'opportunité de changer de mode d'utilisation afin d'atteindre les objectifs souhaités sous les contraintes imposées (par exemple en utilisant un circuit ou une alimentation de secours),
- de définir la commande et la coordination des différentes ressources matérielles, ceci aussi bien pour le mode courant que lors de changement de mode,
- de demander la modification des objectifs (par exemple diminution de la production) et/ou des niveaux de contrainte (par exemple validation incomplète d'une donnée car la redondance physique est non satisfaite) dans le cas où aucun mode d'utilisation automatisé ne permettrait de les respecter, compte tenu de l'état du système.

De cette analyse, il résulte que la liste des traitements qui permettent d'assurer la conduite de l'installation pour produire en quantité et qualité, est longue. Toutefois le travail de l'opérateur se trouve facilité par les aides à la conduite implantables au niveau du système d'automatisation. Ces aides s'orientent selon trois axes principaux : la surveillance, le pilotage et l'élaboration de l'information.

### I.5.1.1 Les traitements de surveillance

Les traitements de surveillance ont initialement pour fonction de détecter les défauts, de les localiser dans la mesure du possible et éventuellement d'en rechercher les causes. Ils doivent être construits de façon à détecter aussi bien les défauts au niveau des ressources matérielles du système d'automatisation (capteurs, actionneurs, régulateurs, automates, ...) que du processus physique lui-même.

Les traitements de surveillance reposent sur différentes techniques de validation : vérification de bilans, méthode de l'espace de parité, techniques de reconnaissances des formes [Pat 89], [Dub 90], [Rag 90], ... Ils permettent une validation des informations mises à la disposition des opérateurs et des automatismes. Ceux-ci bénéficient alors de l'image de l'état réel du système automatisé et de la disponibilité opérationnelle des objets le constituant. Il s'ensuit que les décisions et les actions sont mieux adaptées.

### I.5.1.2 Les traitements de pilotage

La réalisation des objectifs de production qui se caractérisent par une transformation des matières premières et de l'énergie en des produits finis, passe par le pilotage des actionneurs. Cette tâche de pilotage qui peut parfois apparaître répétitive et fastidieuse est souvent automatisable. C'est le cas par exemple, avec la mise en place d'algorithmes de régulation (élaboration d'une action en fonction de la différence entre une valeur de consigne et une valeur mesurée) et des algorithmes de séquencement (cas des automates où une action fait suite à une autre lorsque les conditions de passage de l'une à l'autre sont satisfaites) [Bha 90], [Mar 87]. Un niveau d'aide plus élevé peut être atteint par l'implémentation de modules d'aide à la décision capable de proposer, d'argumenter, d'évaluer des actions possibles de l'opérateur [Ara 89], [Far 85].

### I.5.1.3 Elaboration d'informations complémentaires

L'ensemble des capteurs fournit des données aux automatismes, qui peuvent être transmises aux opérateurs de conduite. En fait, compte tenu du nombre de données disponibles sur un système automatisé de production, il est intéressant d'élaborer des informations synthétiques qui seront mises à la disposition de l'opérateur et lui permettront d'avoir une vue globale de certaines parties du système (par exemple courbe de tendances, histogrammes, ...). Les traitements d'élaboration de l'information ont la charge de cette fonction.

### I.5.2 Les aides à la maintenance

Il existe plusieurs formes de maintenance :

- la maintenance corrective qui consiste à réparer et à remettre en conformité l'installation après avoir constaté la défaillance et/ou ses conséquences,
- la maintenance préventive systématique qui est effectuée selon un échéancier, établi en fonction des données constructeurs et/ou de l'expérience, dans l'intention de réduire la probabilité de défaillance d'un bien ou un service rendu. Elle se concrétise par exemple, par le remplacement d'une pièce lorsqu'un nombre prédéterminé d'heures d'exploitation est atteint ou lorsqu'un nombre donné d'événements sont survenus. Elle nécessite donc de connaître l'activité du système (par exemple un joint est garanti jusqu'à mille manoeuvres, une vanne nécessite d'être graissée tous les trois mois),
- la maintenance préventive conditionnelle ou maintenance prédictive qui est une maintenance subordonnée à un type d'événement prédéterminé. Elle repose sur l'analyse du comportement du système afin d'en prévoir les dégradations progressives.

Une opération de maintenance, qu'elle soit préventive ou corrective nécessite des ressources variées (outils, temps, opérateur). De plus, elle doit être exécutée selon un mode opératoire précis. Si les ressources et les modes opératoires sont parfaitement définis pour les opérations de maintenance préventive systématique, ils ne le sont pour les autres opérations de maintenance que si un diagnostic précis du défaut est réalisé. De cette analyse découle que trois classes de traitements peuvent être implantés : la surveillance, l'ordonnancement, l'élaboration d'informations.

#### I.5.2.1 Les traitements de surveillance

Les algorithmes de surveillance ont pour mission, comme nous l'avons déjà mentionné dans le paragraphe relatif aux aides à la conduite, de détecter, localiser et

diagnostiquer les défauts. Ils contribuent à l'élaboration de la liste des opérations de maintenance à réaliser. Cette liste est utilisée par l'ordonnanceur de maintenance. Elle inclut la date, la nature du défaut avec une localisation et un diagnostic plus au moins précis selon les possibilités. Dans le cas de la maintenance systématique, cette activité peut se résumer à la comparaison des compteurs à des seuils fixés.

### I.5.2.2 Les traitements d'ordonnancement

L'ordonnancement assure la planification temporelle des opérations de maintenance de façon à optimiser l'utilisation des ressources et à rendre le système opérationnel le plus tôt possible. Par conséquent, il constitue une aide précieuse pour l'opérateur de maintenance en l'assistant dans sa prise de décision concernant la planification, la préparation et le déroulement de l'exécution des travaux de remise en conformité de l'installation.

### I.5.2.3 Elaboration d'informations complémentaires

Les deux classes de traitements (surveillance et ordonnancement) sont accompagnées d'une élaboration d'informations. En effet, la mise en place de compteurs d'événements ou la totalisation du nombre d'heures de fonctionnement sont des capteurs virtuels que l'on pourra implanter. De la même façon, les comptes-rendus de maintenance (en particulier, la durée des réparations) et les historiques des pannes permettent d'évaluer certains critères de fonctionnement (moyenne des temps de bon fonctionnement, moyenne des temps techniques de réparation, ...), ces critères pouvant bien sûr aider à la planification des opérations de maintenance.

### I.5.3 Les aides à la gestion

Les activités de gestion se subdivisent en deux classes : gestion de production et gestion technique.

### I.5.3.1 Les aides à la gestion de production

La fonction de gestion de production fixe les objectifs de production. Elle détermine entre autres les quantités à produire en fonction des besoins du marché, des capacités d'investissement, ...

Elle détermine également les méthodes, les outils, et les moyens humains nécessaires à une production fructueuse.

Le système d'automatisation participe à la gestion de production en apportant des aides de surveillance, d'ordonnancement, et d'élaboration d'informations.

#### Les traitements de surveillance

Les algorithmes de surveillance ont essentiellement pour fonction d'élaborer l'image de l'état réel du système automatisé et de la disponibilité opérationnelle des objets qui le constituent. Ils permettent ainsi d'organiser au mieux la production en connaissant les véritables capacités de l'installation.

#### Les traitements d'ordonnancement

Les traitements d'ordonnancement planifient l'utilisation des ressources matérielles en fonction de leur disponibilité opérationnelle, des besoins (quantité à produire) et des délais.

#### Elaboration d'informations complémentaires

L'élaboration des informations autres que celles produites par les capteurs permet d'enrichir l'information relative au système. Ainsi, connaissant certaines données initiales et sachant à tout instant quelles sont les quantités de produit générées et consommées, il est par exemple possible d'informer l'utilisateur, sur

l'état des stocks, sur les commandes d'approvisionnement à renouveler, sur les prix de revient estimés, ...

### I.5.3.2 Les traitements de gestion technique

La fonction de gestion technique vise à optimiser l'utilisation des matériels support de production afin de produire à moindre coût pour des niveaux de qualité du produit et de sûreté égaux. En particulier, elle vise à réduire les durées d'indisponibilité des équipements en gérant les stocks des pièces et outils qui leur sont nécessaires. Les aides relatives à la gestion technique apportées par le système d'automatisation s'expriment en termes de traitements de surveillance et d'élaboration de l'information.

#### Les traitements de surveillance

La détection, la localisation, le diagnostic des défauts permettent jour après jour de mieux connaître le matériel, en particulier de prendre connaissance de ses faiblesses, de ses dérives. L'exploitation des informations issues des algorithmes de surveillance peut conduire à des corrections sur l'utilisation du matériel.

#### Elaboration d'informations complémentaires

La gestion technique utilise également des informations non directement produites par les capteurs. On peut citer entre autres : la consommation électrique d'un équipement, l'état des stocks de pièces de rechange, ...

Ces traitements élaborent des informations utiles à la gestion technique. Ces informations ne sont pas produites directement par les capteurs.

## I.6 Conclusion

Dans ce chapitre, nous avons présenté les différentes composantes du système automatisé de production à savoir, les utilisateurs du système, le processus physique et

le système d'automatisation. Celui-ci constitue le "centre nerveux" du système, et assure un ensemble de fonctions. Différents modèles permettant une structuration de ces fonctions ont été présentés. Trois grandes classes d'aides relatives à la phase d'exploitation du système, ont été identifiées : il s'agit de celles destinées à la conduite, celles destinées à la maintenance et celles destinées à la gestion. Les traitements nécessaires à leur réalisation ont été présentés et détaillés d'une manière informelle. Ils constituent l'architecture fonctionnelle du système d'automatisation. L'objectif du chapitre suivant est de décrire plus précisément cette architecture en formalisant les éléments qui la composent, grâce à une modélisation par atomes de distribution.

On doit utiliser les modèles, mais
ne pas y croire

**Henri Teil**

# Chapitre II

# Modélisation du système d'automatisation par atomes de distribution

*Introduction* ..... 41
*II.1 Structuration du système d'automatisation* ..... 41
*II.2 Modélisation de l'architecture fonctionnelle du système d'automatisation* ..... 44
*II.2.1 Les modules* ..... 45
*II.2.2 Les capsules* ..... 47
*II.2.3 Les atomes de distribution* ..... 49
*II.2.3.1 Caractérisation des atomes de distribution* ..... 52
*II.2.3.1.1 L'activité de traitement d'un atome de distribution* ..... 52
*II.2.3.1.2 L'activité de gestion d'un atome de distribution* ..... 53
*II.2.4 Caractérisation des variables* ..... 54
*II.2.4.1 Les variables produites* ..... 54
*II.2.4.2 Les variables consommées* ..... 54
*II.2.5 Classification des atomes de distribution* ..... 55
*II.2.6 Redondance des atomes de distribution* ..... 56
*II.2.7 Gestion des états d'un atome de distribution* ..... 58
*II.2.7.1 Vue externe des états d'un atome de distribution* ..... 59
*II.2.7.2 Vue interne des états d'un atome de distribution* ..... 60
*II.2.7.2.1 Suspension* ..... 61
*II.2.7.2.2 Répétition* ..... 62
*II.2.7.2.3 Contre ordre* ..... 63
*II.2.8 Relations entre les atomes de distribution* ..... 64
*II.2.8.1 Relations d'échanges d'information* ..... 66
*II.2.8.2 Mesure de la relation d'échanges d'information* ..... 67
*II.3 Ordonnancement des atomes de distribution* ..... 68
*II.4 Implantation des atomes* ..... 70
*II.5 Conclusion* ..... 72

## Introduction

Le premier chapitre présentait une description informelle des traitements du système automatisé de production. L'objectif de ce chapitre est la formalisation de cette description. Cette tâche sera réalisée à l'aide d'un modèle que nous avons voulu général, appelé modèle par atomes de distribution. Ces derniers constituent une composante essentielle de l'architecture fonctionnelle du système d'automatisation.

Dans un premier temps, nous présenterons une structuration du système d'automatisation qui constitue une formalisation préliminaire du partitionnement du système. Puis nous présentons notre approche de modélisation du système d'automatisation par atomes de distribution.

## II.1 Structuration du système d'automatisation

Le système d'automatisation à travers ses deux interfaces spécifiques rend des services aux opérateurs et au processus physique. L'ensemble des services rendus par le système d'automatisation repose sur un ensemble de traitements, implantés sous forme matérielle (carte de filtrage analogique, logique combinatoire ou séquentielle câblée) ou sous forme logicielle. Nous avons vu au chapitre précédent l'ensemble des traitements à mettre en place pour réaliser les fonctions d'aide à la conduite, à la maintenance et à la gestion. On note A cet ensemble de traitements.

Quelle que soit la forme sous laquelle se présenteront les traitements, on note $C_e(A)$ l'ensemble des informations traversant les deux interfaces (hommes/SA et SA/processus) de l'extérieur vers A, et $P_e(A)$ l'ensemble des informations traversant les deux interfaces de A vers l'extérieur.

$C_e(A)$ est l'ensemble des données consommées par l'application A, $P_e(A)$ est l'ensemble des données produites (figure 2.1). Les signaux représentent les informations analogiques, tandis que les codes désignent les informations numériques.

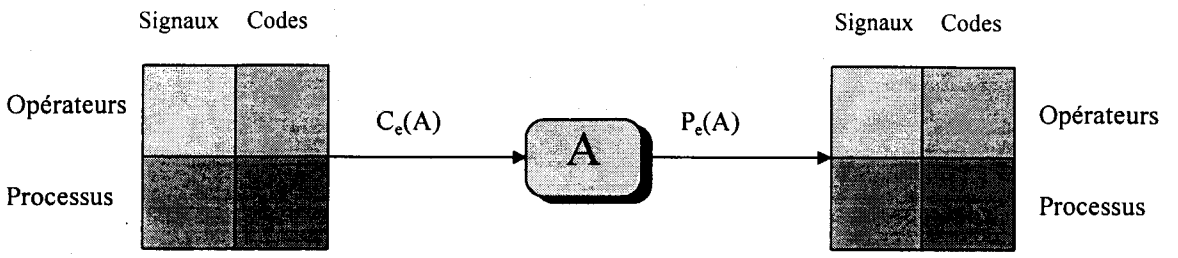



**Figure 2.1**

*Application, données consommées et produites*

La distribution du système d'automatisation sur un certain nombre de sites, conduit à affecter à chacun d'entre eux une partie des traitements de A, ainsi qu'une partie des interfaces opérateurs et processus.

Soit $A_I$ une partie de A, dont l'exécution a été confiée à un site donné. L'environnement de $A_I$ est constitué, d'une part, de l'environnement de A (opérateurs et processus) et d'autre part, des traitements de son complément dans A, noté $CA_I$ (une ou plusieurs portions d'application). De même que les opérateurs et le processus, les traitements de $A_I$ produisent (consomment) des informations, dont certaines sont consommées (produites) par $CA_I$. C'est le rôle des moyens de communication inter-sites de mettre à la disposition de $A_I$, parmi les données produites par $CA_I$ celles qu'il consomme et réciproquement pour $CA_I$. La figure 2.2 illustre cette situation.

Notons que l'intersection des ensembles de données consommées par $A_I$ et par son complément peut être non vide :

$C_e(A_I) \cap C_e(CA_I) \neq \varnothing$ si une partie des données consommées par les traitements de $A_I$, est consommée également par les traitements de $CA_I$. Sachant que $C_e(CA_I) = C_e(A \backslash A_I)$.

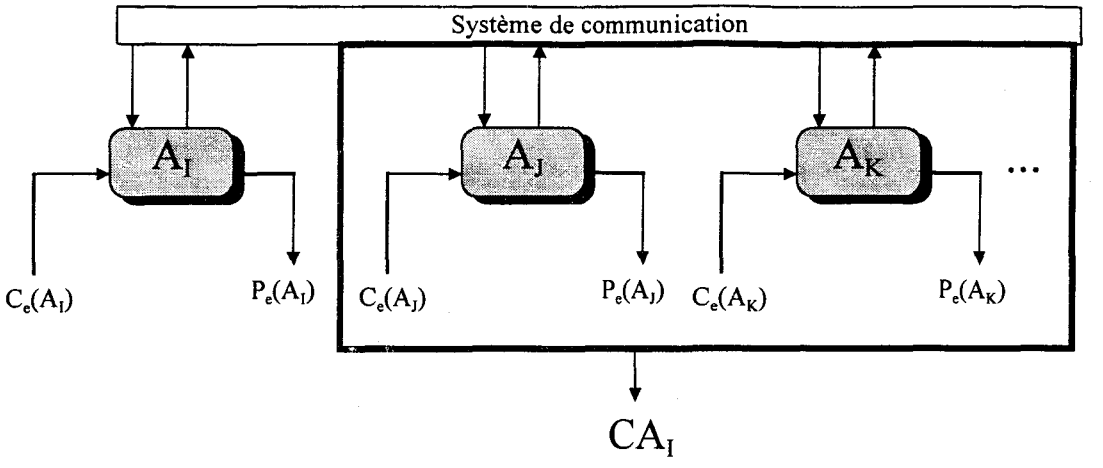



**Figure 2.2**

*Application distribuée*

Par contre, nous considérons qu'une donnée n'est produite que par un seul producteur, l'intersection des ensembles de données produites par $A_I$ et par son complément est donc vide :

$$P_e(A_I) \cap P_e(CA_I)=\varnothing$$

Sachant que $P_e(CA_I)= P_e(A\backslash CA_J)$

Dans la suite, les indices e s'appliquent aux données échangées avec l'environnement (extérieur à A), les indices i s'appliquent aux données échangées avec d'autres parties de A (intérieur de A).

Le système caractérisé par l'ensemble des traitements A est distribué sur un ensemble de sites J et l'on a :

$$\bigcup_{I \in J} A_I = A$$

et $A_I \neq \varnothing$ pour tout $I \in J$

L'ensemble des $A_I$ constitue alors une partition de A,

- les parties de A sont non vides.

- les intersections entre les parties de A (deux à deux) sont vides, puisqu'un traitement ne peut pas être implanté sur deux sites différents,
- l'union des parties de A est égale à A.

et les relations suivantes sont également vérifiées :

$$\bigcup_{I \in J} C_e(A_I) = C_e(A)$$

$$\bigcup_{I \in J} P_e(A_I) = P_e(A)$$

On voit que, pour définir un problème de distribution, pour développer des méthodes de génération et d'évaluation des solutions de ce problème, on devra considérer l'ensemble A, l'ensemble de ses parties, de ses partitions, ... Il convient alors de définir et de caractériser précisément les éléments de cet ensemble et les structures qui les lient.

## II.2 Modélisation de l'architecture fonctionnelle du système d'automatisation

Chaque fonction du système est caractérisée par le service qu'elle rend et la structure représente l'ensemble des fonctions et les liens de données entre fonctions. Le comportement de l'ensemble des fonctions est décrit par l'ensemble des liens de contrôle.

Dans la plupart des travaux relatifs aux systèmes distribués (pour l'informatique, ou l'automatisation), la définition et la caractérisation des traitements élémentaires, appelés suivant les domaines d'applications : processus, capsules, modules, programmes, ... ou dans le cadre de ce travail "atomes de distribution", ont été présentés et caractérisés d'une manière sommaire. Les travaux qui se sont intéressés à la modélisation et l'utilisation des traitements élémentaires sont fortement orientés "systèmes informatiques" [Sto 78], [Ram 89], [Mun 90] et présentent des limites pour une utilisation dans un autre contexte. Le domaine qui nous concerne plus précisément est celui des Systèmes Automatisés de Production à Intelligence Distribuée (SAPID),

issu du développement des capteurs et actionneurs intelligents, et des réseaux de terrain. Cette instrumentation, avec ses capacités de traitement et de communication, offre de nouvelles possibilités au système automatisé de production dans la mesure où elle implique dès l'origine une certaine distribution des traitements.

La littérature qui traite de la modélisation des entités élémentaires d'une architecture fonctionnelle n'est pas abondante. Elle se limite généralement à la présentation de l'aspect "relationnel" entre ces entités, ou entre ces entités et les équipements qui les supportent : en présentant des coûts de communication [Sto 78], des coûts d'exécution [She 85], [Tal 91]. D'autres travaux relatifs aux domaines de l'ordonnancement s'attachent plutôt au caractère temporel [Car 94], [Ram 89].

Nous avons voulu entamer la modélisation des entités élémentaires de l'architecture fonctionnelle avec une spécification sommaire des caractéristiques de ses entités. Nous voulions que notre atome soit modulaire, réutilisable, autonome, ...

Deux modèles nous ont paru très proche de notre spécification, et qui s'incluent dans notre domaine de travail sont : les modèles de module [Tho 80] et de capsule [Asl 92]. Nous présentons succinctement ces modèles, leurs propriétés et leurs limites par rapport à notre domaine de travail. Puis nous détaillerons le modèle que nous avons développé basé sur la notion d'atome de distribution. Enfin nous décrirons les relations existantes entre atomes, les moyens de mesure de ces relations et leur classification. Ces informations sont très utiles pour la simplification et la résolution du problème de répartition.

### II.2.1 Les modules

Dans le système SYGARE [Tho 80], une application est vue comme un ensemble de tâches parallèles coopérant pour la réalisation d'un certain objectif. Les tâches sont décomposées en entités élémentaires appelées modules.

Un module exprime un algorithme et est perçu comme un automate séquentiel élaborant des sorties à partir d'entrées définies. Il constitue l'unité de répartition.

La communication entre les modules se fait par l'intermédiaire de canaux virtuels reliant une ou plusieurs entrées à une ou plusieurs sorties appelées données transmissibles. Les protocoles relatifs aux échanges de données sont définis au niveau d'une structure dite de connexions intermodules indépendantes des modules eux-mêmes.

En cas de partage de données entre les modules et si les conflits peuvent survenir, une opération de duplication des exemplaires de la donnée est réalisée.

Le système SYGARE propose une spécification d'une application en trois niveaux :

1. un niveau d'expression algorithmique des modules (strictement séquentiels) à l'aide d'un langage de programmation,
2. un niveau d'expression des enchaînements entre modules d'une même tâche. Dans ces enchaînements on peut exprimer le parallélisme et la séquentialité.
3. un niveau d'expression statique du parallélisme définie en termes de relations entre les événements et les tâches.

L'avantage de cette structuration de l'application est l'indépendance des niveaux de programmation. Ainsi, il est possible de modifier une tâche sans considération de la structure de celle-ci résultante du niveau 3, et inversement. Il est également possible de modifier le code d'un modèle sans altérer la structuration obtenue au niveau 2 et 3.

Il est à signaler que ce modèle stipule que le corps d'un module ne peut contenir que des actions de synchronisation ou de communication. Les points de synchronisation sont les seuls début et fin des modules qui les composent. La communication des données se fera également à chacun de ces points.

D'autres travaux ont été conduits dans [Ben 84], ils avaient pour objectif le développement d'outils de programmation nécessaires pour le niveau 3 [Tho 80]. Les concepts relatifs aux événements ont été particulièrement étudiés.

Une autre approche a été proposée dans [Sak 84]; elle consiste à décrire une application en se basant sur l'expression des communications entre les entités constituant l'application. L'approche consiste en un développement d'une solution en deux étapes :

1. la description fonctionnelle interne qui permet de décomposer l'application en un réseau de modules communicants, ainsi que la description du réseau des modules obtenus,
2. la programmation de l'application qui tient compte des contraintes exprimées dans la description fonctionnelle, et les contraintes liées à la répartition des modules sur l'ensemble des processeurs disponibles.

La programmation d'un module aboutit à une unité appelée "capsule". Celle-ci sera décrite dans le paragraphe suivant.

### II.2.2 Les capsules

Dans le cadre des applications informatiques temps réel et en particulier du projet SCEPTRE 2 [Asl 92], la nécessité de décomposition de l'application en composantes élémentaires a conduit à l'introduction de la notion de capsule. Une capsule est la composante élémentaire de l'application; c'est un objet destiné à isoler totalement ou partiellement un sous ensemble de l'application avec les propriétés suivantes :

- autonomie : une capsule est une unité de chargement susceptible d'être mise au point de façon indépendante. Elle est obligatoirement chargée sur un seul site, (2.1)
- création et initialisation : dans cette phase, l'exécutif lance le(s) objet(s) d'initialisation de la capsule. Ces objets sont définis comme attributs de la capsule lors de sa création, (2.2)
- reconfiguration : une capsule est une unité reconfigurable de l'application et ceci indépendamment du reste de l'application, (2.3)
- isolation et protection : le code d'une capsule demeure inviolable entre la création et la suppression de cette capsule. Toute exception levée au sein d'une capsule ne peut être traitée que par un objet de cette capsule ou par l'exécutif. Toute anomalie logicielle apparaissant au cours du fonctionnement d'une capsule ne doit pas affecter le reste de l'application. (2.4)

La capsule est caractérisée par :

- son interface qui définit l'ensemble des objets publics qui lui permettent de communiquer avec le monde extérieur, (2.5)
- son corps constitué par un ensemble d'objets, (2.6)
- ses attributs qui définissent l'ensemble des caractéristiques physiques et logiques qui vont permettre la gestion de la capsule dans les différentes phases de création, de chargement et d'exécution, (2.7)
- ses liens de communication dynamiques gérés par l'exécutif, ce qui évite la recompilation de l'application à chaque changement de capsule. L'application constituée de capsules, est ainsi reconfigurable en ligne, c'est à dire qu'elle ne nécessite ni réinitialisation ni arrêt du système en cours d'exécution. (2.8)

Elle est caractérisée aussi par son information d'état gérée par l'exécutif. Celui-ci assure à tout moment la cohérence entre les demandes d'opérations sur les capsules et l'état de ces capsules. Les états principaux répertoriés pour la gestion des capsules sont les suivants :

- non_existante : la capsule n'est pas définie,
- chargée : la capsule est chargée sur un site donné,
- en service : une demande de création de la capsule a été faite,
- active : une demande de démarrage de la capsule a été faite.

La modélisation de l'application du système d'automatisation à l'aide des capsules présente des avantages : (2.1), (2.3), (2.4) et (2.8). Effectivement ces propriétés garantissent l'autonomie de l'atome de distribution, ce qui facilitera sa mise au point, sa configuration de façon indépendante de l'application, ainsi que son isolation et sa protection par rapport aux autres éléments du système.

Cette modélisation présente également des insuffisances :

- au niveau de la définition du contrôle relatif à chaque capsule, nous pensons que celui-ci est insuffisant dans le cadre des systèmes d'automatisation, puisqu'il ne tient pas compte des états relatifs à la sémantique du traitement (mode d'utilisation, mode de marche [Bay 92a]), ni à ceux rendant le traitement plus adapté aux besoins utilisateurs (suspension, répétition, arrêt, ...),

- une capsule est une fusion entre traitement et contrôle, ce qui rend difficile la différenciation entre ces éléments, et par conséquent leur répartition séparée (traitement ou contrôle) ou combinée (traitement et contrôle).
- enfin, la caractérisation des données (données consommées, données produites) propres aux capsules, n'entre pas dans la définition précédemment donnée.

Pour ces raisons, nous proposons une modélisation structurelle des traitements élémentaires, que nous avons voulu générale et qui est caractérisée par :

- la séparation entre traitement, contrôle, et communication, ce qui laissera le choix aux utilisateurs de l'architecture adaptée à leurs besoins (distribuer les traitements et/ou le contrôle), et laissera la communication à la charge de la machine support,
- la caractérisation claire et précise des traitements, de leur contrôle, et des données qu'ils manipulent, ainsi que l'introduction de contraintes temporelles relatives à ces dernières, qui assureront entre autre le maintien de la cohérence du système,
- le caractère composition, permettant l'obtention de traitements de niveau hiérarchique plus élevé en faisant appel à des règles de composition.

De plus, la recherche et/ou l'application des méthodes de répartition nécessite l'étude des relations entre atomes de distribution, les moyens de mesure de ces relations et finalement leur classification. Ces informations seront indispensables pour la simplification et la résolution du problème de répartition en permettant d'établir des fonctions coût et des contraintes relatives à l'optimisation des solutions réparties.

### II.2.3 Les atomes de distribution

Bien que la modélisation des traitements élémentaires ne soit pas très détaillée, la terminologie relative à ceux-ci est riche dans les travaux traitant des domaines d'informatique parallèle et distribuée, ainsi que du temps réel [Mun 90], [Tal 91], [Che 89].

Le mot tâche est utilisé pour désigner soit un traitement élémentaire [Bro 84], soit un groupe de traitements élémentaires [Tho 80]. Il est défini comme une description algorithmique des mécanismes permettant de réaliser une fonction de l'application. Plus concrètement, il s'agit d'un code exécutable avec ou sans instance de contexte.

Une tâche est également définie avec plus de détail dans le projet SCEPTRE [Bro 84], comme un agent actif responsable de l'exécution par une machine d'un programme composé à partir du répertoire des instructions de cette machine. Elle ne peut entreprendre l'exécution d'une instruction de ce programme qu'après avoir terminé l'exécution de l'instruction précédente. En ce sens elle est séquentielle.

Une tâche SCEPTRE possède un nom et des attributs caractéristiques de la machine sur laquelle elle s'exécute et de la façon dont on la gère. Elle peut être programmée, microprogrammée ou même câblée (cas des tâches gérant les unités d'échange ou les coupleurs d'entrées/sorties).

Le mot processus est également très utilisé pour décrire un traitement élémentaire. Il est défini comme une suite d'actions exécutables nécessairement sur le même site, et à laquelle est associé un même contexte d'exécution.

La diversité de terminologie est dépendante des points de vues, et des domaines d'application. C'est pour cette raison que nous avons jugé nécessaire de définir et de caractériser les atomes de distribution, en se référant à notre domaine d'application : les systèmes automatisés de production en particulier et les applications temps réel en général. Dans cet objectif, un traitement élémentaire est nommé atome de distribution.

On définit un atome de distribution comme la plus petite entité non décomposable de l'ensemble des traitements du système d'automatisation. L'aspect "arbitraire" de cette définition offre à l'utilisateur une liberté de décision sur le niveau de décomposition de l'application ainsi que la possibilité de réutilisation de portions d'applications existantes qu'on est contraint de laisser "compactes".

L'ensemble des atomes de distribution constitue l'ensemble A introduit ci-dessus. Cependant, on a vu qu'il convenait de distinguer les aspects relatifs à l'implémentation physique d'un atome et ceux relatifs à sa fonction. Dans ce sens, on introduit l'ensemble S des services rendus par le système d'automatisation. Cet ensemble représente l'application vue sous l'aspect fonctionnel. L'ensemble des atomes A est alors vu comme l'ensemble des moyens logiciels (modules de code) ou matériels (fonctions câblées) permettant de rendre ces services.

L'atome de distribution, plus petite entité non décomposable de l'ensemble des traitements du système d'automatisation est, par définition, caractérisé par trois propriétés :

- un atome de distribution possède une identité fonctionnelle : à chaque atome de distribution (élément de A) est associé le service rendu (élément de S), par une application univoque : un atome de distribution donné ne rend qu'un seul service. Si les atomes de distribution ne sont pas dupliqués, un service n'est rendu que par un seul atome de distribution. Dans le cas contraire plusieurs atomes de distribution de l'ensemble A peuvent rendre le même service. La redondance exprimée ici est une redondance fonctionnelle, c'est à dire que les identificateurs des atomes de distribution dupliqués sont différents, bien que les atomes assurent la même fonction, et rendent le même service,
- un atome de distribution possède une identité géographique : en termes de distribution, l'atome de distribution est insécable, c'est à dire qu'il ne peut être affecté qu'à un seul site,
- un atome de distribution possède une identité d'exécution : en termes d'exécution, l'atome de distribution constitue également un atome insécable, c'est à dire qu'on ne peut en demander une exécution partielle. Cette propriété est liée à celle de l'identité fonctionnelle : elle signifie qu'un service associé à un atome de distribution sera rendu ou non rendu. La notion de service rendu partiellement n'est pas actuellement introduite dans le modèle. La propriété relative à l'identité d'exécution suppose l'existence d'une requête associée à chaque atome de distribution, permettant de demander cette exécution. Cette requête peut éventuellement être paramétrée.

### II.2.3.1 Caractérisation des atomes de distribution

Un atome de distribution (figure 2.3), que nous noterons, suivant les besoins "a" ou $a_i$, est caractérisé par son activité de traitement et son activité de gestion.

#### II.2.3.1.1 L'activité de traitement d'un atome de distribution

Soit X l'ensemble des variables, qui constitue la base de données du système. Ces variables sont de différents types, et elles sont utilisées (produites et/ou consommées) par les atomes de distribution de l'ensemble A.

Soit F(X) l'ensemble des parties de X.

L'activité de traitement de l'atome de distribution est caractérisée par :

- **un ensemble de variables produites** : P(a), à cet ensemble peut être associé la date de dernière production et éventuellement une information de validité. P est une application définie de la manière suivante :

$$P : A \rightarrow F(X)$$
$$a \rightarrow P(a)$$

- **un ensemble de variables consommées** : C(a), à chacune desquelles peut être associé un délai de péremption. C est une application définie de la manière suivante :

$$C : A \rightarrow F(X)$$
$$a \rightarrow C(a)$$

- **une règle de transformation :** T(a) des variables consommées en variables produites, qui constitue le modèle de comportement de l'atome de distribution. Cette règle de transformation peut être paramétrée.

D'un point de vue pratique, plusieurs versions d'une même variable - c'est à dire des valeurs de cette variable produites à des instants différents - peuvent être présentes simultanément dans la mémoire de la machine support.

### II.2.3.1.2 L'activité de gestion d'un atome de distribution

L'activité de gestion de l'atome de distribution est caractérisée par :

- **un graphe d'états** G(a) décrivant l'évolution des états de l'atome de distribution,
- **un ensemble de variables de contrôle** : Y(a) permettant d'évaluer les transitions du graphe d'états G(a),
- **une variable produite**: Z(a), indiquant l'état dans lequel se trouve l'atome a,
- **un ensemble de requêtes** : Q(a) qu'il émet éventuellement pour activer d'autres atomes de distribution,

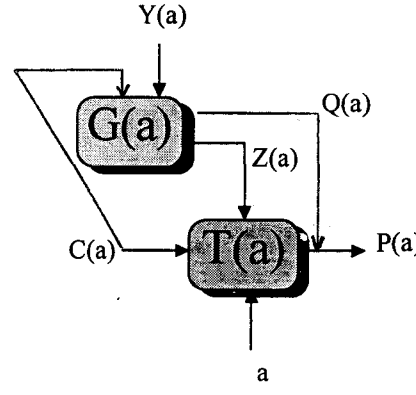



**Figure 2.3**

*Atome de distribution*

Certaines variables de contrôle du graphe d'états G(a) peuvent provenir de l'ensemble des variables produites par l'atome de distribution : l'intersection $(P(a) \cap Y(a))$ peut donc être non vide; cela signifie que l'évolution des états dépend d'événements liés aux valeurs des variables produites par l'atome de distribution. Dans le cas contraire $(P(a) \cap Y(a) = \varnothing)$ l'évolution des états de l'atome de distribution ne dépend que de l'extérieur.

## II.2.4 Caractérisation des variables

On distingue deux types de variables : les variables produites et les variables consommées.

### II.2.4.1 Les variables produites

Un atome de distribution produit un ensemble de variables P(a). Pour chaque variable x, l'application $P^{-1}(x)$ définit l'atome de distribution (unique) qui produit x, structurellement. $P^{-1}$ est définie de la manière suivante :

$$P^{-1} : X \rightarrow A$$
$$x \rightarrow P^{-1}(x)$$

On associe à chaque couple (atome de distribution producteur, version d'une variable produite), une date de production de la version $x_k$ de x :

$$\tau_k(P^{-1}(x))$$

On suppose que les dates de mise à jour des variables produites par un même atome de distribution sont identiques.

### II.2.4.2 Les variables consommées

Un atome de distribution consomme un ensemble de variables C(a). Celui-ci définit l'ensemble des variables consommées par l'atome de distribution "a" d'un point de vue structurel (liste des variables ), mais il est clair qu'un atome de distribution peut consommer plusieurs versions, décalées dans le temps, de la même variable. Ainsi, par exemple, l'atome de distribution suivant :

$$y_k = f( x_k, x_{k-1} )$$

consomme la variable x (structurellement) et produit la variable y.

Pour chaque variable x, l'application $C^{-1}(x)$ définit l'ensemble des atomes de distribution qui la consomment structurellement, c'est à dire qui en consomment au moins une version. $C^{-1}$ est définie de la manière suivante :

$$C^{-1} : X \rightarrow F(A)$$
$$x \rightarrow C^{-1}(x)$$

Chaque variable consommée doit être suffisamment fraîche pour qu'elle puisse être utilisée. Un délai de péremption $\delta(a,x)$ est associé à un couple (atome de distribution consommateur, version d'une variable consommée). Sachant que $x_k$ est produite à l'instant $\tau_k(P^{-1}(x))$, la date de fin de validité temporelle peut être calculée par :

$$\psi(a,x_k)=\tau_k(P^{-1}(x))+\delta(a,x)$$

Ces contraintes temporelles constituent des moyens de vérification de la cohérence temporelle du système d'automatisation distribué [Bay 92b].

## II.2.5 Classification des atomes de distribution

La définition des ensembles de variables produites et consommées pour chaque atome de distribution permet de les classer en trois classes :

- les atomes de distribution d'entrée : ce sont les atomes, qui ne consomment pas de variables mais en produisent. Cette catégorie est caractérisée par un ensemble vide de variables consommées ($C(a)=\varnothing$); ces atomes de distribution sont par exemple, ceux placés là où l'information relative au processus physique est acquise (ils ne consomment que des signaux),
- les atomes de distribution de sortie : ce sont les atomes de distribution qui ne produisent pas de variable mais en consomment. Cette catégorie est caractérisée par un ensemble de variables produites vide ($P(a)=\varnothing$); ces atomes de distribution sont par exemple, ceux placés là où l'information est fournie aux périphériques de sortie (ils ne génèrent alors, que des signaux),

- les atomes de distribution internes : ce sont les atomes de distribution qui consomment et produisent des variables. Cette catégorie est caractérisée par des ensembles de variables produites et de variables consommées non vides ($P(a)\neq\varnothing$ et $C(a)\neq\varnothing$). Un placement optimal de ces atomes nécessite l'application de méthodes de répartition,

La classe des atomes de distribution internes peut être divisée en deux :

- les atomes de distribution statiques : ce sont les atomes de distribution qui ne consomment pas de variable produite par eux mêmes, c'est à dire que l'intersection de l'ensemble des variables produites et de l'ensemble des variables consommées est vide ($P(a) \cap C(a)= \varnothing$). Les variables statiques sont des variables dont les atomes de distribution producteurs sont statiques (ne consomment pas des variables qu'ils produisent). Elles peuvent, cependant, exister dans un nombre de versions supérieur à un, selon les besoins des traitements consommateurs.
- les atomes de distribution dynamiques : ce sont les atomes de distribution qui consomment une partie des variables qu'ils produisent, c'est à dire que l'intersection de l'ensemble des variables produites et de l'ensemble des variables consommées est non vide ($P(a) \cap C(a)\neq\varnothing$). Les variables dynamiques sont des variables dont les atomes de distribution producteurs sont dynamiques (consomment une partie des variables qu'ils produisent). Les variables dynamiques doivent être initialisées.

### II.2.6 Redondance des atomes de distribution

Pour respecter les contraintes de sûreté de fonctionnement, pour assurer la disponibilité des traitements et des données, le concepteur du système d'automatisation peut recourir à la réplication de certains atomes de distribution.

L'implémentation redondante des atomes de distribution peut se réaliser en deux modes :

- sous forme de copies multiples à redondance passive. Cette forme permet l'activation d'une ou plusieurs copies d'atomes de distribution passifs, en cas

d'arrêt ou de mauvais fonctionnement des supports matériels (instruments, cartes, ...) sur lesquels les atomes de distribution jusque là actifs, sont implantés,

- sous forme de copies multiples à redondance active. Cette forme impose l'activation de toutes les copies pendant le fonctionnement du système. Dans ce cas la réplication des atomes contribue également à l'optimisation du temps de réponse en augmentant le parallélisme : par exemple, des atomes consommateurs peuvent accéder simultanément à des variables produites par des atomes producteurs redondants.

Quel que soit le mode de redondance des atomes de distribution, il doit assurer [Ara 94] le meilleur compromis entre :

- la tolérance aux fautes logicielles : la complexité des logiciels et les évolutions fonctionnelles régulières nécessitent à la fois une méthodologie de développement adaptée et une architecture supportant les anomalies résiduelles dues notamment à des concurrences complexes d'événements,
- les performances temps réel : les temps de réponse demandés exigent entre autres, une analyse critique de la messagerie entre atomes,
- la continuité et la qualité de service : les durées d'indisponibilité de fonctions et le taux de pertes de traitements suite à une défaillance doivent être nulles ou minimales. Il est, par ailleurs, obligatoire de réinsérer et synchroniser des répliques sans interruption de fonctionnement,
- le coût : le sur-dimensionnement (puissance de calcul, capacité mémoire, nombre de processeurs, bande passante de communication) doit conserver la compétitivité du système.

La réplication des atomes va donc engendrer la réplication des données. Celle-ci permettra d'augmenter la disponibilité et la sûreté de fonctionnement relatives aux données, dans le cas de pannes de machines ou de problèmes touchant partiellement le système de communication (quelques portions par exemples) [Gif 79], [Ber 81], [Ren 88]. Elle permet également de diminuer la charge réseau grâce à la disponibilité

des données sur plusieurs équipements, par contre elle augmentera la charge des équipements compte tenu de l'implantation des atomes redondants.

## II.2.7 Gestion des états d'un atome de distribution

En plus de la caractérisation des atomes de distribution effectuée ci-dessus, un autre aspect important, à savoir le contrôle relatif aux atomes de distribution, est à étudier. Cet aspect nous permet d'avoir une vue complète sur les changements d'états de l'atome de distribution et la manière de les gérer.

Disposant d'un modèle externe, mis en place dans le cadre de travaux sur l'instrumentation intelligente [Sta 94a], nous avons choisi d'utiliser le même formalisme que nous avons affiné. Dans ce sens, nous associons à chaque atome de distribution un certain nombre d'états qui peuvent être représentés par l'arborescence ci-dessous (figure 2.5) :

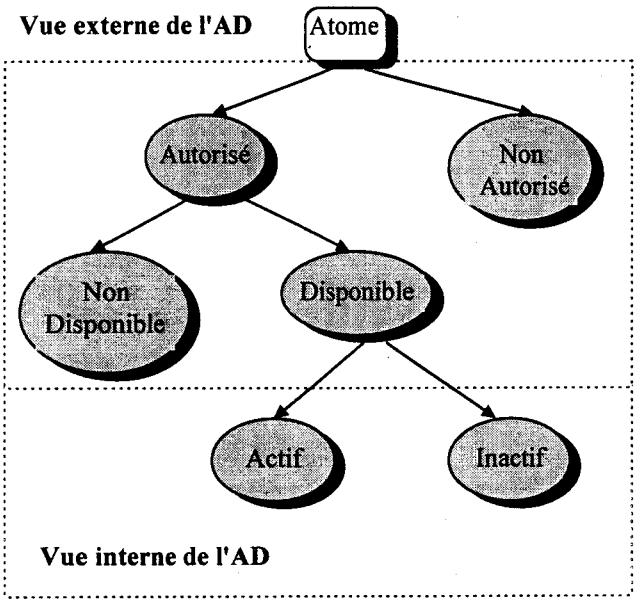



**Figure 2.5**

*Les états d'un atome de distribution*

### II.2.7.1 Vue externe des états d'un atome de distribution

La vue externe des états d'un atome de distribution permet de le décrire du point de vue d'un utilisateur, c'est à dire du point de vue d'une entité susceptible de formuler une requête d'exécution.

L'ensemble des services proposés par le système est organisé en modes d'utilisation [Bay 92b].

A l'entrée du système dans un mode d'utilisation comportant le service rempli par l'atome de distribution, celui-ci devient ***autorisé***. Une requête relative à un service autorisé sera acceptée par le système de gestion des états des atomes de distribution, elle sera refusée dans le cas contraire. Cependant, il ne suffit pas que la requête soit acceptée pour que le service associé à l'atome de distribution puisse être rendu : encore faut-il que celui-ci ait la possibilité de s'exécuter.

L'atome de distribution devient ***disponible*** si les supports matériels de l'activité T(a) sont disponibles et les ressources informationnelles (les variables de C(a)) sont validées (figure 2.6). Dans le cas contraire, les conditions nécessaires à son exécution correcte n'étant pas réunies, l'atome de distribution est indisponible, et la requête, bien qu'autorisée, ne peut être satisfaite.

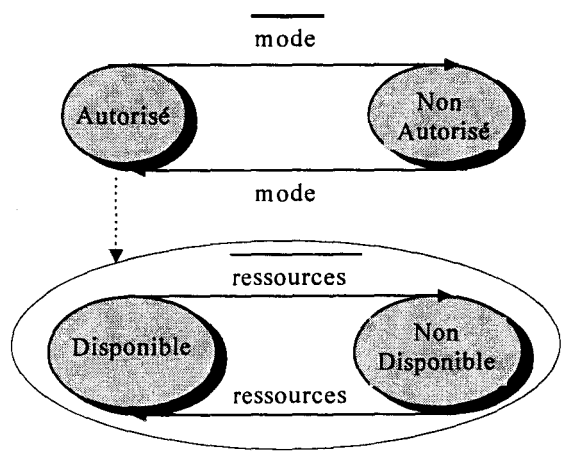



**Figure 2.6**

*Autorisation et disponibilité d'un atome de distribution*

**Légende de la figure 2.6**

mode : l'entrée du système dans un mode d'utilisation comportant le service délivré par l'atome de distribution

$\overline{\text{mode}}$ : entrée du système dans un mode d'utilisation ne comportant pas le service rempli par l'atome de distribution

ressources : les supports matériels de l'activité T(a) sont disponibles et les ressources informationnelles (les variables de C(a)) sont validées

$\overline{\text{ressources}}$ : l'une des conditions ci-dessus est fausse

### II.2.7.2 Vue interne des états d'un atome de distribution

La vue interne décrit les états d'un atome de distribution tels qu'ils sont gérés par le site auquel son exécution est confiée. Lorsqu'un atome de distribution est disponible, il peut se trouver dans différents états (figure 2.7). L'état ***inactif*** concerne un atome de distribution disponible, mais dont l'exécution n'a pas été demandée. L'atome de distribution passe dans l'état ***en attente*** dès qu'une requête relative à son exécution a été présentée, et qu'elle est valide, c'est à dire que son origine et son mode de transmission sont autorisés. Cependant, il est possible qu'à un moment donné, plusieurs requêtes relatives à des atomes de distribution différents implantés sur le même site et tous disponibles soient présentées. En général, un seul atome de distribution pourra passer dans l'état ***en exécution***, les autres restent dans l'état ***en attente***.

L'ensemble des atomes de distribution en attente à un moment donné constitue la file d'attente. Lorsque l'atome de distribution en exécution se termine, le choix du prochain atome de distribution à mettre en exécution, est réalisé par un ***ordonnanceur*** suivant une certaine ***stratégie***.

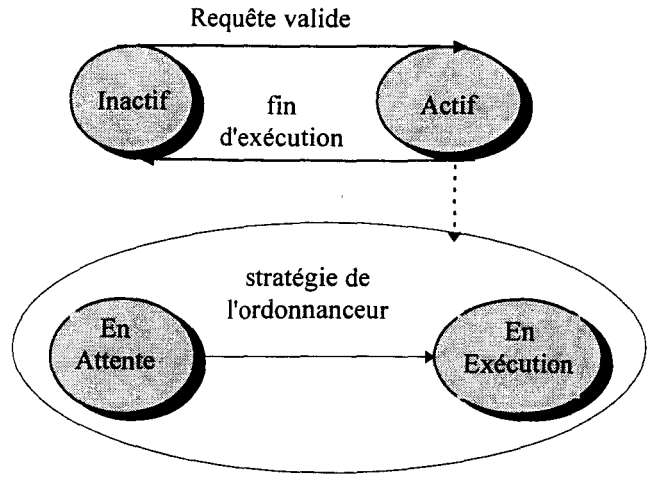



**Figure 2.7**

*Mécanismes de transition d'états de l'atome de distribution*

### II.2.7.2.1 Suspension

Lorsque l'exécution d'un atome de distribution se déroule sans qu'elle puisse être suspendue (pour exécuter un atome de distribution de priorité plus importante, par exemple), on dit que l'atome de distribution est ***non préemptif***. Dans le cas contraire - atomes de distribution préemptifs - l'ordonnanceur gère non seulement la liste des atomes de distribution en attente mais aussi la liste des atomes de distribution suspendus. L'état ***suspendu*** doit alors être ajouté au graphe des états représenté dans la partie inférieure de la figure 2.7, comme le montre la figure 2.8.

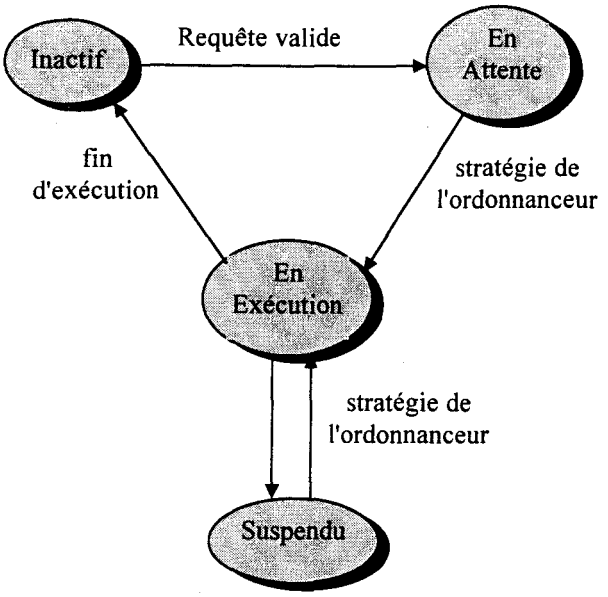



**Figure 2.8**

*Mécanisme de transition d'états dans le cas d'atomes de distribution préemptifs*

## II.2.7.2.2 Répétition

On peut prévoir la possibilité de multiples exécutions successives d'un atome de distribution, c'est à dire la possibilité qu'un atome de distribution retourne dans l'état *en attente* (au lieu de *inactif*) sur l'événement de fin d'exécution, sans qu'il soit nécessaire pour cela de présenter à nouveau, à chaque fois, la requête associée.

Pour introduire la possibilité d'exécution répétée d'un atome de distribution, le schéma de gestion des états de la figure 2.8 doit être modifié, en ajoutant la possibilité qu'un atome de distribution retourne dans l'état *en attente* (figure 2.9) après exécution, aussi longtemps que l'événement de "fin d'exécution répétée" ne sera pas survenu. Pour un tel atome de distribution, le schéma devient :

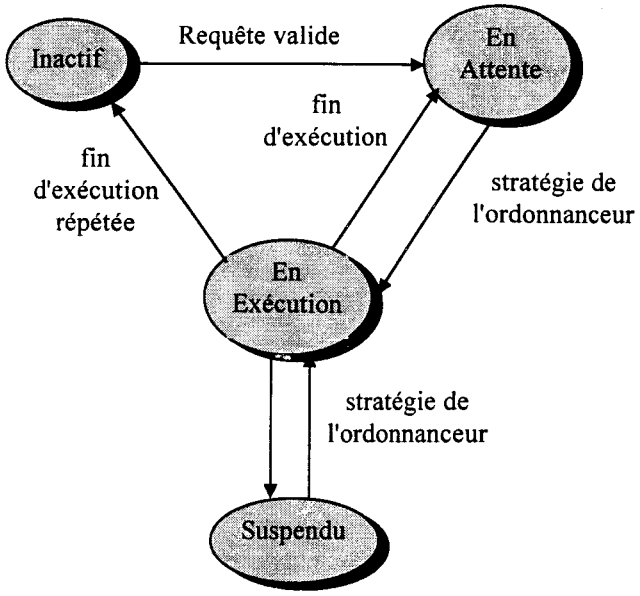



**Figure 2.9**

*Mécanisme de transition d'états pour la répétition d'un atome de distribution*

### II.2.7.2.3 Contre ordre

On peut envisager la possibilité de stopper l'exécution d'un atome de distribution sans qu'il puisse transmettre le résultat de son exécution. Les graphes de transition précédents ne nous permettent pas de réaliser ce souhait, puisqu'avec l'état *en exécution*, on confond le traitement effectué par l'atome de distribution et la fourniture de son résultat. Une possibilité de prise en compte d'un contre ordre consiste à décomposer l'état *en exécution* en deux états *en calcul* et *en mise à jour* (voir figure 2.10). L'état *en calcul* concerne l'application de la règle de transformation aux données consommées, tandis que l'état *en mise à jour* concerne l'écriture des variables produites dans la base de données du système.

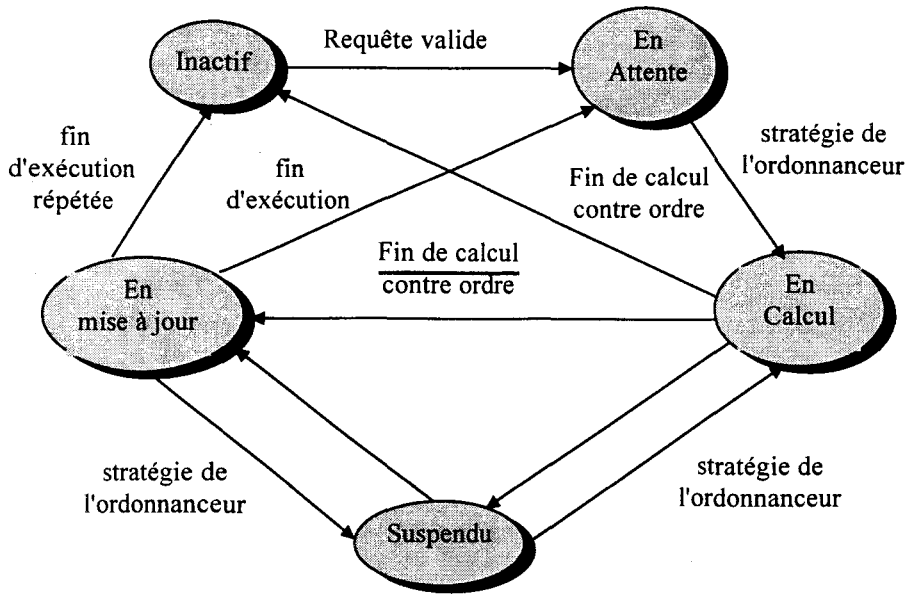



**Figure 2.10**

*Transition d'états avec contre ordre*

## II.2.8 Relations entre les atomes de distribution

Un atome de distribution $a_i$ peut consommer des données produites par un atome de distribution $a_j$, ce type de relation est, de toute évidence, une relation de transfert d'informations, qui se traduira par une charge réseau et des délais d'acheminement si les deux atomes de distribution sont affectés à des machines différentes.

Cette relation résulte de l'interconnexion des atomes de distribution par des flux de données, c'est pourquoi nous développons d'abord la relation de transfert d'informations. A ce niveau, nous considérons l'ensemble des données manipulées par les atomes de distribution, sans entrer dans le détail de la hiérarchie des flux d'informations. En effet, les systèmes automatisés de production assurent des fonctions multiples qui s'échangent des messages divers. Une telle diversité des messages conduit à qualifier et hiérarchiser des types de flux d'informations, liés aux contraintes à respecter dans l'échange des messages :

- les informations échangées entre les capteurs, actionneurs et automatismes de commande directe du procédé définissent un flux d'information le plus souvent périodique mettant en oeuvre beaucoup de données de faible volume mais aux

contraintes temporelles strictes (déterminisme de temps d'accès aux informations, respect d'un délai d'échange, ...),

- les informations relevant essentiellement de la coordination du fonctionnement du procédé piloté, ont un caractère moins déterministe (les contraintes temporelles sont moins strictes), et correspondent plus à des données synthétiques relatives au procédé (historiques, paramètres, calibrages d'équipements, ...),
- les informations de gestion de production sont constituées de messages de volume important, mais non astreints à des contraintes temporelles fortes (données de conception et fabrication assistées par ordinateur, données relatives aux plannings de production, ordres de commandes clients, ...).

Quels que soient les volumes de données échangées et les contraintes temporelles qui les caractérisent, on dira que deux atomes de distribution sont en relation lorsqu'une partie des données produites par l'un est consommée par l'autre. C'est une relation de transfert de données.

Pour modéliser les relations entre atomes de distribution on peut utiliser les concepts relatifs à la théorie des graphes. Ainsi l'ensemble des atomes de distribution du système d'automatisation à répartir, peut être représenté à l'aide d'un graphe G(A, U) dont A constitue l'ensemble des sommets représentant les atomes de distribution et U suivant le cas des arêtes ou des arcs désignant les relations de transfert de données entre les atomes de distribution. Les éléments de U peuvent être pondérés par des valeurs indiquant les coûts de communications unidirectionnels (si on utilise un graphe orienté) ou bidirectionnels (si on utilise un graphe non orienté) entre les différents atomes de distribution. La figure 2.11 illustre des relations de transfert de données, entre des atomes de distribution du système d'automatisation.

Nous présenterons dans ce qui suit les relations d'échanges d'informations, ainsi que les relations de précédence liant les atomes de distribution.

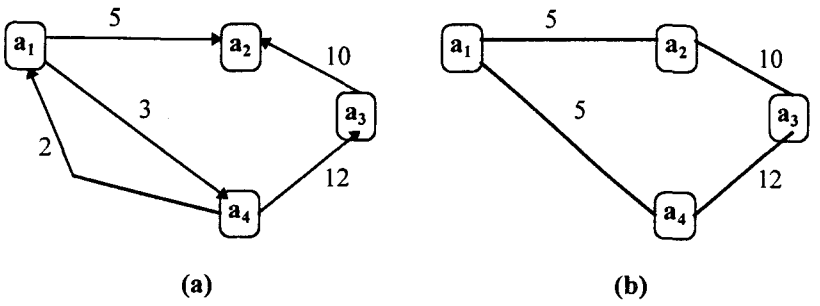



**Figure 2.11**

*Relations entre atomes de distribution*

*(a) Liens unidirectionnels (b) Liens bidirectionnels*

## II.2.8.1 Relations d'échanges d'information

Soit $P(a_i)$ (respectivement $C(a_i)$) l'ensemble des variables produites (respectivement consommées) par l'atome de distribution $a_i$,

Soit $P(a_j)$ (respectivement $C(a_j)$) l'ensemble des variables produites (respectivement consommées) par l'atome de distribution $a_j$,

On définit une relation $\varphi$ :

$$\varphi : AxA \rightarrow P(A)$$
$$(a_i,a_j) \rightarrow \varphi(a_i,a_j) = P(a_i) \cap C(a_j)$$

l'ensemble $\varphi(a_i,a_j)$ est constitué des variables communiquées de l'atome de distribution $a_i$ vers l'atome $a_j$.

**Définitions**

- deux atomes de distribution $a_i,a_j$ communiquent signifie que :

$$\varphi(a_i,a_j) \neq \varnothing \text{ ou } \varphi(a_j,a_i) \neq \varnothing,$$

- deux atomes de distribution $a_i,a_j$ communiquent symétriquement signifie que :

$$\varphi(a_i,a_j) \neq \varnothing \text{ et } \varphi(a_j,a_i) \neq \varnothing,$$

- deux atomes de distribution $a_i$,$a_j$ sont indépendants signifie que :

$$\varphi(a_i,a_j) = \varnothing \text{ et } \varphi(a_j,a_i) = \varnothing,$$

- deux atomes de distribution $a_i$,$a_j$ sont en communication orientée signifie que :

$$\varphi(a_i,a_j) = \varnothing \text{ et } \varphi(a_j,a_i) \neq \varnothing,$$

$$\text{ou } \varphi(a_i,a_j) \neq \varnothing \text{ et } \varphi(a_j,a_i) = \varnothing.$$

**Propriété**

- $\varphi(a_i,a_j) \cap \varphi(a_j,a_i) = \varnothing$, puisque une même variable ne peut pas être produite par deux traitements différents, c'est à dire $P(a_i) \cap P(a_j) = \varnothing$.

### II.2.8.2 Mesure de la relation d'échanges d'information

Soit $E_{ij}$ l'ensemble des variables communiquées de $a_i$ à $a_j$, et de $a_j$ à $a_i$. Cet ensemble constitue également l'ensemble des variables produites par $a_i$ (respectivement $a_j$) et consommées par $a_j$ (respectivement $a_i$) :

$$E_{ij} = \varphi(a_i,a_j) \cup \varphi(a_j,a_i) = (P(a_i) \cap C(a_j)) \cup (P(a_j) \cap C(a_i))$$

La mesure de la charge réseau engendrée par les échanges de variables entre $a_i$ et $a_j$ est naturellement fonction des éléments de $E_{ij}$. On considère que cette charge est nulle si les deux atomes de distribution concernés sont sur un même équipement.

Soit $V(a_i,a_j)$ la charge réseau, $V(a_i,a_j) = \sigma(E_{ij})$. La fonction $\sigma$ est dépendante :

- du type des variables transmises à travers le système de communication. En effet la transmission de deux variables de types différents, engendrera des charges différentes (exemple : transmission d'un entier, transmission d'un tableau d'entiers),
- du profil de transmission des variables : avant de transmettre les variables, un protocole de communication rajoute des données de contrôle (adresse de destinataire, données de cohérence, ...),

- des fréquences des échanges entre les atomes de distribution. En effet les atomes de distribution communiquant à des fréquences élevées engendreront des charges réseaux plus élevées.

Quelques exemples de calcul pratique des charges réseau seront donnés au chapitre III.

## II.3 Ordonnancement des atomes de distribution

Ordonnancer consiste à faire le planning d'exécution d'un ensemble fini d'atomes de distribution soumis à des contraintes temporelles et à des contraintes de ressources. Cette tâche est réalisée à l'aide des algorithmes d'ordonnancement.

La littérature abonde d'algorithmes d'ordonnancement dans les systèmes d'automatisation en particulier et les systèmes temps réel en général. Des travaux de synthèse ont été proposés dans [Cas 88], [Xu 93] et [Ala 92], [Car 94].

Les algorithmes d'ordonnancement sont séparables en deux catégories distinctes selon que leur objet est l'ordonnancement local ou global [Cas 88]. Un algorithme d'ordonnancement local se charge de l'ordonnancement d'un sous-ensemble d'atomes de distribution implantés sur une machine donnée. Cependant, un algorithme d'ordonnancement global se charge de l'ordonnancement de tous les atomes de distribution de A.

Les stratégies d'ordonnancement local ou global des tâches sont des éléments déterminants [Bay 95] d'une part du choix des machines, des exécutifs et des moyens de communication, d'autre part de la distribution des atomes sur une architecture matérielle donnée. En effet, les tâches dans une architecture opérationnelle résultent de la projection d'un atome de l'architecture fonctionnelle. Cet atome est généralement soumis à des contraintes de temps ; il est donc primordial que les tâches soient exécutées et terminées en respectant ces contraintes. C'est sur l'algorithme d'ordonnancement que repose la solution du problème de distribution.

Les algorithmes d'ordonnancement globaux et locaux peuvent être également regroupés en fonction du mode d'utilisation du processeur (préemptif/non préemptif). Le comportement des atomes durant le fonctionnement du système, et la connaissance que les algorithmes ont de ce comportement, représente un autre critère de classement [Ala 92] :

- Les algorithmes d'ordonnancement dynamiques : à un instant donné, ils sont capables de générer une nouvelle séquence d'ordonnancement en fonction de l'état courant du système en général, et des nouvelles demandes d'exécution des atomes de distribution en particulier. Les algorithmes adaptatifs font partie de cette classe. Ils prennent en compte toute circonstance nouvelle (un atome plus prioritaire que d'autres, par exemple).
- Les algorithmes d'ordonnancement semi-dynamiques : ils doivent connaître les prochaines dates des demandes d'exécution relatives aux atomes de distribution. Ainsi le comportement des atomes de distribution est partiellement prédéterminé.
- Les algorithmes statiques : ils doivent connaître, au démarrage du système, tous les paramètres temporels relatifs aux atomes de distribution de la configuration à ordonnancer. Ils déterminent une séquence d'ordonnancement sans que ce choix puisse être ultérieurement remis en cause.

Les algorithmes présentés ci-dessus peuvent être centralisés (implantés sur un seul équipement qui gère le tout), ou distribués (implantés sur plusieurs équipements). Ils peuvent être optimaux (un algorithme est dit optimal si, lorsqu'il échoue tous les autres algorithmes échoueront), comme ils peuvent être sous-optimaux quand ils déterminent des solutions approchées des meilleures séquences. Une classification détaillée peut être retrouvée dans [Car 94], [Ala 92].

## II.4 Implantation des atomes

L'architecture matérielle du système d'automatisation est caractérisée par son hétérogénéité. En effet les équipements peuvent provenir de constructeurs différents imposant leurs propres systèmes d'exploitation, langages de développement, ... En dépit de cette hétérogénéité les atomes doivent échanger des données, d'où la nécessité de disposer d'une norme facilitant cette tâche.

MMS (Manufacturing Messsage Specification) est une norme (ISO 8506) de messagerie industrielle de la couche application qui assure l'interopérabilité des équipements de production hétérogènes connectés par un réseau. Elle offre un ensemble de services génériques qui permettent d'accéder aux fonctionnalités spécifiques de chaque équipement de production de manière transparente. Le protocole de la couche application définit un ensemble de services, sous un schéma de communication client/serveur, permettant la conception des systèmes automatisés de production distribués.

Le modèle VMD (Virtuel Manufacturing Device) est le concept de base de MMS. Il définit le comportement externe d'un *serveur* MMS. Celui-ci est un équipement de production qui contient un VMD accompagné de ces objets, et dont les fonctionnalités particulières sont projetées sur les services MMS. Cela permet à l'application d'accéder de façon homogène aux fonctionnalités de tous les équipements en cachant leurs particularités. Le comportement externe visible de chaque serveur MMS est défini par :

- les objets contenus dans un objet VMD,
- les comportements des services pour accéder et manipuler ces objets.

Un *client* est un processus d'application qui demande des services (des données ou des actions). Les applications sont définies par des relations de type client/serveur entre les processus d'application et les équipements de production.

Les objets et les équipements virtuels conformes au modèle VMD sont indépendants de la marque, du langage, du système d'exploitation, ... Chaque développeur d'une application MMS serveur est responsable de l'encapsulation des détails relatifs aux objets réels, en fournissant une fonction exécutive. Celle-ci convertit les objets réels en objets virtuels fournis par le modèle VMD et auxquels les clients MMS font appel. Le modèle VMD possède une fonction exécutive qui exécute les demandes de services associées aux objets VMD et qui constitue le noyau du modèle d'exécution VMD. Il possède un modèle d'exécution flexible qui se fonde sur les invocations de programmes composées de procédures et de données contenues dans des domaines.

Le modèle d'exécution du VMD définit deux objets pour contrôler l'exécution des programmes inclus dans un VMD :

- l'objet *domaine* : un domaine MMS est défini comme un objet qui représente les ressources incluses dans un équipement (par exemple la mémoire dans laquelle le VMD est stocké). Il peut être vide ou rempli par des informations telles que les données produites par un atome, le code des atomes, ...
- l'objet *Invocation de Programmes* : c'est la réunion d'un ou plusieurs domaines, contenant le code d'un ou plusieurs atomes, les données produites et consommées par les atomes, de façon à définir un programme exécutable. C'est à travers la manipulation des invocations de programmes qu'un processus d'application MMS contrôle l'exécution des programmes d'un VMD. Les invocations de programmes peuvent être démarrées (Started), arrêtées (Stopped), initialisées (Reset), ... par les clients MMS. L'invocation d'un programme est une exécution qui consiste en une collection d'un ou plusieurs domaines. Les équipements simples ayant des structures d'exécutions simples supportent généralement une seule invocation de programme, en faisant appel à un seul domaine.

En conclusion, une solution d'implantation des traitements sur les équipements peut être vue comme une projection d'un sous-ensemble d'atomes sur un VMD associé à un équipement. Le domaine du VMD contiendra entre autre, les données produites par les atomes. A l'invocation d'un atome, plusieurs autres domaines pourraient être chargés, à savoir les domaines des autres atomes produisant des données ou réalisant des actions nécessaires à l'exécution de l'atome invoqué.

## II.5 Conclusion

Dans ce chapitre nous avons présenté un modèle structurel d'atome de distribution. Celui-ci constitue l'élément de base de l'architecture fonctionnelle. Il est caractérisé par deux activités : une activité de traitement et une activité de gestion. L'intérêt d'une telle décomposition est la possibilité de répartir les traitements indépendamment de leur gestion. Le modèle proposé est intéressant par son indépendance vis-à-vis de l'architecture matérielle. Il est caractérisé aussi par sa flexibilité par rapport à cette dernière puisqu'il sépare les éléments de traitement, de communication et de contrôle. En plus il fournit des outils permettant d'établir des fonctions coût, des contraintes et des moyens de vérification de cohérence du système d'automatisation. Ces outils sont indispensables pour le développement de méthodes permettant la distribution des systèmes d'automatisation.

Dans le chapitre suivant nous nous intéressons aux critères, contraintes et algorithmes de répartition, permettant de projeter l'architecture fonctionnelle sur l'architecture matérielle.

When an idea is wanting,
a word can always be found to take
its place

**Goethe**

# Chapitre III

# Des stratégies pour répartir le système d'automatisation

*Introduction* ..... 75
*III.1 Les solutions admissibles* ..... 75
*III.2 Classification des stratégies de répartition* ..... 76
*III.3 Les critères de répartition* ..... 78
*III.3.1 Le coût d'exécution d'un atome de distribution* ..... 79
*III.3.2 Les coûts de communication entre atomes de distribution* ..... 81
*III.3.3 Le coût d'interférence entre atomes de distribution* ..... 84
*III.4 Les contraintes* ..... 85
*III.4.1 Les contraintes logiques* ..... 86
*III.4.2 Les contraintes physiques* ..... 87
*III.5 Fonctions coûts et architecture du système d'automatisation* ..... 88
*III.6 Les algorithmes de répartition* ..... 89
*III.6.1 Les algorithmes itératifs de recherche locale* ..... 90
*III.6.2 Recherche d'un homomorphisme faible entre graphes* ..... 92
*III.6.3 Partition des coupes minimales d'un graphe* ..... 95
*III.6.5 Les algorithmes génétiques* ..... 97
*III.6.6 Autres approches utilisées* ..... 100
*III.7 Conclusion* ..... 101

## Introduction

L'objectif de ce chapitre est de présenter et d'adapter un ensemble de stratégies de répartition rencontrées essentiellement dans la littérature informatique parallèle et distribuée, au contexte des systèmes automatisés de production. Ainsi nous présenterons dans ce qui suit des critères, des contraintes et des algorithmes permettant de projeter l'architecture fonctionnelle du système d'automatisation sur l'architecture matérielle de ce dernier.

Nous considérons que l'architecture fonctionnelle du système d'automatisation est constituée d'un ensemble d'atomes de distribution $A= \{a_1, a_2, ..., a_n\}$ qui communiquent entre eux par envoi et réception de messages, et que l'architecture matérielle du système d'automatisation est composée d'un ensemble de machines $M= \{m_1, m_2, ..., m_k\}$ qui communiquent entre elles par l'intermédiaire d'un ou plusieurs systèmes de communication.

## III.1 Les solutions admissibles

Rappelons que notre objectif est de partitionner l'ensemble des atomes de distribution en classes. Ces classes seront ensuite affectées aux machines du système d'automatisation.

Pour un ensemble de contraintes donné, seules certaines, parmi les partitions de l'ensemble A en k classes, seront admissibles. Pour réaliser cette tâche, nous sommes confrontés à un ensemble de difficultés :

- une première difficulté du problème consiste à définir l'ensemble des contraintes retenues,

- une deuxième difficulté consiste à évaluer si une partition candidate les satisfait ou non. Une partition qui satisfait les contraintes est appelée partition admissible,
- une dernière difficulté consiste à choisir une solution, parmi toutes celles qui satisfont les contraintes. Il s'agit là du problème de l'optimisation de la distribution, qui suppose que l'on ait défini un (ou plusieurs) critère(s) permettant de comparer entre les différentes solutions admissibles. Le problème de la définition et de l'évaluation du (des) critère(s) est comparable à celui de la définition et de l'évaluation des contraintes. On se trouve confronté à un problème NP-complet, le nombre de partitions d'un ensemble de n éléments en k classes étant très important, il est hors de question sauf cas très particuliers, de rechercher les solutions par énumération. La dernière difficulté est ainsi relative à la mise au point d'algorithmes fournissant des résultats acceptables dans un temps acceptable.

La distribution aboutira donc à la constitution d'un ensemble d'applications locales (un sous-ensemble d'atomes de distribution) $Z=\{z_1, z_2, ..., z_k\}$, une répartition de la base de la base de données $D=\{d_1, d_2, ..., d_k\}$ ($d_i$ est une partie de X), et d'un ensemble d'applications de contrôle $G=\{g_1, g_2, ..., g_k\}$. Ce travail est consacré à la répartition de l'ensemble des atomes de distribution A.

## III.2 Classification des stratégies de répartition

Il existe deux grandes classes de stratégies de répartition (voir figure 3.1) :

- la première classe concerne les stratégies de répartition statique. Ce type de stratégie consiste à effectuer le placement des atomes de distribution sur les machines avant la mise en marche du système. Le placement ainsi effectué ne changera pas durant le fonctionnement du système. Ces stratégies ne tiennent pas compte du comportement dynamique de ce dernier. Elles supposent que la charge totale du système reste inchangée pendant l'exécution des atomes de distribution. De telles situations sont fréquentes

dans les systèmes automatisés de production, puisque les machines sont dédiées à des applications particulières,

- la deuxième classe concerne les stratégies de répartition dynamique. Ce type de stratégie consiste à affecter les atomes de distribution aux machines pendant le fonctionnement du système. Ces stratégies conduisent à exécuter un atome de distribution sur une machine, qui peut être différente de celle qui a supporté sa dernière exécution. Ainsi, un atome de distribution peut *migrer* d'une machine à une autre, en fonction de la charge courante du système.

Nous nous sommes intéressés dans ce travail à la première classe de stratégies de répartition. La plupart de ces stratégies se basent soit sur des méthodes exactes, soit sur des méthodes approchées.

Les méthodes exactes donnent des solutions optimales. Elles utilisent généralement les techniques issues de la théorie des graphes [Sto 77], [She 85], ou celles de la programmation mathématique [Bok 81a], [Che 90], [Sin 87], ...

Les méthodes approchées donnent des solutions réalisables avec une complexité polynomiale. Une multitude d'exemples concernant ces méthodes se trouve dans la littérature : le groupement de processus [Per 83], [Ram 86], [Nic 90], limitation du routage [Bok 81a], ...

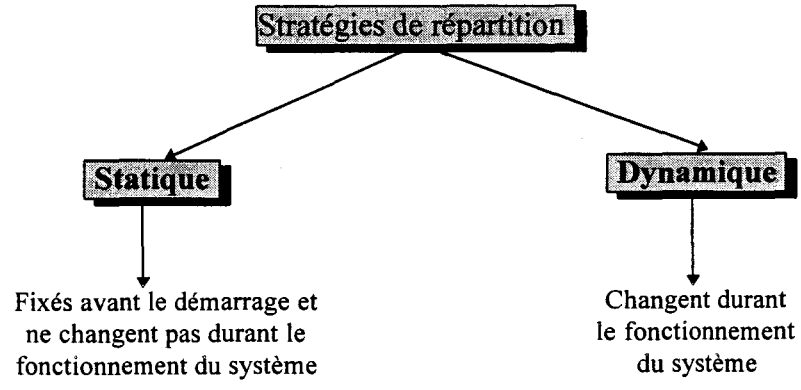



**Figure 3.1**

*Classification des stratégies de répartition*

Ces approches font généralement appel à des fonctions coût appelées aussi fonctions objectif, pour évaluer les performances des répartitions possibles et en choisir une. Elles font aussi appel à des heuristiques et des contraintes pour tenir compte des exigences des utilisateurs et du système et afin d'accélérer la recherche de la solution compte tenu de l'aspect combinatoire du problème de distribution.

Nous définissons dans ce qui suit la fonction coût V, un ensemble de contraintes et des heuristiques qui nous serviront comme des outils permettant la répartition de notre système d'automatisation.

## III.3 Les critères de répartition

Diverses fonctions coût sont utilisées comme critères de performance, permettant de comparer les placements possibles des éléments de A sur les éléments de M.

Un placement optimal est celui qui minimise la fonction coût. Celle-ci dépend des coûts d'exécution des atomes de distribution placés sur les diverses machines, et des coûts de communication et d'interférence entre atomes de distribution. Il existe d'autres critères comme par exemple le taux d'utilisation des ressources (processeur, mémoire, liens de communication) et la tolérance aux pannes.

Le problème dans sa généralité est NP_complet, c'est-à-dire qu'il n'existe pas d'algorithme exact pour la résolution du problème, pour lesquels le temps maximum de résolution soit borné par une fonction polynomiale de la taille du problème [Gar 79].

Il faut noter qu'il existe des cas particuliers où le problème d'allocation peut être réduit à un problème polynomial :

- pour un système bimachines, des recherches effectuées par Stone [Sto77] ont montré comment un placement optimal peut être trouvé efficacement ;
- pour des machines identiques supportant chacune au plus deux atomes de distribution, un algorithme a été proposé pour trouver un placement optimal

en un temps polynomial dans le cas où le nombre d'atomes de distribution est inférieur ou égal à deux fois le nombre de machines [Lo 88].

- dans le cas où le graphe de précédence entre processus est un p-arbre partiel (sous-graphe d'un p-arbre) et pour un nombre quelconque de machines, en particulier dans le cas d'un arbre (p=2) et dans le cas d'un graphe série-parallèle (p=3), des algorithmes polynomiaux ont été proposés [Fer 89], [Bok 81b], [Tow 86].

Nous listons dans ce qui suit les paramètres mis en jeu pour l'évaluation du placement des atomes de distribution sur les différentes machines, dans l'objectif d'obtenir une architecture opérationnelle admissible vérifiant un ensemble de contraintes, et satisfaisant un ensemble de critères. Dans ce chapitre et en guise d'illustration, nous considérons uniquement *la contrainte d'exploitation de toutes les machines*, qui stipule que chacune des machines doit supporter au moins un atome de distribution.

Dans ce qui suit, nous utiliserons un exemple à quatre atomes de distribution et trois machines pour illustrer la définition des diverses fonctions coûts envisageables :

Soit $A= \{a_1, a_2, a_3, a_4\}$ l'ensemble des atomes de distribution et soit $M=\{m_1, m_2, m_3\}$ l'ensemble des machines.

## III.3.1 Le coût d'exécution d'un atome de distribution

Le coût d'exécution e(a, m) traduit le coût à supporter en plaçant un atome de distribution *a* sur une machine m. Ce coût est généralement valué en unités de temps (e(a, m)=20 signifie que l'atome "a" met 20 unités de temps pour s'exécuter sur la machine m, par exemple). Il est possible de tenir compte de la fréquence d'exécution d'un atome sur les machines supports. Ainsi le coût d'exécution e(a, m) sera multiplié par la fréquence d'exécution de l'atome. Ainsi le coût d'exécution tenant compte de la fréquence exprimera la charge moyenne de la machine support par unité de temps.

Le coût d'exécution peut être soit évalué à partir des simulations, soit calculé par des méthodes basées sur la complexité des algorithmes (le nombre de boucles, le nombre d'appels de sous programmes, ...). Ce coût d'exécution varie en fonction de la puissance de traitement des machines.

Les coûts d'exécution peuvent être formellement décrits par une application e définie de la manière suivante :

$$e : A \times M \rightarrow \Re$$

$$(a, m) \rightarrow e(a, m)$$

La charge d'une machine $m_j$ est estimée à $\sum_{i=1}^{n} e(a_i, m_j)x_{ij}$. Tandis que la charge totale du système est estimée à :

$$f_e(R(A)) = \sum_{i=1}^{n}\sum_{j=1}^{k} e(a_i, m_j)x_{ij}$$

telle que $x_{ij}=1$ si $a_i$ est implanté sur la machine $m_j$ 0 sinon. La table 3.1 présente les coûts d'exécution des atomes de distribution sur les machines de notre exemple.

| | $m_1$ | $m_2$ | $m_3$ |
|---|---|---|---|
| $a_1$ | **10** | 20 | 30 |
| $a_2$ | 40 | **5** | 10 |
| $a_3$ | 70 | **50** | 80 |
| $a_4$ | 50 | 80 | **20** |

**Table 3.1**

*Coûts d'exécution des atomes de distribution sur les machines.*

Une des solutions envisageables est {($a_1$, $m_1$), ($a_2$, $m_2$), ($a_3$, $m_2$),($a_4$, $m_3$)} et le coût associé est égal à 85.

Les charges des machines sont évaluées à :

charge de la machine $m_1$ = 10,

charge de la machine $m_2$ = 5+50=55,

charge de la machine $m_3$ = 20.

## III.3.2 Les coûts de communication entre atomes de distribution

Nous allons essayer de proposer un modèle qui permet de calculer la charge réseau et par conséquent le coût de communication (voir chapitre II). Nous nous limiterons dans ce travail à deux cas : le premier concerne l'envoi indépendant des données, et le deuxième concerne l'envoi regroupé des données. Nous considérons dans ce paragraphe que les atomes de distribution sont activés de façon périodique, et que la communication s'effectue en point à point.

### Premier cas : Transmission indépendante des variables

Soit $x \in E_{ij}$. Soit l(x) la longueur de la variable x exprimée en octets, par exemple.

Soit f(x) la fréquence de transmission de la variable x. On suppose que cette fréquence est égale à la fréquence de tir $f_i$ de son atome de distribution producteur $a_i=(P^{-1}(x))$.

Nous estimons la charge réseau engendrée par la transmission de la variable x de $a_i$ vers $a_j$ par :

$$v(x)=f(x)l(x)$$

sachant qu'on suppose que $a_i$ et $a_j$ sont placés sur des machines différentes.

La charge réseau engendrée par la transmission de tous les éléments de Eij, peut être déduite de la formule précédente ainsi

$$V(a_i, a_j) = \sigma(Eij) = \sum_{x \in Eij} f(x)l(x) \text{ bits/s}$$

Le protocole de communication rajoute généralement un ensemble d'informations à chaque variable transmise (source, destination, données de contrôle, ...), ceci engendre également une charge additionnelle qui peut être évaluée à s. La nouvelle charge réseau devient :

$$V(a_i,a_j)= \sigma(Eij) = \sum_{x \in Eij} f(x)(l(x) + s)$$

**Deuxième cas : Transmission regroupée des variables**

Dans un objectif de minimisation de la charge réseau, plusieurs données à transmettre, peuvent être regroupées dans des structures uniques (trame par exemple), et acheminées à travers le système de communication.

Soit $l(\vartheta_{ij})$ (respectivement $l(\vartheta_{ji})$) la charge engendrée par la transmission des variables de $a_i$ (respectivement $a_j$) vers aj (respectivement $a_i$).

$l(\vartheta_{ij})= \sum_{\varphi(ai,aj)} l(x)$ et $l(\vartheta_{ji})= \sum_{\varphi(aj,ai)} l(x)$.

La charge engendrée sans tenir compte des fréquences de transmission de données sera égale à :

$$l(\vartheta_{ij})+l(\vartheta_{ji})$$

Les $l(\vartheta_{ij})$ octets seront envoyés de $a_i$ vers $a_j$, tandis que les $l(\vartheta_{ji})$ octets seront envoyés de $a_j$ vers $a_i$. Il s'agit de minimiser le nombre de trames à envoyer sur le réseau, de façon à optimiser le temps de réponse et à libérer le réseau le plus rapidement possible.

Une trame est généralement composée :

- d'un préambule contenant l'adresse du producteur et du consommateur,

- d'un corps contenant les variables,
- d'un postambule permettant de vérifier entre autre la cohérence des données envoyées par rapport aux données reçues.

Soit $ntr_i$ le nombre de trames nécessaire à l'envoi des $l(\vartheta_{ij})$ octets de $a_i$ vers $a_j$.

Soit $ntr_j$ le nombre de trames nécessaire à l'envoi des $l(\vartheta_{ji})$ octets de $a_j$ vers $a_i$.

Soit $\omega_{ij}^{k}$ (respectivement $\omega_{ji}^{k}$) la k ème trame envoyée de $a_i$ (respectivement $a_j$) vers $a_j$ (respectivement $a_i$).

La charge réseau est donc fonction de $f_i$, $f_j$, $\omega_{ij}^{k}$, et $\omega_{ji}^{k}$.

$$l(\omega_{ij}^{k})+l(\omega_{ji}^{k})$$

sachant que :

$$l(\omega_{ij}^{k})=\sum_{k=1}^{nbti} f_i \omega_{ij}^{k} \text{ et } l(\omega_{ji}^{k})=\sum_{k=1}^{nbtj} f_j \omega_{ji}^{k}$$

et étant donné que la fréquence d'envoi des trames sont égales à celle de l'envoi des variables.

Dans ce paragraphe, nous avons présenté un modèle simple, permettant de calculer la charge réseau, engendrée par les échanges d'information entre deux atomes de distribution.

Les coûts de communications entre atomes de distribution présentés au chapitre II, peuvent être formellement décrits par une application V définie de la manière suivante :

$$V : A \times A \rightarrow \Re$$

$$(a_i, a_j) \rightarrow V(a_i, a_j)$$

Le coût de communication total entre les atomes de distribution sur les machines peut être évalué par la fonction suivante :

$$f_c(R(A))= \sum_{p=1}^{k}\sum_{q=1}^{k}\sum_{i=1}^{n}\sum_{j=1}^{n} V(a_i, a_j)x_{ip}x_{jq} \text{ avec } p\neq q$$

La table 3.2 présente les coûts de communication entre atomes de distribution.

| | $a_1$ | $a_2$ | $a_3$ | $a_4$ |
|---|---|---|---|---|
| $a_1$ | 0 | 20 | 0 | 40 |
| $a_2$ | | 0 | 5 | 70 |
| $a_3$ | | | 0 | 35 |
| $a_4$ | | | | 0 |

**Table 3.2**

*Coûts de communication entre atomes de distribution*

Une des solutions envisageable est {$(a_1, m_1)$, $(a_4, m_1)$, $(a_3, m_2)$, $(a_2, m_3)$} et le coût associé est égal à 130.

### III.3.3 Le coût d'interférence entre atomes de distribution

Les coûts de communications entre atome de distribution représentent un moyen d'attraction des atomes de distribution entre eux. Ainsi en regroupant ensemble les atomes de distribution ayant un coût de communication élevé on minimise le coût de communication total.

Les coûts d'exécution des atomes de distribution sur les machines représentent également un moyen d'attraction entre ces derniers (machines et atomes). Ainsi en plaçant les atomes de distribution sur les machines telle que leur coût d'exécution soit minimal, on réduit le coût d'exécution total.

Le coût d'interférence entre atomes de distribution quant à lui, représente un moyen de répulsion puisqu'il permet d'éviter le regroupement de certains atomes de distribution. Il reflète le degré d'incompatibilité du placement de deux atomes de distribution sur la même machine. Par exemple deux atomes de distribution utilisant beaucoup d'espace mémoire auront un coût d'interférence plus important que celui entre

deux atomes de distribution dont l'un utilise beaucoup d'espace mémoire et l'autre beaucoup les entrées sorties. Les simulations montrent que l'ajout des coûts d'interférence, est un facteur qui améliore considérablement le degré de parallélisme dans le placement des atomes de distribution en utilisant la méthode des coupes minimales [Mun 90].

Les coûts d'interférence entre atomes de distribution peuvent être formellement décrits par une application h définie de la manière suivante :

$$h : A \times A \rightarrow \Re$$

$$(a_i, a_j) \rightarrow h(a_i, a_j)$$

Le coût d'interférence moyen entre les atomes de distribution sur les machines peut être évalué par la fonction suivante :

$$f_h(R(A)) = \sum_{p=1}^{k} \sum_{i=1}^{n} \sum_{j=1}^{n} h(a_i, a_j) x_{ip} x_{jp}$$

La table 3.3 présente les coûts d'interférence entre atomes de distribution.

Une des solutions envisageable est $\{(a_1, m_1), (a_4, m_1), (a_2, m_2), (a_3, m_3)\}$ et le coût associé est égal à 40.

| | $a_1$ | $a_2$ | $a_3$ | $a_4$ |
|---|---|---|---|---|
| $a_1$ | 0 | 100 | 0 | 40 |
| $a_2$ | | 0 | 5 | 99 |
| $a_3$ | | | 0 | 35 |
| $a_4$ | | | | 0 |

**Table 3.3**

*Coûts d'interférence entre atomes de distribution*

## III.4 Les contraintes

Les contraintes sont dans la plupart des cas dépendantes de l'architecture matérielle (charge et taille mémoire de la machine, charge réseau, contraintes de

ressources, contraintes temporelles, ...). Ces contraintes facilitent généralement la répartition en imposant des affectations (tel atome de distribution doit être sur telle machine) ou de non affectation (tel atome de distribution ne doit pas être sur telle machine). Les contraintes permettent aussi d'éviter le regroupement des atomes de distribution sur une même machine (attraction due à la communication entre eux, puisqu'on a tendance à regrouper les atomes de distribution les plus communicants, si le critère fait intervenir les coûts de communication).

Nous listons dans ce qui suit quelques contraintes. Ces dernières peuvent être classées en deux classes :

- Les contraintes indépendantes du matériel appelées contraintes logiques,
- Les contraintes dépendantes du matériel appelées contraintes physiques.

### III.4.1 Les contraintes logiques

#### Contraintes d'affectation obligatoire

Parmi les atomes de distribution composant le système d'automatisation, il en existe quelques uns qui ne peuvent s'exécuter que sur des machines ayant des caractéristiques bien définies. La détermination de ces machines permet de réaliser ces affectations obligatoires (nécessaires à l'obtention d'une solution), en disant que tel atome de distribution (ou tel sous ensemble d'atomes) est affecté à telle machine qui est la seule à satisfaire ses besoins. Les affectations obligatoires constitueront un sous ensemble des placements AO= $\{(a_i, m_j)$ tel que $a_i \in A$ et $m_j \in M\}$.

#### Contraintes de regroupement obligatoire

Les contraintes de regroupement obligatoire énumèrent les sous-ensembles d'atomes de distribution qui doivent être implantés sur une même machine. Ainsi l'affectation d'un atome de distribution appartenant à un sous-ensemble donné, à

une machine donnée, impliquera l'affectation du reste des éléments du sous-ensemble à la même machine.

**Contraintes de séparation obligatoire**

Ces contraintes permettent d'éviter le regroupement des atomes de distribution qui ne peuvent pas coexister sur une même machine. Ainsi l'affectation d'un atome de distribution appartenant à un sous-ensemble donné, à une machine donnée, impliquera l'impossibilité du placement du reste des éléments du sous-ensemble sur la même machine.

## III.4.2 Les contraintes physiques

**Contrainte de taille mémoire des machines**

Il est évident que le placement doit satisfaire la contrainte concernant la mémoire disponible sur une machine.

Soit $S_j$ la taille mémoire d'une machine $m_j$ et soit $s_i$ la taille mémoire utilisée par l'atome de distribution $a_i$. La contrainte suivante limite le nombre des atomes de distribution implantables simultanément sur la machine $m_j$.

$$\forall m_j \in M, \sum_{i=1}^{n} x_{ij} s_i \leq S_j$$

**Contrainte de nombre d'atomes de distribution**

Soit $N_j$ le nombre maximum d'atomes de distribution qu'on peut implanter sur une machine $m_j$ (l'ordonnanceur qui ne peut gérer qu'un nombre limité d'atomes de distribution, par exemple). La contrainte suivante limite le nombre des atomes de distribution à implanter sur la machine $m_j$ à $N_j$ atomes de distribution :

$$\forall m_j \in M, \sum_{i=1}^{n} x_{ij} \leq N_j$$

Cette dernière contrainte peut être combinée avec la contrainte d'utilisation totale pour donner :

$$\forall\ m_j \in M,\ 0 < \sum_{i=1}^{n} x_{ij} \leq N_j$$

**Contrainte de flux maximum**

Soit $\wp_{ij}$ la capacité du système de communication reliant deux machines $m_i$ et $m_j$. Cette capacité est supposée invariable.

Soit $V(m_i,m_j)$, le flux de données échangées entre les deux machines. Ce flux ne doit pas excéder la capacité du système de communication. Ainsi, nous avons la contrainte suivante :

$$V(m_i,m_j) \leq \wp_{ij}$$

## III.5 Fonctions coûts et architecture du système d'automatisation

Deux types d'architecture matérielle du système d'automatisation peuvent être considérés : homogène ou hétérogène.

Un système d'automatisation ayant une architecture homogène est un système dont les machines qui le composent sont dotées de puissances de traitement identiques. La fonction coût relative à ce type d'architecture tient compte des coûts qui lient les atomes de distribution entre eux : coûts de communication, coûts d'interférence.

Un système d'automatisation ayant une architecture hétérogène est un système dont les machines qui le composent, sont dotées de puissances de traitement différentes, d'où l'importance de tenir compte des coûts d'exécution des atomes de distribution sur les machines. La fonction coût relative à ce type d'architecture tient compte, en plus des coûts d'exécution des atomes de distribution sur les machines, des coûts qui lient les atomes de distribution entre eux à savoir les coûts de communication, coûts d'interférence.

## III.6 Les algorithmes de répartition

Dans la plupart des modèles proposés, l'architecture fonctionnelle du système d'automatisation peut être représentée à l'aide d'un graphe non orienté G(A, U) dont les sommets symbolisent les atomes de distribution et les arêtes décrivent les liens bidirectionnels de communication entre ces atomes. Les arêtes sont pondérées par les coûts de communication entre atomes de distribution (figure 3.2).

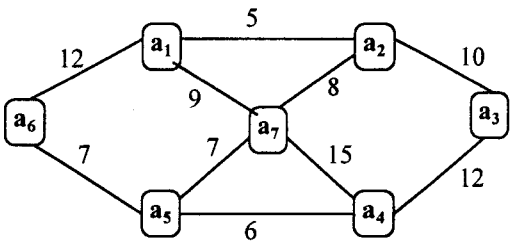



**Figure 3.2**

*Architecture fonctionnelle du système d'automatisation*

Rappelons que l'architecture matérielle est décrite aussi à l'aide d'un graphe connexe et non orienté dont les sommets désignent les machines sur lesquelles seront implantés les atomes de distribution, et les arêtes décrivent les liens physiques entre ces machines (possibilité d'avoir plusieurs réseaux) (figure 3.3). Les arêtes peuvent être pondérées par la capacité du système de communication.

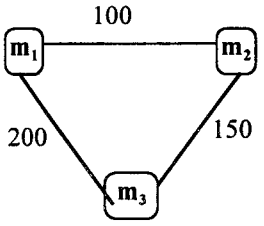



**Figure 3.3**

*Architecture matérielle du système d'automatisation*

A partir de cette modélisation le but est de réaliser un placement optimal des atomes de distribution sur les différentes machines. Ceci passe par la transformation de la représentation initiale (graphes des architectures fonctionnelle et matérielle du système d'automatisation), à une représentation finale qui nous décrit une solution répartie, ou architecture opérationnelle.

Dans ce qui suit nous présentons une classe d'algorithmes issus des méthodes approchées, et deux algorithmes issus des méthodes exactes. Les méthodes approchées peuvent converger vers des solutions optimales à l'aide d'algorithmes tels que les algorithmes gloutons, le recuit simulé qu'on présentera. Nous terminerons par la présentation des algorithmes génétiques qui font partie de nouvelles méthodes prometteuses.

### III.6.1 Les algorithmes itératifs de recherche locale

Un algorithme itératif de recherche locale part d'une configuration ou solution initiale et essaie de l'améliorer [Aho 74]. L'amélioration consiste à effectuer des transformations locales successives de la configuration initiale. Le coût de la configuration générée est évalué. Si le coût de la configuration générée est moindre que celui de la configuration courante, la configuration générée est acceptée et remplace ainsi la configuration courante; sinon, la configuration courante est gardée. Ce processus est réitéré jusqu'à ce qu'aucune transformation locale ne réduise le coût de la configuration courante.

Les algorithmes itératifs de recherche locale garantissent de trouver une solution optimale uniquement dans le cas où la fonction coût est unimodale. Dans le cas contraire, un minimum local est généralement trouvé [Cha 79].

Afin de pouvoir quitter un minimum local pour tenter de converger vers l'optimum global (voir figure 3.4), la technique du recuit simulé a été utilisée dans plusieurs travaux [Ste 85], [Bol 88], [Had 88]. Cette méthode utilise une analogie avec les concepts de la mécanique statistique [Kir 83]. La méthode du recuit simulé a été

appliquée à la plupart des problèmes classiques d'optimisation combinatoire [Laa 87] : le voyageur de commerce, l'affectation quadratique, le partitionnement de graphes, etc., et du point de vue pratique, aux divers problèmes de conception des circuits intégrés tels que [She 87] :

- le placement des composants constituant un circuit : comment placer des composants de tailles très diverses et leurs interconnexions, sur un plan, en minimisant la surface occupée, la longueur des connexions,
- le partitionnement de circuits : étant donné un ensemble de blocs à placer sur deux modules, comment disposer ces blocs de façon à minimiser le nombre de connexions entre les deux modules (critère de localité), tout en gardant à peu près autant de blocs sur un module que sur l'autre (critère de distribution de charge).

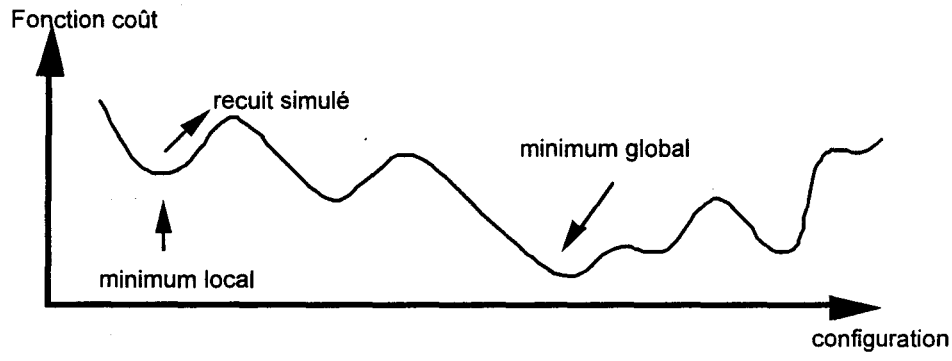



**Figure 3.4**

*L'amélioration itérative conduit à des minima locaux ; le recuit simulé permet de converger vers l'optimum global.*

La configuration initiale est générée aléatoirement, et lui sont associés une température et un état d'énergie élevés. L'énergie d'une configuration est donnée par la fonction coût utilisée, et le but du recuit simulé est de minimiser cette énergie. La configuration est refroidie suivant un ordonnancement donné de la température. A chaque température, un nombre de transformations de la configuration est effectué jusqu'à l'équilibre thermique. Une configuration générée est acceptée suivant une probabilité qui dépend de son coût et de la température courante. L'équilibre thermique

est atteint quand un nombre maximum prédéterminé de transformations ont été acceptées ou générées. Ces choix permettent de générer un nombre suffisant de transformations tout en minimisant le temps passé à hautes températures. La température est mise à jour et le même processus est exécuté à un autre pallier de température. Quand la température décroît, de moins en moins de configurations augmentant l'énergie du système sont acceptées. Le processus du recuit s'arrête quand le système atteint sa température minimale.

## III.6.2 Recherche d'un homomorphisme faible entre graphes

### Définition d'un homorphisme faible entre deux graphes

Soient les deux graphes $G_1(V_1, E_1)$ et $G_2(V_2, E_2)$. Il existe un homorphisme faible entre $G_1$ et $G_2$, s'il existe une projection **H** définie de la manière suivante :

$$\mathbf{H} : V_1 \rightarrow V_2 \text{ tel que } \forall\, s, s' \in V_1,$$
$$(s, s') \in E_1 \text{ et } (\mathbf{H}(s), \mathbf{H}(s')) \in E_2$$

La méthode consiste à trouver un homomorphisme faible optimal entre le graphe des Atomes de distribution G(A, U) et le graphe des machines G(M, E), qui aboutira à la transformation des deux graphes en un seul (figure 3.5). Cette tâche est réalisée dans [She 85] à l'aide de l'algorithme A* et en utilisant le critère minmax. Elle consiste à placer deux Atomes de distribution voisins (communicants) sur deux machines voisines (communicantes). Comme il est en général impossible de garantir que des atomes de distribution communicants seront placés sur des machines communicantes [Mun 90], à chaque noeud du graphe de l'architecture opérationnelle a été associée une boucle [She 85] (voir figure 3.6), ee qui permet de placer les atomes de distribution communicants sur une même machine.

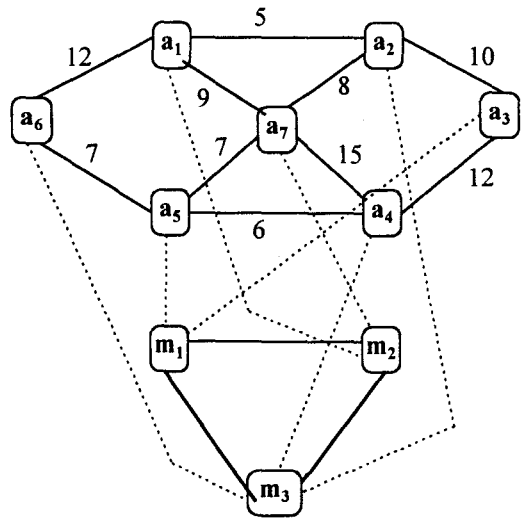



**Figure 3.5**

*Homomorphisme faible entre le graphe des Atomes de distribution et le graphe des machines*

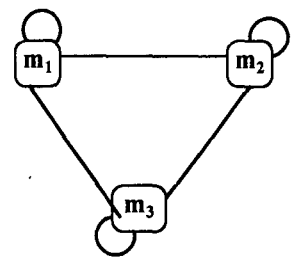



**Figure 3.6**

*Graphe des machines avec boucle*

La méthode présentée ci-dessus (recherche d'un homomorphisme faible optimal), utilise des techniques de l'intelligence artificielle (l'algorithme A*). Elle garantit un placement optimal et un équilibrage de charge des machines. Elle tient compte de trois types de coûts (coûts de communications entre atomes de distribution, coûts de communications entre machines et coûts d'exécution des atomes de distribution sur les machines). Il n'a pas été prouvé que l'algorithme de recherche d'un homomorphisme faible optimal, est NP-complet mais aucun algorithme polynomial n'a été trouvé [Rea 77].

**Exemple [She 85]**

Soit A= $\{a_1, a_2, a_3, a_4\}$ l'ensemble des atomes de distribution et soit M=$\{m_1, m_2, m_3\}$ l'ensemble des machines. La table 3.1 représente les coûts d'exécution, et la figure 3.7 décrit les coûts de communication entre atomes.

| | $m_1$ | $m_2$ | $m_3$ |
|---|---|---|---|
| $a_1$ | **10** | 20 | 30 |
| $a_2$ | 40 | **5** | 10 |
| $a_3$ | 70 | **50** | 80 |
| $a_4$ | 50 | 80 | **20** |

**Table 3.1**

*Coûts d'exécution des atomes de distribution sur les machines.*

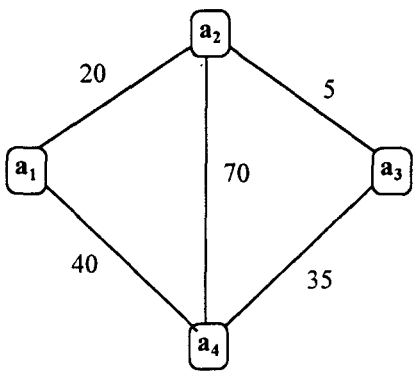



**Figure 3.7**

*Graphe de coûts de communication*

L'homorphisme faible optimal correspond à l'affectation de $a_1$, $a_2$, $a_4$ à la machine $m_3$; et l'atome $a_3$ à la machine $m_2$. La figure 3.8 décrit ces résultats.

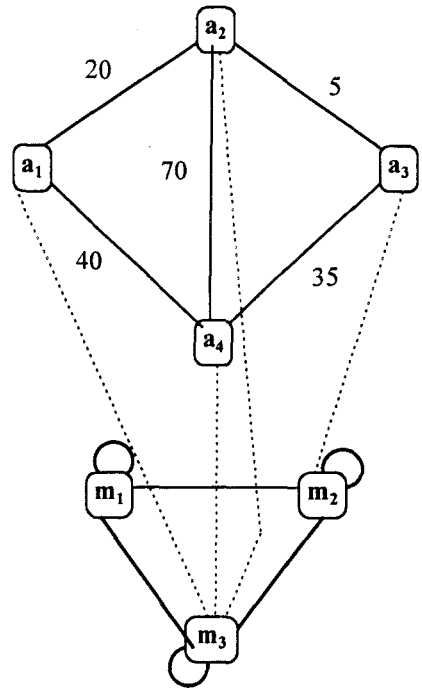



**Figure 3.8**

*Graphe résultant de l'homorphisme faible optimal*

## III.6.3 Partition des coupes minimales d'un graphe

Contrairement à l'approche précédente, celle-ci [Sto 77], [Sto 78], [Lo 88] commence par transformer les deux graphes G(A, U) et G(M, E) en un seul graphe G(AM, UE) dont les sommets représentent les machines et les atomes de distribution, et dont les arêtes représentent d'une part les liens entre atomes de distribution et d'autre part les liens entre atomes de distribution et machines. Afin que le coût total (coûts d'exécution et coûts de communication entre atomes de distribution) soit conservé, les arêtes reliant les atomes de distribution et les machines sont pondérées par un coût moyen d'exécution $w_{iq}$ :

$$w_{iq} = \frac{1}{n-1} \sum_{p \neq q} e(a_i, m_p) - \frac{n-2}{n-1} e(a_i, m_q)$$

A partir de cette transformation, le placement des atomes de distribution est effectué par une partition du graphe G en k coupes minimales à l'aide de l'algorithme

(mincut algorithm), chacune contenant une machine et l'ensemble des atomes de distribution qui lui ont été affectés.

Cette méthode (recherche des coupes minimales) utilise l'algorithme de recherche d'une coupe minimale qui est analogue au problème de recherche d'un flot maximal dans un réseau de transport. Elle tient compte de trois types de coûts (coûts de communications entre atomes de distribution, coûts d'interférence entre atomes de distribution et coûts d'exécution des atomes de distribution sur les machines). Il a été prouvé que l'algorithme utilisé devient NP-complet pour (k >=3) [Sin 87].

**Exemple [Sto 77]**

Soit $A=\{a_1, a_2, a_3, a_4, a_5, a_6\}$ l'ensemble des atomes, et M={m1, m2} l'ensemble des machines. Les coûts d'exécution des atomes sur chacune des machines et les coûts de communication entre atomes sont représentés respectivement dans la table 3.4 et la figure 3.9.

Dans cet exemple (n=2), une arête reliant un processus à un processeur est valuée par le coût d'exécution du processus sur l'autre processeur. En exécutant l'algorithme de coupe minimale sur le graphe, on obtient la coupe délimitée par la ligne en gras (figure 3.10). Ce problème est analogue au problème du flot maximal dans un réseau de transport, pour lequel il existe des algorithmes efficaces [For 62].

| | **m1** | **m2** |
|---|---|---|
| **a1** | **5** | 10 |
| **a2** | 2 | ∞ |
| **a3** | 4 | **4** |
| **a4** | 6 | 3 |
| **a5** | 5 | **2** |
| **a6** | ∞ | 3 |

**Table 3.4**

*Coûts d'exécution des atomes de distribution sur les machines.*

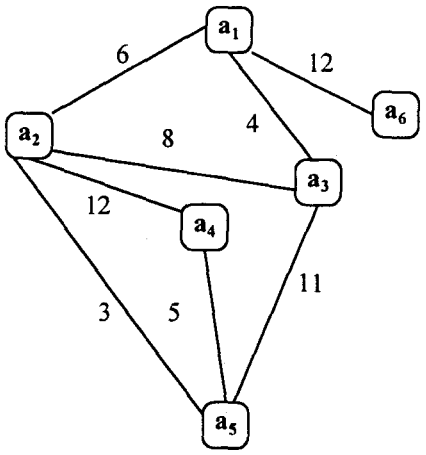



**Figure 3.9**

*Graphe de communication*

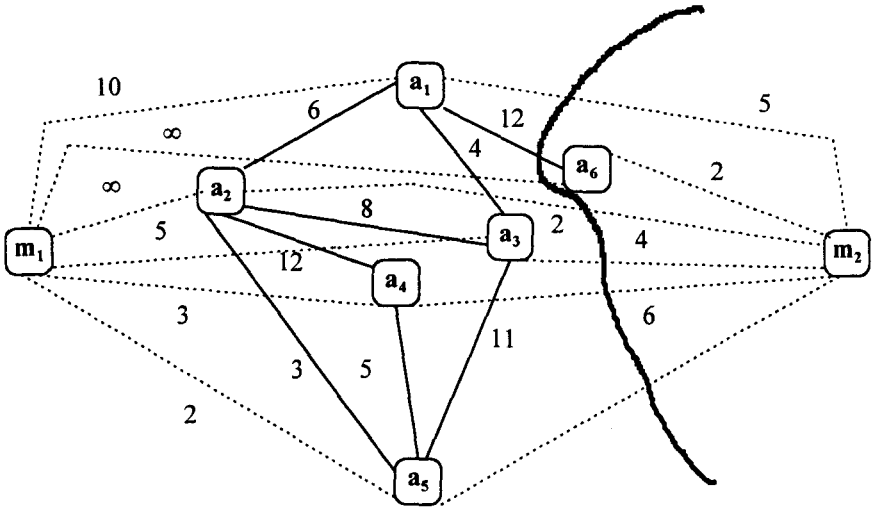



**Figure 3.10**

*Graphe de coupe minimale*

## III.6.5 Les algorithmes génétiques

Les algorithmes génétiques composent une famille très intéressante d'algorithmes d'optimisation. Ils sont basés sur la mécanique de la sélection naturelle. Leur développement avait pour but de modéliser des systèmes adaptatifs naturels et de construire des systèmes artificiels doués des mêmes propriétés. Cette motivation repose

sur la constatation suivante : les mécanismes de l'évolution naturelle ont permis la création de solutions au problème considérable qui est l'adaptation au milieu.

Le principe des algorithmes génétiques est relativement simple [Gol 89].

Soit S un espace de recherche de taille $k^n$ : chaque point de cet espace étant représenté par un vecteur (*individu*) de taille n de k symboles,

Soit f une fonction objectif de S dans R qui associe une valeur réelle à chaque point de S.

Soit un ensemble initial de vecteurs (*individus*) de S appelé la population initiale. Chaque individu représente une solution potentielle.

Une phase dite de *"reproduction"* de cette population est d'abord effectuée : elle consiste à appliquer des opérateurs dits génétiques sur les individus de la population pour générer de nouveaux individus de S. Le principe fondamental des algorithmes génétiques est : "meilleur est un individu, plus sa probabilité de *sélection* est grande". En termes mathématiques, cela veut dire que la probabilité P de sélection croît quand f décroît.

La reproduction a pour but de faire évoluer la population en y propageant les caractéristiques des individus les plus valables qui, suivant la métaphore évolutionniste, sont les plus adaptés.

Une taille constante de la population étant requise, une phase de *"remplacement"* est effectuée. Elle consiste à remplacer les plus mauvais individus de la population courante par les meilleurs individus produits. La taille constante de la population induit donc un phénomène de compétition entre les individus. Le processus génétique est réitéré sur la nouvelle population jusqu'à un certain critère d'arrêt.

Les individus ainsi soumis à une sélection et une reproduction vont s'adapter et trouver une niche écologique qui correspond à un minimum dans l'espace de recherche.

L'algorithme génétique standard est le suivant :

**Algorithme** génétique standard;
Générer aléatoirement une population initiale d'individus.
**Evaluation :** affecter à chaque individu son coût.
***Répéter***

| | |
|---|---|
| **Sélection :** | établir une liste de paires d'individus susceptibles de se reproduire. Le critère de sélection favorisant les meilleurs individus. |
| **Reproduction :** | appliquer les opérateurs génétiques à toutes les paires d'individus sélectionnées. |
| **Evaluation :** | affecter à chaque individu produit son coût. |
| **Remplacement :** | produire la nouvelle population en remplaçant les individus de manière à favoriser encore les meilleurs. |

***Jusqu'à*** ce qu'une "bonne" solution soit trouvée.

Les opérateurs génétiques les plus utilisés durant la reproduction sont le *"crossover"* et la *"mutation"*. Le crossover combine deux chaînes pour produire des chaînes avec des bits associés aux deux. L'opérateur de mutation change aléatoirement l'état des bits d'une chaîne.

Deux paramètres $P_c$ et $P_m$ doivent être définis. Ils représentent respectivement la probabilité d'application des opérateurs de crossover et de mutation. D'autres opérateurs ont aussi été utilisés dans la littérature, dont l'opérateur d'*inversion* [Coh 87] et plusieurs variantes de l'opérateur de crossover conçus pour des problèmes spécifiques [Sha 90]. L'opérateur de crossover classique ne peut être appliqué sans modification à des problèmes d'optimisation combinatoire, où les symboles d'un vecteur représentant une solution possible ne peuvent être dupliqués. Un exemple d'un tel problème d'optimisation est le problème du voyageur de commerce, où l'application de l'opérateur de crossover classique génère des configurations incorectes.

BU LILLE I

Les algorithmes génétiques ont été utilisés avec la condition suivante [Tal 91] : supposons qu'on ait un graphe G de n atomes sur k machines. Chacune des machines est étiquetée par un symbole (par exemple entre 1 et k). Un placement donné de A est représenté par un vecteur de taille n de ces symboles. A l'atome i est associé l'élément i du vecteur dont la valeur sera le numéro de la machine qui lui est assigné (table 3.5).

| **Machine** | 1 | 2 | 3 | 3 | 2 | 4 |
|---|---|---|---|---|---|---|
| **Atomes** | 1 | 2 | 3 | 4 | 5 | 6 |

**Table 3.5**

*Une étape de placement*

L'application de l'opérateur de mutation revient à changer le placement des atomes sur d'autres machines. L'application de l'opérateur de crossover consiste à combiner deux placements donnés.

### III.6.6 Autres approches utilisées

D'autres approches ont été utilisées pour résoudre le problème d'allocation : les chaînes de Markov [Cho 82], la théorie des files d'attente [Bry 81], la théorie de l'évolution [Müh 87], les réseaux de neurones [Ack 87], les réseaux de Pétri [Car 84], la théorie des jeux [Ser 91], etc.

Les méthodes d'allocation présentées ne tiennent pas compte des temps d'attente dus à l'utilisation partagée des liens de communication et des processeurs. Le modèle des files d'attente prenant en compte les délais d'attente dus aux communications a été utilisé dans [Hou 87] pour représenter l'architecture cible.

La plupart des travaux présentés ne tiennent pas compte non plus des contraintes de précédence associées à l'architecture fonctionnelle du système d'automatisation.

La plupart des techniques proposées [Kas 84], [Bap 87], [Shi 90] sont inspirées des heuristiques d'ordonnancement du chemin critique ("critical path") [Cof 76] et du "Highest Levels First with Estimated Times" [Ada 74].

## III.7 Conclusion

Dans ce chapitre, nous avons défini et formalisé le problème de répartition des atomes de distribution. Ensuite nous avons présenté un ensemble de critères, de contraintes et de stratégies permettant la résolution de ce problème.

Les stratégies de répartition présentées sont statiques. Elles sont basées soit sur des algorithmes approchés permettant d'aboutir à des optima locaux : les algorithmes itératifs, soit sur des méthodes exactes : algorithme de recherche d'un homomorphisme faible optimal, de coupe minimale.

La convergence vers des optima globaux pourra se faire à l'aide d'algorithmes prometteurs tel que le recuit simulé : celui-ci, ainsi que le principe des algorithmes génétiques ont été également décrits.

Les algorithmes issus de méthodes exactes ou approchées ont été généralement utilisés pour répartir des systèmes informatiques, et pourront être exploités pour la répartition de l'architecture fonctionnelle du système d'automatisation.

Dans le chapitre suivant, on présentera une nouvelle stratégie basée sur une approche de classification.

Cet exemple montre clairement qu'il n'y a pas de trajectoire en soi mais seulement une trajectoire par rapport à un corps de référence déterminé.

**Albert Einstein**
**"La relativité"**

## Introduction

Le but des méthodes de classification [Ler 70], [Jam 78], [Cel 89], est de regrouper les individus en un nombre fini de classes homogènes. Ces classes sont obtenues au moyen d'algorithmes formalisés, et non pas par des méthodes subjectives ou visuelles faisant appel à l'intuition du praticien. Les méthodes les plus utilisées sont des méthodes basées sur des heuristiques, c'est à dire qu'elles font appel à des algorithmes capables de donner des résultats satisfaisants sans pour autant explorer toutes les solutions possibles.

On distingue en général trois types de méthodes heuristiques basées sur l'approche de classification :

- les méthodes de classification hiérarchiques telles que la méthode ascendante (agrégations successives d'individus, puis de groupes) ou descendante (dichotomies successives),
- les méthodes métriques ou de coalescence telles que la méthode de regroupement autour de centres mobiles [For 65], la méthode de J. B. Mac Queen [Que 67] ou la méthode des nuées dynamiques [Did 72],
- les méthodes statistiques telles que par exemple la méthode des noyaux de Rosenblatt-Parzen (méthode de détection des zones à forte densité dans l'espace des observations) [Par 60], [Pos 87].

Dans ce travail notre intérêt se porte essentiellement sur l'utilisation de la première classe des méthodes décrites ci-dessus, pour la répartition du système d'automatisation. Cette classe de méthodes apparaît adéquate à la structuration hiérarchique (relation de composition) des fonctions du système d'automatisation. En plus cette approche est nouvelle dans notre contexte et n'a pas fait l'objet de travaux de recherches antérieurs, ce qui rend sa modélisation et son implémentation attractives.

Dans ce qui suit, nous proposerons tout d'abord deux algorithmes hiérarchiques de répartition : l'algorithme de regroupement (R1), l'algorithme de décomposition (R2). Ensuite ces deux algorithmes seront combinés pour donner naissance à un algorithme mixte dont la convergence vers une solution répartie optimale localement, sera démontrée.

## IV.1 Indépendance, hiérarchisation des atomes de distribution

De manière évidente, la décomposition du graphe relatif à l'architecture fonctionnelle du système d'automatisation G(A,U) (A est l'ensemble des atomes, et U l'ensemble des arcs ou les arêtes reliant les atomes) en composantes simplement connexes fournit un ensemble de sous-graphes qui correspondent à des sous-applications n'échangeant aucune donnée (en effet, quelle que soit la paire de sommets choisis tels que chacun d'entre eux appartienne à une composante simplement connexe différente, il n'existe aucun arc de l'un à l'autre). Deux composantes simplement connexes du graphe peuvent être affectées indifféremment à la même machine ou à des machines différentes sans que cela n'influe sur la charge réseau. Les sous-ensembles de traitements associés sont indépendants les uns des autres du point de vue logique (une dépendance temporelle pourrait exister, ce point n'ayant pour le moment pas encore fait l'objet de travaux au moins dans notre équipe de recherche).

Une composante simplement connexe du graphe comprend un sous-ensemble d'atomes de distribution tels que chacun d'entre eux est lié à au moins un autre atome de distribution de la même composante. La hiérarchisation des traitements à l'intérieur d'une même composante simplement connexe est le résultat du fait que le graphe est orienté, et définit ainsi un préordre sur l'ensemble des atomes de distribution qui constituent ses sommets.

Notons $a_i > a_j$ l'existence d'un arc entre les atomes de distribution $a_i$ et $a_j$. La décomposition hiérarchique du sous-graphe relatif à une composante simplement connexe fournit une relation d'ordre total associée à un ensemble de composantes fortement connexes. L'exemple suivant illustre cette décomposition.

**Exemple :**

Soit A={$a_1$, $a_2$, $a_3$, $a_4$, $a_5$, $a_6$, $a_7$, $a_8$, $a_9$, $a_{10}$, $a_{11}$}.

Soient le graphe G(A,U) et sa matrice booléenne associée $M_g$ telle que $\forall g_{ij} \in M_g$ on a $g_{ij}=1$ si $a_i$ communique avec $a_j$ 0 sinon. La figure 4.1 nous donne la matrice et le graphe correspondant.

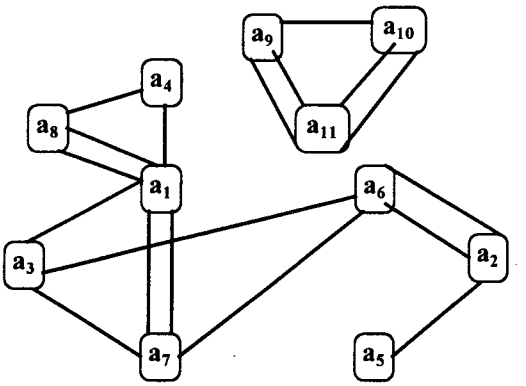



**Figure 4.1 (a)**

*G(A, U)*

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

**Figure 4.1 (b)**

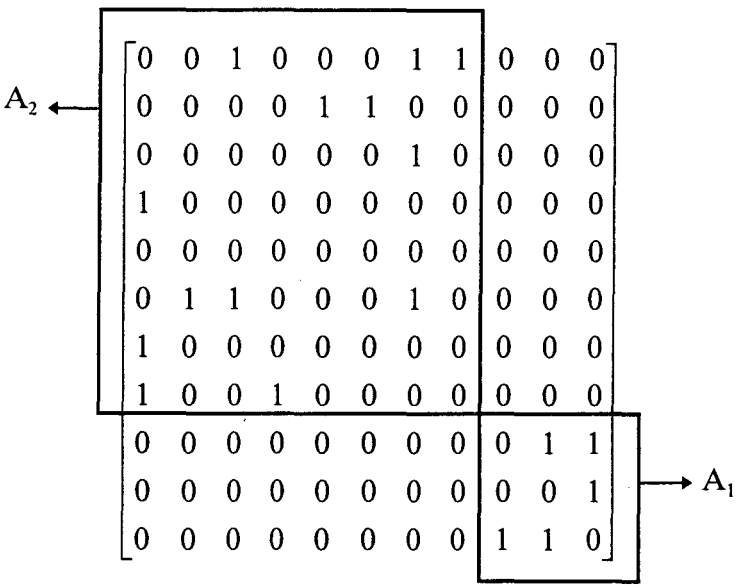
*Matrice $M_g$ associée à G(A, U)*

Cherchons tout d'abord les composantes connexes simples. Les composantes identifiées sont deux sous-systèmes : le sous-système $A_1$ et le sous-système $A_2$ :

$A_1=\{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$,

$A_2=\{a_9, a_{10}, a_{11}\}$.

Dans la figure 4.2, on peut identifier facilement les deux sous-systèmes $A_1$ et $A_2$.




| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

**Figure 4.2**

*Les composantes connexes simples $A_1$ et $A_2$*

Une interprétation de cette décomposition à l'aide de la propriété de connexité consiste à attester que les sous-systèmes identifiés sont indépendants, c'est à dire non communiquants à priori, et que leur répartition peut s'effectuer indépendamment ce qui simplifierait le problème de répartition. Il est clair que chacun de ces sous-systèmes peut disposer de sa propre architecture matérielle, comme il est possible d'en partager une avec un ou d'autres sous-systèmes.

Cherchons les composantes connexes fortes dans le graphe orienté associé à G(A, U) (figure 3). Les composantes identifiées sont les sous-systèmes : $A_{11}$, $A_{12}$, $A_{13}$, $A_{14}$, $A_{15}$, $A_{16}$, $A_{21}$, $A_{22}$, et $A_{23}$.

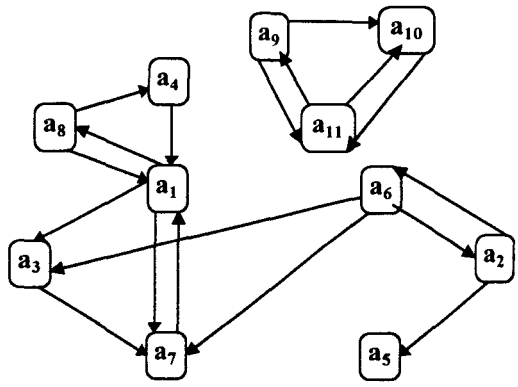



**Figure 4.3**

*Graphe orienté associé à G(A, U)*

$A_{11}=\{a_1, a_7\}$

$A_{12}=\{a_1, a_3, a_7\}$

$A_{13}=\{a_1, a_7, a_8\}$

$A_{14}=\{a_1, a_3, a_7, a_8\}$

$A_{15}=\{a_1, a_3, a_4, a_7, a_8\}$

$A_{16}=\{a_2, a_6\}$

$A_{21}=\{a_{10}, a_{11}\}$

$A_{22}=\{a_9, a_{11}\}$

$A_{23}=\{a_9, a_{10}, a_{11}\}$

Cette décomposition est plus fine que celle effectuée à l'aide de la connexité simple. En effet celle-ci nous donne généralement un nombre plus important de sous-systèmes et peut aussi servir pour la détermination d'une solution initiale au problème de répartition.

## IV.2 Répartition du système d'automatisation

L'objectif est de partitionner l'architecture fonctionnelle composée essentiellement de l'ensemble des atomes de distribution A= $\{a_1, a_2, ..., a_n\}$ en k sous-ensembles $\{z_1, z_2, ..., z_k\}$. Chacun de ces sous-ensembles sera implanté dans une unité de traitement ou machine (capteur, actionneur, automate programmable, ...) (voir figure 4.4). L'ensemble des unités de traitement est noté M = $\{m_1, m_2, ...,m_k\}$. Ces unités de traitement sont reliées par un système de communication.

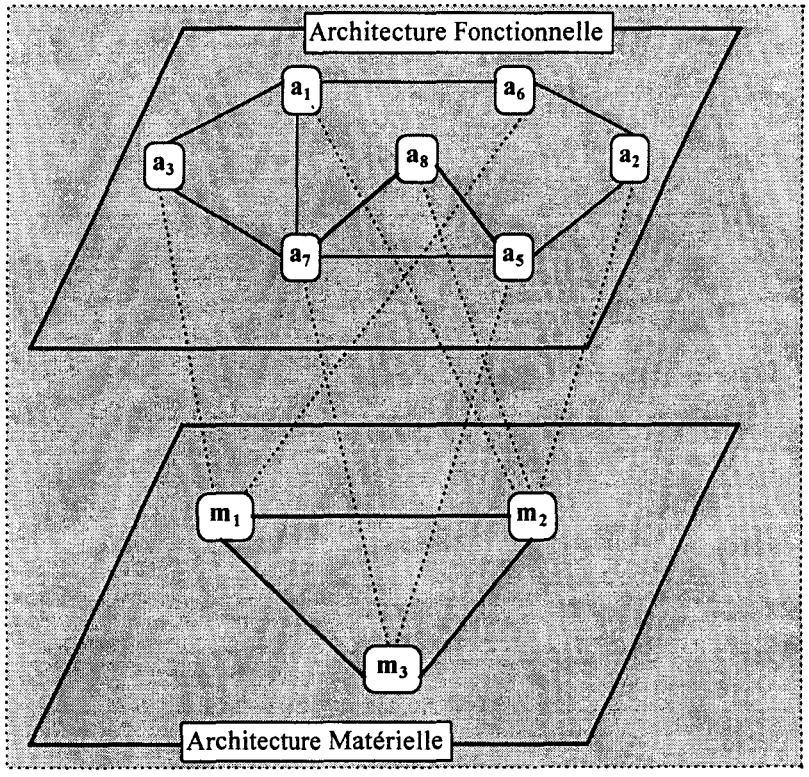



**Figure 4.4**

*Projection de l'architecture fonctionnelle sur l'architecture matérielle*

Soit F(A) l'ensemble des parties de A. On dit que $Z=\{z_1, z_2, ..., z_k\}$ est une partition en k sous-ensembles de A si et seulement si :

- $z_i \in F(A)$ et $z_i \neq \varnothing$ pour i : 1 ... k,

- $\forall i \neq j, z_i \cap z_j = \varnothing$,
- $\bigcup_{i:1..k} z_i = A$.

## IV.2.1 Les inf-demi-treillis

Soit W(A) l'ensemble de toutes les partitions de A. Nous établissons dans cet ensemble la relation <= : **"être plus fine que"**, définie de la manière suivante:

$$\text{Si } S, Q \in W(A), S <= Q \Leftrightarrow \forall s \in S, \exists q \in Q / s \subseteq q$$

Par ailleurs, si S <= Q, chaque classe q est l'union des classes s qu'elle contient :

$$\forall q \in Q, q = \bigcup_{si \subseteq qj} s$$

Ainsi, Q détermine une partition des classes de S.

La relation <= est un ordre partiel, c'est à dire qu'elle vérifie les trois conditions suivantes :

- elle est réflexive : $\forall Z \in W(A)$ on a $Z <= Z$,
- elle est antisymétrique : $S <= Q$ et $Q <= S \Rightarrow S = Q$,
- elle est transitive : $S <= Z$ et $Z <= Q \Rightarrow S <= Q$.

W(A) muni de la relation <= constitue un treillis (voir figure 4.5): n'importe quel couple S, Q d'éléments de W(A) possède un ***infimum*** (noté $S \cap Q$) et un ***supremum*** (noté $S \cup Q$).

L'infimum est défini ainsi :

$S \cap Q = \{ s \cap q, s \in S, q \in Q$ telle que $s \cap q \neq \varnothing \}$.

Pour l'obtention de $S \cup Q$, plus compliquée à décrire, voir [Ler 76].

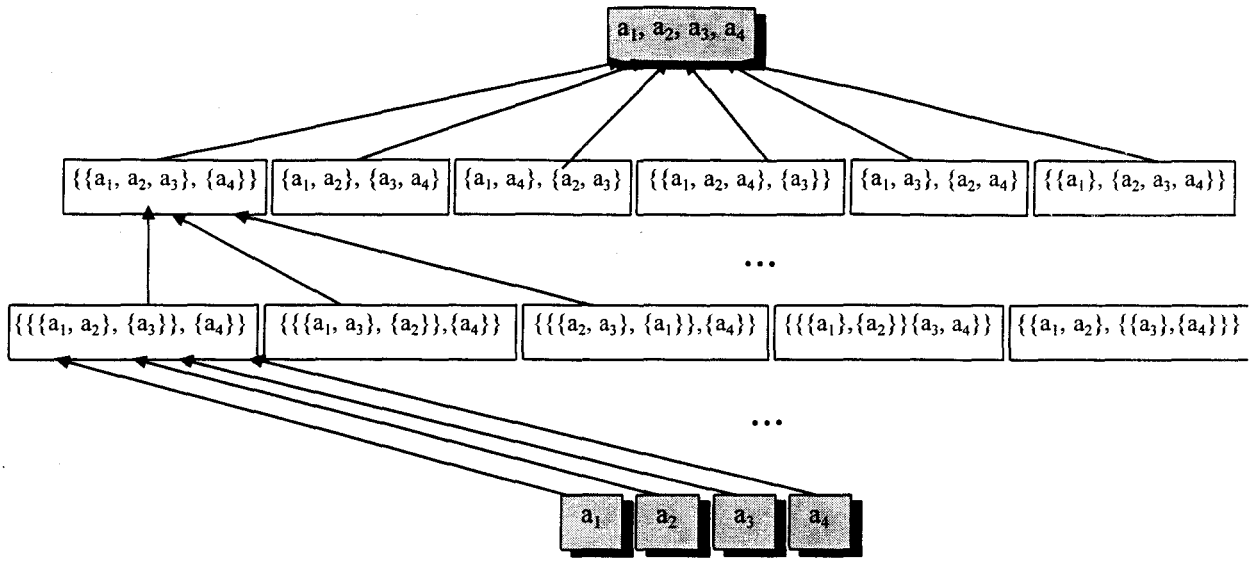



**Figure 4..5**

*Treillis de l'application A={a1, a2, a3, a4}*

**Propriétés de l'infimum**

$\forall$ S, Q, Z $\in$ W(A)

- $(S \cap Q) <= S$
- $(S \cap Q) <= Q$
- $Z <= S$ et $Z <= Q \Rightarrow Z <= S \cap Q$

**Propriétés du supremum**

$\forall$ S, Q, Z $\in$ W(A)

- $S <= (S \cup Q)$
- $Q <= (S \cup Q)$
- $S <= Z$ et $Q <= Z \Rightarrow (S \cup Q) <= Z$

Les relations entre <= et les opérations $\cap$ et $\cup$ peuvent aussi être établies par l'équivalence suivante [Bir 64] :

$P <= Q \Leftrightarrow P \cap Q = P \Leftrightarrow P \cup Q = Q.$

**Proposition**

L'ensemble W(A) de toutes les partitions de A, muni de l'opération $\cap$ :

$\cap : W(A) \times W(A) \rightarrow W(A)$

$(S, Q) \rightarrow S \cap Q$

constitue un inf-demi-treillis, c'est à dire, vérifie les conditions suivantes, $\forall$ Z, Q, $S \in W(A)$ :

- idempotence : $Z \cap Z = Z$,
- commutativité : $S \cap Q = Q \cap S$,
- associativité : $Z \cap (Q \cap S) = (Z \cap Q) \cap S$.

## IV.2.2 Les sup-demi-treillis

Dans l'ensemble F(A), la relation d'inclusion $\subseteq$ définit un ordre partiel :

$\forall$ X, Y, I $\in$ F(A),

- réflexivité : $X \subseteq X$,
- antisymétrie : $X \subseteq Y$ et $Y \subseteq X \Rightarrow X=Y$,
- transitivité : $X \subseteq I$ et $I \subseteq Y \Rightarrow X \subseteq Y$.

De plus pour tout couple X, Y éléments de F(A), il existe un infimum qui correspond à l'intersection $X \cap Y$ et un supremum qui correspond à la réunion $X \cup Y$ tels que :

**Propriétés de l'infimum**

$\forall$ X, Y, I $\in$ F(A),

- $(X \cap Y) \subseteq X$
- $(X \cap Y) \subseteq Y$

- $I \subseteq X$ et $I \subseteq Y \Rightarrow I \subseteq (X \cap Y)$

**Propriétés du supremum**

$\forall X, Y, I \in F(A)$,

- $X \subseteq (X \cup Y)$
- $Y \subseteq (X \cup Y)$
- $X \subseteq I$ et $Y \subseteq I \Rightarrow (X \cup Y) \subseteq I$

La relation entre $\subseteq$, $\cup$ et $\cap$ est donnée par :

$X \subseteq Y \Leftrightarrow (X \cup Y) = Y \Leftrightarrow (X \cap Y) = X$

Ainsi l'ensemble des parties F(A) muni de la relation $\subseteq$ et les opérateurs $\cup$ et $\cap$ devient un treillis ou plus exactement une algèbre de boole [Bir 64] . Mais nous retenons pour les besoins de ce travail seulement une structure moins riche, celle de sup-demi-treillis.

**Proposition**

L'ensemble F(A) muni de l'opérateur $\cup$ défini de la manière suivante :

$$\cup : F(A) \times F(A) \rightarrow F(A)$$
$$(X, Y) \rightarrow (X \cup Y)$$

constitue un sup-demi-treillis, c'est à dire qu'il vérifie les conditions suivantes :

- idempotence : $X \cup X = X$,
- commutativité : $X \cup Y = Y \cup X$,
- associativité : $X \cup (I \cup Y) = (X \cup I) \cup Y$.

**Démonstration**

C'est une conséquence directe des propriétés élémentaires de l'opérateur union.

W(A) et <= constitue un treillis et $W_k(A)$ est l'ensemble de toutes les partitions en k classes qui correspondent au niveau k du treillis. Notons que $W_k(A) \subseteq W(A)$.

## IV.3 Détermination d'une fonction coût

### IV.3.1 Définition de la fonction coût

Dans l'objectif de distribuer les atomes constituant l'ensemble A sur les machines de M, on a présenté dans le chapitre III, une fonction V, qui évalue le coût de communication entre atomes de distribution. Cette fonction peut être généralisée pour estimer les coûts de communication entre les portions d'application. L'objectif est donc de trouver une solution répartie du système en question qui minimise la fonction coût du trafic total V en tenant compte d'un ensemble de contraintes.

Deux sous applications sont en relation signifie qu'une partie des données produites par les atomes de distribution appartenant à l'une d'elles, sont consommées par certains des atomes de distribution appartenant à l'autre.

Soit $z_i$, $z_j$ deux parties de A. Soit $P(z_i)$ et $P(z_j)$ les ensembles de données produites par $z_i$ et $z_j$. Soit $C(z_i)$ et $C(z_j)$ les ensembles des données consommées par $z_i$ et $z_j$.

$\varphi(z_i, z_j)=P(z_i) \cap C(z_j) \subset X$ constitue les données produites par les atomes de distribution de $z_i$ et consommées par les atomes de distribution de $z_j$. Rappelons que X

est l'ensemble des données manipulées par les atomes de distribution du système d'automatisation.

Il est maintenant facile d'évaluer le coût de communication entre deux sous-applications. Celui-ci dépend du volume des données échangées entre elles.

L'ensemble des données échangées entre deux sous-applications $z_i$ et $z_j$ est :

$$E(z_i,z_j)= \varphi(z_i, z_j) \cup \varphi(z_j, z_i)$$

Le coût engendré par les données échangées entre elles est :

$$V(z_i,z_j)= \sigma(E(z_i,z_j))$$

Ce coût est aussi égal à la somme des coûts des échanges entre les atomes de distribution de $z_i$ et les atomes de distribution de $z_j$ :

$$\sum_{a_i \in z_i} \sum_{a_j \in z_j} V(a_i, a_j)$$

rappelons que $\sigma$ dépend des types données échangées, du protocole réseau, des fréquences des échanges.

La valeur $V(Z)=V(z_1, z_2, ..., z_k)$ estime le coût de communication du système qui dépend des coûts de communication entre les portions d'applications qui le composent :

$$V(z_1, z_2, ..., z_k)= \sum_{z_i \in Z} \sum_{z_j \in Z \setminus z_i} V(z_i, z_j)$$

**Discussions**

- La fonction $V(z_i,z_j)$ évalue la charge de communication entre les deux unités de traitement $m_i$ et $m_j$ en utilisant un modèle fondé sur l'expression des besoins des sous applications confiées à ces machines. Ce modèle prend en compte, en particulier :

- les types des variables échangées, qui se traduisent par des longueurs différentes,
- les protocoles du réseau, qui définissent le format des trames échangées, afin de tenir compte, dans le calcul de la charge, de la transmission des messages, trames et bits de service,
- le regroupement de messages à l'intérieur d'une même trame, dans le respect du protocole du réseau, dans le but d'optimiser le rendement de transmission.

- La fonction $V(z_i,z_j)$ n'exprime que la charge du système de communication liée aux échanges de données entre les machines $m_i$ et $m_j$, sans tenir compte des aspects temporels relatifs à l'exécution des sous applications $z_i$ et $z_j$. Ces aspects temporels peuvent être considérés d'un double point de vue :
  - l'exécution des sous applications $z_i$ et $z_j$ peut nécessiter, en plus de l'échange des données produites par l'une et consommées par l'autre, l'échange de messages de synchronisation, destinés à garantir une cohérence temporelle d'exécution. Ces messages de synchronisation doivent bien entendu entrer dans le calcul de la charge du système de communication,
  - le transport des données par le système de communication n'est pas instantané. Il convient donc de le caractériser non seulement par son débit, comme nous l'avons fait plus haut, mais aussi par les délais qu'il introduit dans l'acheminement des variables. Ces délais sont liés au type et au protocole du réseau, et influent sur la fraîcheur des variables qu'un atome de distribution consommateur reçoit par l'intermédiaire de celui-ci. Rappelons que chaque variable consommée par un atome de distribution est caractérisée par un délai de péremption, au delà duquel elle devient impropre à la consommation par l'atome de distribution en question.
- La définition des sous-applications que nous avons donnée est associée au sous-ensemble des atomes de distribution de l'application globale confiés à l'une des unités de traitement de l'architecture distribuée. Cette définition peut être généralisée en considérant des groupes d'unités de traitement. Une sous

application est alors définie par l'ensemble des atomes de distribution confiés à un sous-ensemble de l'ensemble des unités de traitement de l'architecture distribuée. L'intérêt de cette généralisation est qu'elle introduit la possibilité de considérer des sous-ensembles de machines caractérisés par une propriété commune de type fonctionnel, géographique, architectural, ... Des exemples sont donnés par :

- le sous-ensemble des unités de traitement prenant en charge la commande (ou une partie de la commande) d'un équipement élémentaire,
- le sous ensemble des unités de traitement situées dans une salle donnée,
- le sous-ensemble des unités de traitement connectées au même tronçon de réseau. Ce dernier exemple est particulièrement intéressant car il offre la possibilité de considérer des architectures distribuées autour de plusieurs tronçons de réseau, chacun d'entre eux pouvant, de plus, être de type différent. Les charges et délais de transmission entre machines connectées au même tronçon de réseau ont été discutés plus haut. Pour ce qui concerne les échanges entre unités de traitement connectées à des tronçons de réseau différents, il convient de tenir compte des délais supplémentaires introduits, dans chaque sens, par les mécanismes de passage d'un tronçon à l'autre. La prise en compte de sous-ensembles de machines connectées au même tronçon de réseau offre la possibilité d'aborder le problème de la distribution des traitements de manière hiérarchisée.

### IV.3.2 Les contraintes

Il existe une multitude de contraintes qui influent sur la solution finale du problème de répartition du système d'automatisation. Nous avons retenu celles relatives aux charges machines pour illustrer la façon dont les algorithmes que nous proposons tiennent compte des contraintes. La contrainte de charge présentée dans ce chapitre,

suppose que la charge d'une machine dépend essentiellement du nombre de données chargées dans la mémoire de celle-ci. Le modèle suivant est développé :

Nous supposons que la mémoire de chaque machine $m_i$ est capable de supporter un certain nombre de données $d_i$. Pour mesurer la capacité nécessaire de la machine $U(m_i)$ provoquée par l'implémentation d'un ensemble d'atomes de distribution, nous devons simplement calculer la quantité de données produites et consommées par ces atomes :

$$U(m_i) =. |P(m_i) \cup C(m_i)|$$

La contrainte de capacité mémoire est décrite de la manière suivante :

$$U(m_i) <= d_i.$$

## IV.4 L'approche de classification

Rappelons qu'un nombre important d'approches ont été proposées pour la résolution des problèmes de répartition des atomes de distribution. Elles sont généralement basées sur des techniques de la théorie de graphes, des techniques de programmation mathématique, des techniques heuristiques. Ces approches supposent que les machines composant le système sont soit homogènes, c'est à dire que les coûts d'exécution sont négligeables, soit hétérogènes, c'est à dire qu'on doit tenir compte des coûts d'exécution sur les machines, mais ignorent les contraintes logiques (les regroupements obligatoires, ...) et les contraintes physiques (des affectations obligatoires, de charge, ...), ce qui est primordial pour les systèmes d'automatisation.

Dans l'objectif de trouver une partition admissible qui minimise la fonction coût du trafic en tenant compte d'un ensemble de contraintes, nous formulons le problème de répartition comme un problème de classification et nous proposons une méthode mixte qui associe des opérateurs de classification hiérarchique : l'opérateur d'agrégation **R1** et l'opérateur de décomposition **R2**. L'opérateur d'agrégation regroupe deux classes dont le coût des flux de données échangées est maximal, tandis que l'opérateur de

décomposition sépare une classe en deux sous-classes dont le coût de communication est minimal. Supposons que k est le nombre de machines de l'architecture répartie, nous montrons que l'algorithme mixte converge vers une solution optimale localement dans $W_k(A)$.

### IV.4.1 L'algorithme ascendant (R1)

L'algorithme ascendant R1 est un algorithme itératif qui démarre avec une partition de k classes $Z=\{z_1, z_2, ..., z_k\}$. Le nombre de classes de la solution initiale dépend du placement qui précède le démarrage de l'algorithme. Il résulte des contraintes d'affectation obligatoire, de regroupement obligatoire, et de séparation obligatoire. Le nombre maximum de classes peut être égal au cardinal de A ( $|A|$) si on démarre avec la partition atomique $Z=\{\{a_1\}, \{a_2\}, ..., \{a_n\}\}$.

Dans chaque itération l'algorithme R1 se déplace d'un niveau du treillis vers le suivant en réalisant une procédure d'agrégation qui consiste à regrouper, sur une même machine, les deux portions d'application caractérisées par le coût de communication le plus important. Il est clair que du moment où les deux portions d'applications sont unifiées sur la même machine, la charge du système de communication diminue, et la nouvelle charge peut être facilement recalculée. De plus, connaissant les caractéristiques relatives aux portions d'applications, il est possible d'évaluer la charge de la machine qui les supporte. A chaque procédure d'agrégation les contraintes peuvent être violées, par exemple, la charge de la machine choisie augmente et peut dépasser celle autorisée. Il est obligatoire alors d'inclure un mécanisme qui permet d'assurer le respect des contraintes, par exemple, un mécanisme qui certifie le respect de la charge maximale de la machine concernée. Ce mécanisme est facile à définir : quant on passe d'un niveau du treillis à un autre, seulement les regroupements conduisant à une charge machine acceptable sont autorisés. L'algorithme s'arrête quand la procédure d'agrégation n'est plus réalisable sauf en violant les contraintes de charge.

L'algorithme R1 est résumé par l'application suivante :

R1 : $W_k(A) \rightarrow W_{k-1}(A)$

$Z_k \rightarrow R1(Z_k)=Z_k \setminus \{z_\alpha, z_\beta\} \cup \{z_\alpha \cup z_\beta\}$, k : 2 ... n-1, $z_\alpha, z_\beta \in Z_k$ ,

telles que :

$V(z_\alpha, z_\beta)= \max_{i,j}(V(z_i, z_j))$ , i, j : 1 ... k.

Nous aurons alors : $V(R1(Z_k))=V(Z_k)-V(z_\alpha, z_\beta)$

**Algorithme** R1;

**Début**

k:=n-1;

**Répéter**

$Z_k := Z_k \setminus \{z_\alpha, z_\beta\} \cup \{z_\alpha \cup z_\beta\}$ **telle que** $V(z_\alpha, z_\beta)= \max_{i,j}(V(z_i, z_j))$

$V(R1(Z_k)):=V(Z_k)-V(z_\alpha, z_\beta)$

k:=k-1

**Jusqu'à** k=2

**Fin**

## Complexité de l'algorithme R1

L'algorithme R1 n'est évidement pas optimal. On peut seulement être sûr que l'algorithme trouvera les optimums locaux $Z^*_1$ et $Z^*_n$ appelés partitions triviales, ainsi que $Z^*_{n-1}$ puisque dans ces niveaux il aura examiné toutes les partitions. Pour les autres niveaux, n-2, n-3, ..., 2 on ne peut rien dire puisque l'algorithme n'aura examiné que les partitions qui lui sont accessibles , c'est à dire celles obtenues par regroupement de deux classes de la partition précédente.

Pour un niveau k donné , l'algorithme R1 examine tous les couples de sous-systèmes de la partition $Z_k$, pour trouver les deux sous-systèmes $z_\alpha$ et $z_\beta$ les plus liés. Cela est équivalent à l'examen de toutes les partitions du niveau k-1 moins fines que $Z_k$, pour trouver celle qui minimise V.

On peut dire alors que le passage du niveau k au niveau k-1 fait l'examen d'un nombre de partitions égal au nombre de couples de sous-systèmes de la précédente, c'est à dire :

$$\frac{k(k-1)}{2} \tag{4.9}$$

Au total donc, le nombre de partitions examinées par l'algorithme R1 est :

$$\begin{aligned}
&\sum_{k=n}^{2} \frac{k(k-1)}{2} \\
&= \frac{1}{2} \sum_{k=1}^{n} (k^2 - k) \\
&= \frac{1}{2} \sum_{k=1}^{n} k^2 - \frac{1}{2} \sum_{k=1}^{n} k \\
&= \frac{1}{2} \left( \frac{n(n+1)(2n+1)}{6} \right) - \frac{1}{2} \left( \frac{n(n+1)}{2} \right) \\
&= \frac{n^3 - n}{6}
\end{aligned}$$

d'ou on déduit la complexité de l'algorithme R1 qui est estimée à $n^3$.

## IV.4.2 L'algorithme de décomposition (R2)

L'algorithme de décomposition R2 est comparable au précédent, mais il procède par une construction hiérarchique opposée. Il procède par des séparations successives en deux classes engendrant le coût de communication le moins important, comparé aux autres décompositions possibles du même niveau. Il démarre généralement par la partition contenant une seule classe. Le nombre de classes dépend aussi de la répartition initiale résultante des contraintes initiales variées (affectation obligatoire, regroupement obligatoire, séparation obligatoire, ...). En cas d'absence de contraintes à appliquer avant le démarrage de l'algorithme, le nombre de classes est égal à un si on commence par la solution triviale Z={A}. Cela veut dire qu'au départ les atomes de distribution sont implémentés sur une seule machine.

En démarrant l'algorithme avec des classes contenant un nombre important d'atomes de distribution, les contraintes, telles que les contraintes de charge, les contraintes d'affectation et de séparation obligatoires, ... peuvent être violées. Dans un sens la progression de la procédure de décomposition, continue, tant que cela est possible, jusqu'à assurer le respect total des contraintes.

Le nombre des solutions explorées par l'algorithme de décomposition R2 est plus important que celles explorées par l'algorithme d'agrégation R1, puisque pour trouver le meilleur découpage d'une classe à p éléments en deux sous-classes, nous devons explorer $(2^{p-1}-2)$ possibilités. Cette complexité autorise une solution meilleure mais non optimale, quand le volume de calcul est admissible.

L'algorithme de décomposition peut être résumé par l'application suivante:

$R2 : W_k(A) \rightarrow W_{k+1}(A)$

$Z_k \rightarrow R2(Z_k)=(Z_k \setminus \{z_\alpha\}) \cup \{z_\alpha \setminus y^*, y^*\}$, $k : 1 \ldots n-1$, $z_\alpha \in Z_k$, $y^* \subset z_\alpha$, telle que :

$V(z_\alpha \setminus y^*, y^*) <= V(z_i \setminus y, y)$, $\forall z_i \in Z_k$, $\forall y \subset z_i$ $i : 2 \ldots k$. Nous aurons alors :

$V(R2(Z_k))=V(Z_k)+V(z_\alpha \setminus y^*, y^*)$.

**Algorithme** R2;

**Début**

k:=1;

**Répéter**

$Z_k := Z_k \setminus \{z_\alpha\}) \cup \{z_\alpha \setminus y^*, y^*\}$, **telle que** $V(z_\alpha \setminus y^*, y^*) <= V(z_i \setminus y, y)$, $\forall z_i \in Z_k$, $\forall y \subset z_i$

$V(R2(Z_k))=V(Z_k)+V(z_\alpha \setminus y^*, y^*)$.

k:=k+1

**Jusqu'à** k=n-1

**Fin**

## Complexité de l'algorithme R2

Pas plus que l'algorithme R1, l'algorithme R2 n'est optimal. Les seuls optimums locaux trouvés par R2 sont $Z^*_1$ et $Z^*_n$, ainsi que $Z^*_2$ puisque dans ces niveaux il aura examiné toutes les partitions. Pour les autres niveaux, 3, 4, ..., n-1, on ne peut rien dire puisque l'algorithme n'aura examiné que les partitions qui lui sont accessibles, c'est à dire celles obtenues par l'éclatement en deux d'un des sous-systèmes de la partition précédente.

Toutefois, l'algorithme R2 a des bonnes chances de trouver des partitions dont le coût de communication est petit. Sans que ceci prétende être une démonstration, imaginons qu'il existe une partition optimale $Z^*_k$ au niveau k pour laquelle $V(Z^*_k)=\varepsilon$ est petite. Alors toute partition moins fine que $Z^*_k$ aura un coût de communication égal ou supérieur à $\varepsilon$. Ces partitions forment des chaînes entre $Z^*_1$ et $Z^*_k$ qui traversent tous les niveaux entre $Z^*_1$ et $Z^*_k$. Alors sous l'hypothèse d'un $\varepsilon$ petit, on pourrait espérer que l'algorithme descende par une de ces chaînes jusqu'à $Z^*_k$, puisque dès le niveau 2 et à chacun des niveaux suivants il trouvera des partitions dont le coût de communication engendré est inférieur ou égal à $\varepsilon$.

Ces "chances accrues" d'optimalité se reflètent dans un temps de calcul bien plus élevé que celui de l'algorithme R1.

A chaque niveau l'algorithme R2 examine toutes les décompositions en deux de chacun des sous-systèmes. Cela est équivalent à examiner toutes les partitions obtenues par l'éclatement en deux d'un des sous-systèmes de la partition précédente.

A partir du niveau 2, il suffit d'examiner uniquement les partitions en deux des sous-systèmes créés dans l'étape antérieure.

Soit $z_1$ et $z_2$ ces sous-systèmes. Le nombre d'alternatives examinées est alors

$$f(|z_1|,|z_2|)=\frac{2^{|z_1|}-2}{2}+\frac{2^{|z_2|}-2}{2}$$

En effet, pour $z_1$ il faut tester toutes les partitions de la forme $\{z_1\backslash x, x\}$, avec $x \subseteq z_1$ et $x \neq \varnothing$. Il y a $2^{|z_1|}$ sous ensembles non vides de $z_1$, cette dernière n'étant pas prise en compte; la division par deux est due au fait que $V(z_1\backslash x,x)=V(x, z_1\backslash x)$. Pour $z_2$ le raisonnement est le même.

Au niveau k, il est facile de prouver que $|z_1|+|z_2| < n-(k-2)$ puisqu'il faut que chacun des autres k-2 sous-systèmes aient au moins un atome de distribution. Alors, vu la nature exponentielle de f, on peut dire que :

$$f(|z_1|,|z_2|)<f(n-(k-2)-1,1)$$
$$=2^{n-k-1}. \qquad (4.10)$$

Ainsi, le nombre de partitions examinées pour passer du niveau $k > 2$ au niveau $k+1$ est au maximum $2^{n-k-1}$. Par ailleurs, pour passer du niveau 1 au niveau 2, il faut tester $(2^n-2)/2=2^{n-1}-1$ partitions. En faisant la somme de k=1 à k=n-1 on voit alors que la complexité de R2 est inférieure à :

$$
\begin{aligned}
&(2^{n-1}-1)+\sum_{k=2}^{n-1}(2^{n-k}-1)\\
&=\sum_{k=1}^{n-1}(2^{n-k}-1)\\
&=\sum_{k=1}^{n-1}2^{n-k}-(n-1)\\
&=\sum_{j=1}^{n-1}2^{j}-(n-1)\\
&=\frac{2^{n}-2}{2-1}-(n-1)\\
&=2^{n}-2-n+1\\
&=2^{n}-n-1
\end{aligned}
$$

d'ou on déduit la complexité de l'algorithme R2 qui est estimée à $2^n$.

Il faut remarquer que ce résultat ne constitue qu'une borne supérieure de la complexité de R2, correspondant au pire des cas.

### IV.4.3 L'algorithme mixte

L'algorithme mixte suppose l'existence d'une solution initiale composée de k classes. L'algorithme ne changera pas le nombre de classes mais la répartition fournie par la solution initiale. Il procède par une combinaison successive des deux opérateurs **R1** et **R2**. L'opérateur d'agrégation R1 fournit une partition de (k-1) classes sur laquelle il suffit d'appliquer l'opérateur **R2** pour obtenir le nombre de classes initial k et vice versa. Nous prouverons que la nouvelle partition en k classes est meilleure que la précédente, et donc le point fixe de la procédure itérative est un optimum local de la fonction critère V.

La composition des opérateurs R1 et R2 permet de définir les opérateurs suivants :

$R1oR2 : W_k(A) \rightarrow W_k(A)$

$Z_k \rightarrow R1oR2(Z_k)=R1(R2(Z_k))$

R2oR1 : $W_k(A) \rightarrow W_k(A)$

$Z_k \rightarrow$ R2oR1($Z_k$)=R2(R1($Z_k$)) pour k : 2 ... n-1

**Algorithme** Mixte ($Z_k$);

**Début**

**Répéter**

**Répéter**

Z:=R1oR2($Z_k$)

**Jusqu'à** R1oR2($Z_k$)= $Z_k$

**Répéter**

$Z_k$:=R2oR1($Z_k$)

**Jusqu'à** R2oR1($Z_k$)= $Z_k$

**Jusqu'à** R1oR2($Z_k$)= R2oR1($Z_k$)= $Z_k$

**Fin**

**Proposition (i)**

Soit Z une partition du niveau k, k : 2 ... n-1, nous avons :

**(i1)** V(R1oR2(Z)) <= V(Z),

**(i2)** V(R2oR1(Z)) <= V(Z).

**Démonstration**

**Preuve de (i1)**

R1(Z)=Z\{$z_p$, $z_q$}$\cup$ {$z_p \cup z_q$} avec V($z_p$,$z_q$) > V($z_i$,$z_j$), $\forall$ $z_i$, $z_j \in$ Z, V(R1(Z))=V(Z)- V($z_p$,$z_q$).

Calculons R2(R1(Z)) :

$R1(Z)= \{z_1, z_2, ..., z_{k-1}\}$, $R2(R1(Z))=R1(Z) \setminus \{z_\alpha\} \cup \{z_\alpha \setminus y^*, y^*\}$, $k : 2 \dots n$, $z_\alpha \in Z$, $y^* \subset z_\alpha$, telle que :

$V(z_\alpha \setminus y^*, y^*) < V(z_i \setminus y, y)$, $\forall z_i \in Z_k$, $\forall y \subset z_i$ $i : 1 \dots k$.

En particulier nous avons :

$V((z_p \cup z_q) \setminus y^*, y^*) < ((z_p \cup z_q) \setminus z_p, z_q) = V((z_p \cup z_q) \setminus z_q, z_p) = V(z_p, z_q)$,

nous avons alors :

$V(R2(R1(Z)))=V(R1(Z))+V((z_p \cup z_q) \setminus y^*, y^*)$ qui est inférieure à $V(R1(Z))+V(z_p, z_q)=V(Z)$.

**Preuve de (i2)**

La preuve de (i2) est la même que celle de (i1).

**Proposition (ii)**

Chaque partition optimale de niveau k, $Z_k^*$ est un point fixe des applications R1oR2 et R2oR1, cela signifie que :

$V(R1oR2(Z_k^*))=V(R2oR1(Z_k^*))=V(Z_k^*)$, $\forall k : 2 \dots n-1$.

**Démonstration**

**Preuve de (ii)**

$R2oR1(Z_k^*)$ et $R1oR2(Z_k^*) \in W_k(A)$, par la définition de l'optimalité :

$V(Z_k^*) <= V(R1oR2(Z_k^*))$ et $V(Z_k^*) <= V(R2oR1(Z_k^*))$, $\forall$ k : 2 ... n-1.

Puisque l'utilisation de la proposition précédente fournit l'inégalité dans le sens opposé, cela prouve l'égalité annoncé.

**Complexité de l'algorithme mixte**

L'algorithme mixte fournit pour chaque niveau une ou plusieurs partitions stables. Parmi celles-ci, nous pouvons être sûr que pour les niveaux 1 et n (trivialement), ainsi que pour les niveaux 2 et n-1, les optimums globaux seront trouvés, étant donné que l'algorithme mixte réunit les propriétés des algorithmes R1 et R2.

Pour les autres niveaux k compris strictement entre 3 et n-2, les partitions trouvées possèdent une propriété nécessaire d'optimalité. La question qu'on se pose alors et de savoir si l'ensemble des partitions stables est petit, ou si par contre, presque toutes les partitions sont stables. C'est une question ouverte. Nous pouvons dire seulement que dans plusieurs exemples que nous avons étudié, il a suffit de déterminer à peine quelques partitions stables pour trouver l'optimum global. Par ailleurs, le coût de communication des partitions stables était en général très près de la valeur optimale.

Les quelques exemples que nous avons étudiés nous ont montré aussi que lors du processus de stabilisation à un niveau k compris strictement entre 3 et n-2, on est amené à appliquer R1oR2 ou R2oR1 dans un ordre assez irrégulier : on applique R1oR2 jusqu'à l'obtention d'une partition fixe pour cet opérateur, ensuite on applique R2oR1 jusqu'à l'obtention à nouveau d'une partition fixe pour celui-ci, à nouveau R1oR2, puis R2oR1, ..., jusque stabilisation pour les deux opérateurs. On suppose donc qu'en moyenne j applications d'un opérateur de contraction reviennent à j/2 applications de R1oR2 et j/2 applications de R2oR1. Pour les niveaux k=2 et k=n-1, une seule application de R2 et de R1, respectivement suffisent pour trouver la partition optimale du niveau.

Soit $\aleph(n,k,j/2)$ (respectivement $\Im(n,k,j/2)$) la plus petite borne supérieure du nombre de partitions examinées par j/2 applications de l'opérateur R1oR2 (respectivement R2oR1) au niveau k du treillis de partitions de n atomes de distribution. En combinant les résultat (4.9) et (4.10) on peut dire que :

$$\aleph(n,k,j/2)=\frac{j}{2}\left|(2^{n-k}-1)+\frac{k(k-1)}{2}\right|$$

$$\Im(n,k,j/2)=\frac{j}{2}\left|(2^{n-(k-1)}-1)+\frac{k(k-1)}{2}\right|=\aleph(n,k-1,j/2).$$

Enfin, supposons qu'on se donne j partitions initiales à chaque niveau k, et qu'à chacune d'elles on applique j fois les opérateurs de contraction. Le nombre total des partitions examinées, autrement dit, la complexité de l'algorithme mixte est borné supérieurement par :

$$\frac{2^n-2}{2}+\frac{n(n-1)}{2}\sum_{k=3}^{n-2}\left|\aleph(n,k,j/2)+\Im(n,k,j/2)\right|$$

La complexité de l'algorithme mixte est exponentielle ($2^n$) d'ou l'intérêt d'une implantation parallèle de ce dernier.

## IV.5 Illustration

Soit $A=\{a_1, a_2, a_3, a_4, a_5, a_6\}$ l'ensemble des atomes de distribution du système d'automatisation,

$Z=\{\{a_1\}, \{a_2\}, \{a_3\}, \{a_4\}, \{a_5\}, \{a_6\}\}$,

Soit le tableau initial suivant, décrivant les coûts de communication entre les atomes de distribution de A :

| | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ | $a_6$ |
|---|---|---|---|---|---|---|
| $a_1$ | **0** | 67 | 58 | 87 | 76 | 84 |
| $a_2$ | | **0** | 0 | 17 | 46 | 55 |
| $a_3$ | | | **0** | 3 | 66 | 28 |
| $a_4$ | | | | **0** | 33 | 59 |
| $a_5$ | | | | | **0** | 8 |
| $a_6$ | | | | | | **0** |

Le coût de communication du système dans sa solution initiale est égal au coût de communication entre tous les atomes du système d'automatisation :

$$V(Z)=V(\{a_1\}, \{a_2\}, \{a_3\}, \{a_4\}, \{a_5\}, \{a_6\})=687$$

Supposons qu'on va répartir ces atomes sur trois machines.

Soit $Z=\{\{a_1, a_2\}, \{a_3, a_4\}, \{a_5, a_6\}\}$, une solution initiale. Le tableau suivant, décrit les coûts de communication entre les trois sous-systèmes $z_1=\{a_1, a_2\}$, $z_2=\{a_3, a_4\}$, et $z_3=\{a_5, a_6\}$.

| | $\{a_1, a_2\}$ | $\{a_3, a_4\}$ | $\{a_5, a_6\}$ |
|---|---|---|---|
| $\{a_1, a_2\}$ | **0** | 162 | 261 |
| $\{a_3, a_4\}$ | | **0** | 186 |
| $\{a_5, a_6\}$ | | | **0** |

$$V(Z)=V(\{a_1, a_2\}, \{a_3, a_4\}, \{a_5, a_6\})=609$$

**Premier cas :** l'algorithme mixte démarre par l'opérateur de décomposition R2. Le tableau des coûts de communication devient après l'application de R2 :

| | $\{a_1, a_2\}$ | $\{a_3\}$ | $\{a_4\}$ | $\{a_5, a_6\}$ |
|---|---|---|---|---|
| $\{a_1, a_2\}$ | **0** | 58 | 104 | **261** |
| $\{a_3\}$ | | **0** | 3 | 94 |
| $\{a_4\}$ | | | **0** | 92 |
| $\{a_5, a_6\}$ | | | | **0** |

$$V(Z)=V(\{a_1, a_2\}, \{a_3\}, \{a_4\}, \{a_5, a_6\})=612$$

Le tableau des coûts de communication après l'application de R1 :

| | $\{a_1, a_2, a_5, a_6\}$ | $\{a_3\}$ | $\{a_4\}$ |
|---|---|---|---|
| $\{a_1, a_2, a_5, a_6\}$ | **0** | 152 | 196 |
| $\{a_3\}$ | | **0** | 3 |
| $\{a_4\}$ | | | **0** |

$$V(Z)=(\{a_1, a_2, a_5, a_6\}, \{a_3\}, \{a_4\})=351$$

Le tableau des coûts de communication après l'application de R2 :

| | $\{a_1, a_2, a_6\}$ | $\{a_3\}$ | $\{a_4\}$ | $\{a_5\}$ |
|---|---|---|---|---|
| $\{a_1, a_2, a_6\}$ | **0** | 86 | **163** | 130 |
| $\{a_3\}$ | | **0** | 3 | 66 |
| $\{a_4\}$ | | | **0** | 33 |
| $\{a_5\}$ | | | | **0** |

$$V(Z)=V(\{a_1, a_2, a_6\}, \{a_3\}, \{a_4\}, \{a_5\})=481$$

Le tableau des coûts de communication après l'application de R1 :

| | $\{a_1, a_2, a_4, a_6\}$ | $\{a_3\}$ | $\{a_5\}$ |
|---|---|---|---|
| $\{a_1, a_2, a_4, a_6\}$ | **0** | 89 | 163 |
| $\{a_3\}$ | | **0** | 66 |
| $\{a_5\}$ | | | **0** |

$V(Z)=V(\{a_1, a_2, a_4, a_6\}, \{a_3\}, \{a_5\})=318$

Le tableau des coûts de communication après l'application de R2 :

| | $\{a_1, a_2, a_6\}$ | $\{a_2\}$ | $\{a_3\}$ | $\{a_5\}$ |
|---|---|---|---|---|
| $\{a_1, a_2, a_6\}$ | **0** | **139** | 89 | 117 |
| $\{a_2\}$ | | **0** | 0 | 46 |
| $\{a_3\}$ | | | **0** | 66 |
| $\{a_5\}$ | | | | **0** |

$V(Z)=V(\{a1, a2, a6\}, \{a2\}, \{a3\}, \{a5\})=457$

Le tableau des coûts de communication après l'application de R1 :

| | $\{a_1, a_2, a_4, a_6\}$ | $\{a_3\}$ | $\{a_5\}$ |
|---|---|---|---|
| $\{a_1, a_2, a_4, a_6\}$ | **0** | 89 | 163 |
| $\{a_3\}$ | | **0** | 66 |
| $\{a_5\}$ | | | **0** |

$V(Z)=V(\{a_1, a_2, a_4, a_6\}, \{a_3\}, \{a_5\})=318$

**Deuxième cas :** l'algorithme mixte démarre par l'opérateur d'agrégation R1. Le tableau des coûts de communication devient après l'application de R1 :

| | $\{a_1, a_2, a_5, a_6\}$ | $\{a_3, a_4\}$ |
|---|---|---|
| $\{a_1, a_2, a_5, a_6\}$ | **0** | **348** |
| $\{a_3, a_4\}$ | | **0** |

$V(Z)=(\{a1, a2, a5, a6\}, \{a3, a4\})=348$

Le tableau des coûts de communication après l'application de R2 :

| | $\{a_1, a_2, a_5, a_6\}$ | $\{a_3\}$ | $\{a_4\}$ |
|---|---|---|---|
| $\{a_1, a_2, a_5, a_6\}$ | **0** | 152 | **193** |
| $\{a_3\}$ | | **0** | 3 |
| $\{a_4\}$ | | | **0** |

$V(Z)=V(\{a_1, a_2, a_5, a_6\}, \{a_3\}, \{a_4\})=351$

Le tableau des coûts de communication après l'application de R1 :

| | $\{a_1, a_2, a_4, a_5, a_6\}$ | $\{a_3\}$ |
|---|---|---|
| $\{a_1, a_2, a_4, a_5, a_6\}$ | **0** | **155** |
| $\{a_3\}$ | | **0** |

$V(Z)=(\{a_1, a_2, ,a_4, a_5, a_6\}, \{a_3\})=155$

Le tableau des coûts de communication après l'application de R2 :

| | $\{a_1, a_2, a_4, a_6\}$ | $\{a_3\}$ | $\{a_5\}$ |
|---|---|---|---|
| $\{a_1, a_2, a_4, a_6\}$ | **0** | 89 | **163** |
| $\{a_3\}$ | | **0** | 66 |
| $\{a_5\}$ | | | **0** |

$V(Z)=V(\{a_1, a_2, a_4, a_6\}, \{a_3\}, \{a_5\})=318$

L'application de R1oR2 et celle de R2oR1 donnent comme résultats le même point fixe qui est un optimum local de la fonction coût V.

Le tableau suivant résume l'évolution du coût de communication du système :

| Premier cas | | Deuxième cas | |
|---|---|---|---|
| **Opérateurs** | **V(Z)** | **Opérateurs** | **V(Z)** |
| Initial | 609 | Initial | 609 |
| R2 | 612 | R2 | 348 |
| R1 | 351 | R1 | 351 |
| R2 | 481 | R2 | 155 |
| R1 | **318** | R1 | **318** |
| R2 | 457 | R2 | 457 |
| R1 | **318** | R1 | **318** |

La figure 4.6 décrit l'évolution des coûts de communication, pour les applications successives des opérateurs d'agrégation et de décomposition.

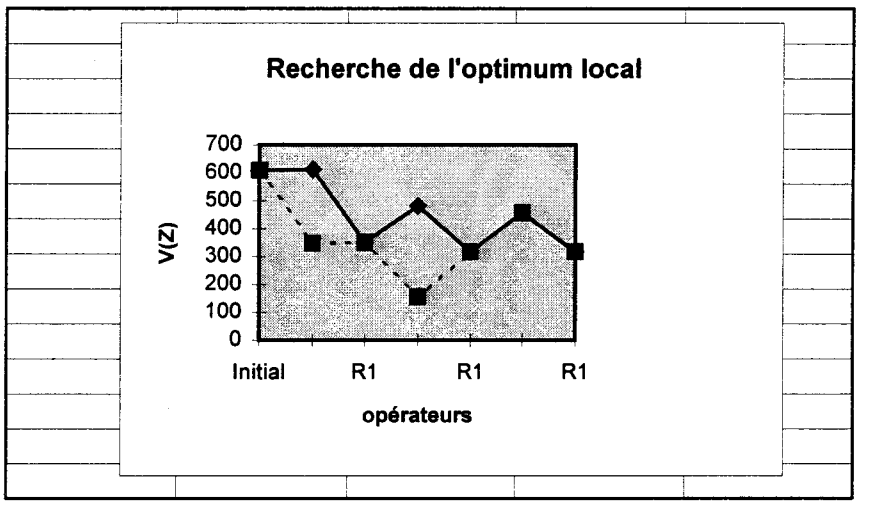



**Figure 4.6**

*Recherche de l'optimum local*

## IV.6 Conclusion

Dans ce chapitre, nous avons présenté trois algorithmes pour déterminer le placement des processus d'un système d'automatisation d'une manière heuristique : l'algorithme d'agrégation, de décomposition et mixte. L'utilisation de l'un de ces algorithmes dépend de plusieurs paramètres : le nombre d'atomes de distribution à répartir, les ressources de calcul, le choix de la solution initiale.

L'algorithme mixte qui est une combinaison des deux autres algorithmes, suppose l'existence d'une solution initiale composée de k classes. Il ne procédera pas au changement de nombre de classes, mais à la modification de la répartition initiale. L'algorithme procède par des combinaisons successives des deux opérateurs R1 et R2. L'opérateur R1 fournit une partition de (k-1) classes sur lesquelles il est suffisant d'appliquer l'opérateur de décomposition pour obtenir de nouveau le nombre initial de classes k. L'algorithme s'arrête à l'obtention d'un point fixe, celui-ci est un optimum local de la fonction coût.

L'algorithme peut commencer par l'opérateur d'agrégation comme il peut commencer par l'opérateur de décomposition. La solution obtenue ne sera pas obligatoirement la même.

# Conclusion générale

Rappelons que le travail présenté dans cette thèse, s'est déroulé dans le cadre d'un projet soutenu par le Ministère de la Recherche et de l'Enseignement Supérieur. Ce projet, qui s'intitule "**Impact des réseaux de terrains et de l'instrumentation intelligente dans la conception des systèmes automatisés de production**" et dont les résultats et les perspectives figuraient dans le rapport de fin de contrat convention N° 92.P.0239, intéressait plusieurs institutions d'horizons et d'objectifs divers. La participation à ce projet nous a permis naturellement des échanges d'idées et des discussions fructueuses avec des chercheurs venant de plusieurs entreprises industrielles et laboratoires de recherche.

Nous avons proposé dans ce travail une modélisation aussi claire et concise que possible des traitements composant le système d'automatisation, ainsi que des stratégies de répartition basée sur une approche de classification. Pour aboutir à ces résultats, nous avons démarré progressivement par :

- une prise de connaissance des traitements du système automatisé de production et les services qu'ils assurent,
- une modélisation détaillée de l'architecture fonctionnelle du système d'automatisation. En effet un nouveau concept : celui d'atome de distribution a été introduit, justifié et caractérisé,
- une formalisation du problème de répartition du système automatisé de production, et l'étude des outils permettant l'aboutissement à des solutions distribuées admissibles et acceptables. Ces outils concernent les critères, les contraintes qui régissent la répartition, ainsi que des algorithmes relevant de la littérature. Certains de ces algorithmes ont été détaillés,

- une proposition d'un ensemble de stratégies basées sur une approche de classification, permettant la projection de l'ensemble des atomes de distribution sur l'architecture matérielle du système d'automatisation.

  Ces stratégies regroupent les outils indispensables à l'aboutissement à une architecture opérationnelle : critères, contraintes et algorithmes de répartition. Trois algorithmes ont été décrits. Nous avons montré que l'algorithme mixte converge vers un optimum local. A partir de celui-ci, il est possible d'aboutir vers l'optimum global en utilisant d'autres algorithmes tel que celui du recuit simulé.

Certains points restent à approfondir et d'autres restent à étudier :

- le premier point, concerne la définition d'une méthodologie de décomposition du système d'automatisation en atomes de distribution. En effet le processus de décomposition n'est pas jusque là formalisé et se base généralement sur l'expérience des concepteurs du systèmes d'automatisation,
- le deuxième point concerne la définition d'une méthodologie d'analyse des critères et des contraintes définissant le problème de la distribution; évaluation et prise en compte dans un cadre multicritères.

  La prise en compte de critères et de contraintes dont certains s'expriment sous forme numérique et d'autres sous forme qualitative (en particulier les contraintes issues de la normalisation) est essentielle,
- le troisième point concerne l'étude des contraintes temporelles régissant la coopération entre atomes de distribution. En effet bien que certaines contraintes aient été présentées, il reste à développer des modèles pour analyser la cohérence temporelle de l'ensemble de atomes de distribution "distribués", ainsi que définir des règles d'exécution (synchronisations, priorités, ...) de la solution distribuée.

  La définition de méthodes de preuve de la cohérence (logique, temporelle) d'une application distribuée est primordiale,

- un quatrième point concerne l'architecture matérielle du système d'automatisation. L'étude de l'interopérabilité des constituants matériels de

l'architecture matérielle s'avère nécessaire puisqu'elle touche entre autres les problèmes relatifs à l'extension et la sûreté du fonctionnement du système. L'intégration des modèles de réseaux pour l'évaluation des propriétés de temps de réponse et de charge réseau dans une architecture distribuée est indispensable.

Quand on proclama que la bibliothèque comprenait tous les livres, la première réaction fut un bonheur extravagant. A l'espoir éperdu succéda comme il est naturel, une dépression excessive.

**Jean-Luis Borges**
**"La bibliothèque de Babel"**

# Bibliographie


[Ack 87] D. H. Ackley, "A Connectionist Machine for Genetic Hillclimbing", Kluwer Academic Publications, Boston, 1987.

[Ada 74] T. L. Adam, "A Comparison of List Schedules for Parallel Processing Systems", Communications of the ACM, Vol. 17, No. 12, December 1974.

[Aho 74] A. V. Aho, J. E. Hopcroft, J. D. Ullman, "The design and analysis of computer algorithms", Reading, MA:Addison-Wesley, 1974.

[Ala 92] M. Alabau, T. Dechaise, "Ordonnancement temps réel par échéance", Techniques et Sciences Informatiques, Vol. 12, No. 3, 1992.

[Ara 89] Arago 8, "Systèmes experts et conduites de processus", Rapport de synthèse du groupe Systèmes experts, Observatoire Français des Techniques Avancées, Masson, 1989.

[Ara 94] Arago 15, "Informatique tolérante aux fautes", Rapport de synthèse du groupe Informatique tolérente aux fautes et des experts consultés, Observatoire Français des Techniques Avancées, Masson, 1994.

[Asl 92] R. Aslamian, "Rapport sur les services offerts aux applications temps réel", SCEPTRE 2, Juin 1992.

[Ban 87] U. Banergjee, C. D. Polychronopoulos, "Processor Allocation for Horizontal and Vertical Parallelism and Related Speedup Bounds", IEEE Tansactions on Computer, Vol. C-36, No. 4, Avril 87.

[Bay 92a] M. Bayart, M. Staroswiecki, "Smart actuators : generic functional architecture, service and cost analysis", IEEE SICICI'92, Singapore, Février 1992.

[Bay 92b] M. Bayart, M. Staroswiecki, "Coherence of distributed applications under critical time constraints", IFAC/IFIP Workshop on Real Time Programming WRTP 92, Bruges, Belgium, 25-26 Juin 1992.

[Bay 95] M. Bayart, F. Simonot-Lion, "Impact de l'émergence des réseaux de terrain et de l'instrumentation intelligente dans la conception des architectures des systèmes d'automatisation de processus", Rapport final : Résultats et perspectives, Projet convention 92.P.0239, Février 1995.

[Ben 84] H. Benmaïza, "Le concept d'événement dans la spécification et la programmation d'applications temps réel", Thèse de doctorat, Université de Nancy I, Mars 1984.

[Ben 92] R. Bent, "Interbus-S, System Description", document constructeur, Phoenix Contact, 1992.

[Ber 81] P.A. Bernstein, N. Goodman, "Concurrency Control in Distributed Database Systems", ACM Computing Surveys, Juin 1981.

[Bha 90] B. Pigeron, H. Mullot, A. Chaix, L. Félix, Y. Aubert, "Boucles de régulation. Etude et mise au point, 2 ème édition, Editions Kirk, Collection Industrie, 1990.

[Bir 64] C. Birkhoff, "Lattice theory", American Mathematical Society, 4 ème édition, 1964.

[Bok 81a] S. H. Bokari, "A Shortest Tree Algorithm for Optimal Assignments Across Space and Time in a Distributed Processor System", IEEE Transactions on Software engineering, Vol. SE-7, No. 6, Novembre 1981.

[Bok 81b] S. H. Bokari, "On the Mapping Problem", IEEE Transactions on Computers, Vol. CE-30, No. 3, Mars 1981.

[Bol 88] S .W. Bollinger, S.F. Midkiff, "Processor and link assignment in multicomputers using simulated annealing", Proc. of the 11th Int. Conf. on Parallel Processing, The Penn. State Univ. Press, pp.1-7, Août 1988.

[Bro 84] F. Browaeys, H. Derrienning, P. Desclaud, H. Fallour, C. Faulle, J. Febvre, J. E. Hanne, M. Kronental, J. J. Simon, D. Vojnovic, "Sceptre : proposition de noyau normalisé pour les exécutifs temps réel", Techniques et Sciences Informatiques, Vol. 3, No. 1, 1984.

[Bry 81] R. M. Bryant, J. R. Agree, "A Queuing Network Approach to the Module Allocation Problem", ACM Computing Surveys, Vol. 21, No. 3, 1981.

[Car 84] J. Carlier, P. Chrétienne, C. Girault, "Modeling Scheduling Problems with Times Petri Nets", LNCS, No. 188, Advances in Petri Nets, 1984.

[Car 94] C. Cardera, "Ordonnancement temps réel par réseaux de neurones", Thèse de Doctorat de l'Institut National Polytechnique de Lorraine, Septembre 1994.

[Cas 88] T. Casavant, J. Kuhl, "A Taxonomy of Scheduling in General-purpose Distributed Computing System", IEEE transactions on Software Engineering, Vol. 14, No. 2, 1988.

[Cel 89] G. Celeux, E. Diday, G. Govaert, Y. Lechevallier, H. Ralambondrainy, "Classification automatique des données", Dunod, 1989.

[Cha 79] J. M. Champarnaud, "Contribution à l'étude du problème du voyageur de commerce", Thèse de l'Université de Franche-Comté, Juin 1979.

[Che 89] H. Chetto, M. Chetto, "Some Results of the Earliest Deadline Algorithm", IEEE Transactions on Software Engineering, Vol. 15, No. 10, Octobre 1989.

[Che 90] G-H. Chen, J-S. Yur, "A branch and bound with underestimates algorithm for the task assignment problem with precedence constraint", 10th Int. Conf. on Distributed Computing Systems, Paris, France, pp.494-501, Mai 1990.

[Cho 82] T. C. K. Chou, J. A. Abraham, "Load Balancing in Distributed Systems", IEEE Transactions on Software Engineering, Vol. SE-8, No. 4, Juillet, 1982.

[Cof 76] E. G. Coffman, "Computer and Job-shop Scheduling Theory", John Wiley and Sons, 1976.

[Coh 87] J.P. Cohoon, S.U. Hedge, W. N. Martin, D. Richards, "Punctuated Equilibra: A parallel genetic algorithm", Proc. of the Second Int. Conf. on Genetic algorithms, MIT, Cambridge, pp.148-154, Juillet 1987.

[Deb 93] J. L. Débouché, "Contribution à la conception des systèmes de contrôle commande", Thèse de doctorat de l'USTL, Novembre 1993.

[Dec 93] J. L. Decotignie, P. Raja, "Fullfilling temporal constraints in fieldbus", ICON'93, Vol. 1, Hawaii, Novembre 1993.

[Del 89] J. L. Delcuvellerie, "Connaissances en conception des architectures de systèmes informatisés d'automatisation - Outil d'analyse de la robustesse aux défaillances d'architectures réparties", Thèse de Doctorat de l'Institut National Polytechnique de Lorraine, Nancy, Mars 1989.

[Des 91] F. M. Desprès, "Automatisation des systèmes de production : du besoin à l'utilisation", Editions KIRK, 1991.

[Did 72] E. Diday, "Nouvelles méthodes et nouveaux concepts en classification automatique et reconnaissance de formes", Thèse de doctorat de l'Université Paris VI, Décembre, 1972.

[Din 90] Deutsches Institut für Normung, "Profibus", Norms DIN 19 245, part 1 et part 2, 1990.

[Dub 90] B. Dubuisson, "Diagnostic et reconnaissance des formes", Traité des Nouvelles Technologies, Série Diagnostic et Maintenance, Hermès, 1990.

[Dum 96] J. J. Duméry, F. Couffin, J. M. Faure, J. P. Frachet, S. Lampérière, "Un cadre de modélisation pour l'analyse du comportement des systèmes automatisés de production complexes et un outil adapté : l'HyperGrafcet", Modélisation des systèmes réactifs, Afcet, Paris, 1996.

[Far 85] H. Farreny, "Les systèmes experts : principes et exemples", Collection Techniques Avancées de l'Informatique, Cépadues, 1985.

[Fer 89] D. Fernandez-Baca, "Allocating modules to processors in a distributed system", IEEE Trans. on Soft. Eng., Vol.SE-15, No.11, pp.1427-1436, Novembre 1989.

[For 62] L.R.Ford, D.R.Fulkerson, "Flows in networks", Princeton, NJ:Princeton Univ. Press, 1962.

[For 65] E. W. Forgy, "Cluster analysis of multivariate data", Biometrics, Vol. 21, N° 3, Septembre 1965.

[Gal 84] D. Galara, J.P. Thomesse, "Proposition d'un système de transmission série multiplexé pour les échanges d'informations entre des capteurs, des actionneurs et des automates reflexes", Ministère de l'industrie et de la Recherche, 1984.

[Gar 79] M.R. Garey, D.S. Johnson, "Computers and intractability: A guide to the theory of NP-completeness", Freeman, San Francisco, 1979.

[Geh 94] A. L. Gehin, "Analyse fonctionnelle et modèle générique des capteurs intelligents : application à la surveillance de l'anesthésie", Thèse de l'Université des Sciences et Technologies de Lille , Janvier 1994.

[Gif 79] D. K. Gifford, "Weighted Voting for Replicated Data", Proceedings 7th Symposium.Operating Systems Principles, Pacific Grove, CA, ACM SIGOPS, December, 1979.

[Gol 89] D.E. Goldberg, "Genetic algorithms in search, optimization, and machine learning", Addison-Wesley, 1989.

[Had 88] P. Haden, F. Berman, "A Comparative study of mapping algorithms for an automated parallel programming environment", Tech. Rep. CS-088, Univ. of California, San Diego, 1988.

[Hou 87] C.E.Houstis, "Allocation of real-time applications to distributed systems", Proceedings of the 1987 Int. Conf. on Parallel Processing, pp.863-867, Août 1987.

[Jam 78] A. Jambu, M. O. Lebeaux, "Classification automatique pour l'analyse des données" : Méthodes et algorithmes, Dunod Décision, 1978.

[Kas 84] H. Kasahara, S. Narita, "Practical multiprocessor scheduling algorithms for efficient parallel processing", IEEE Trans. on Comp., Vol.C-33, No.11, pp.1023-1029, Novembre 1984.

[Kir 83] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, "Optimization by simulated annealing", Science, Vol.220, No.4598, pp.671-680, Mai 1983.

[Krä 87] O. Krämer, H. Mühlenbein, "Mapping strategies in message based multiprocessor systems", Springer Verlag, PARLE'87 Conf., LNCS, Vol.258, pp.213-225, 1987.

[Laa 87] P. J. M. Laarhoven, E. H. L. Aarts, "Simulated annealing: Theory and applications", D. Reidel Pub. Comp., 1987.

[Ler 70] I. C. Lerman, "Les bases de classification automatique", Gauthier-Villars, Collection programmation, Paris, 1970.

[Ler 76] I. C. Lerman, "Reconnaissance et classification des structures finis en analyse de données", Rapport N° 70, IRISA, 1976.

[Lo 88] V. L. Lo, "Heuristic Algorithms for Task Assignment in Distributed Systems", IEEE Transactions on computers, vol. 37, No. 11, Novembre 1988.

[Lor 94a] P. Lorenz, Z. Mammeri, "Temporal Mechanisms in Communication Models Applied to Companion Standards", 2nd IFAC Symposium, Budapest, Juin 1994.

[Lor 94b] P. Lorenz, "Le temps dans les architectures de communication : application au réseau de terrain FIP", Thèse de doctorat, INPL, Nancy, Juillet 1994.

[Ma 82] P. Y. R. Ma, E. Y. S. Lee, M. Tsuchiya, "A task Allocation Model for Distributed Computing Systems", IEEE Transactions on Computers, Vol. C-31, No. 1, January 1982.

[Mam 94] Z. Mammeri, J. P. Thomesse, "Réseaux locaux industriels : FIP et MAP dans les systèmes automatisés", Eyrolles, 1994.

[Map 88] Map 3.0, General Motors, 1988.

[Mar 87] G. Marie, "La pratique des automates programmables industriels. Procédures d'utilisation, langages et techniques de programmation.

Structures et mise en oeuvre", Les manuels professionnels, Collection de l'Usine Nouvelle, Editions du Moniteur, 1987.

[Müh 87] H. Mühlenbein, M. Georges-Schleuter, O. Krämer, "New Solutions to the Mapping Problem of Parallel Systems : the Evolution Approach", Parallel Computing 4, 1987.

[Mun 90] T. Muntean, E. G. Talbi, "Méthodes de placement statique des processus sur architectures parallèles", Technique et Sciences Informatiques, Vol. 10, No. 5, 1991.

[Par 60] E. Parzen, "Modern probability theory and its application", John Wiley and Sons, New York, 1960.

[Pat 89] R. Patton, P. Frank, R. Clark, "Fault Diagnosis in Dynamic System. Theory and Applications - Prentice Hall International Series in Systems and Control Engineering (U.K.), 1989.

[Per 83] Y. Perl, M. Snir, "Circuit partitioning with size and connection constraints", Networks, Vol.13, No.3, pp.365-375, 1983.

[Pos 87] J. G. Postaire, "De l'image à la décision", Dunod Informatique, Paris 1987.

[Pri 94] C. Prins, "Algorithmes de graphes", Editions Eyrolles, Paris, 1994.

[Pou 91] D. L. V. Poussin, "DIAS : Distributed Intelligent Actuators and Sensors", ESPRIT PROJECT 2172., Journée L'instrumentation intelligente", Paris, Avril 1991.

[Que 67] J. B. Mac Queen, "Some methods for classification and analysis of multivariate observations", 5th Berkley Symposium on Mathematical Statistics and Probability, Vol. 1, N° 1, Berkley University of California Press, 1967.

[Rag 90] J. Ragot, M. Darouach, D. Maquin, G. Bloch, "Validation de données et diagnostic", Traité des nouvelles technologies, Hermès, Paris, 1990..

[Ram 86] C.V. Ramamoorthy, J. Srivastava, W-T.Tsai, "A distributed clustering algorithm for large computer networks", 6th Int. Conf. on Distributed Computing Systems, Cambridge, Massachusetts, pp.613-620, Mai 1986.

[Ram 89] K. Ramamrithan, J. Stankovic, W. Zhao, "Distributing Scheduling of Tasks with Deadlines and Resource Requirements", IEEE Transactions on Computers, Août 1989.

[Rea 77] R.C. Read, D.G. Corneil, "The graph isomorphism disease", Journal of Graph Theory, Vol.1, pp.339-363, 1977.

[Ren 88] R. V. Renesse, A. S. Tanenbaum "Voting with Ghosts", Proceedings of 8 th International Conference on Distributed Computing Systems, Juin, 1988.

[Rob 93] M. Robert, M. Marchandiaux, M. Porte, "Capteurs intelligents et méthodologie d'évaluation", Hermès, 1993.

[Rol 91] P. Rolin, Réseaux locaux : normes et protocoles, Hermés, 1991.

[Rou 88] T. J. Routt, "From TOP (3.0) to bottom : architectural close-u", Data Communication, Mai 1988.

[Sak 84] A. Sakari, "FLEXI : Langage de conception d'application de conduite de procédés industriels", Thèse de doctorat, Université de Nancy I, Mars 1984.

[Ser 91] F. Seredynski, "Task allocation by a team of learning automata", Proc. of the 6th Int. Conf. on Methodologies for Intelligent Systems, North Carolina, Oct 1991.

[Sha 90] K. Shahookar, P. Mazumder, "A genetic approach to standard cell placement using meta-genetic parameter optimization", IEEE Trans. on Computer-aided design, Vol.9, No.5, pp.500-511, Mai 1990.

[She 85] C. Shen, W. Tsai, "A Graph Matching Approach to Optimal Task Assignement in Distributed Computing Systems Using A Minmax Criterion", IEEE Transactions on Computers, Vol. C-34, No. 3, Mars 1985.

[She 87] J. Sheild, "Partitioning concurrent VLSI simulation programs onto a multiprocessor by simulated annealing", IEEE Proceedings, Vol.134, Pt.E, No.1, pp.24-30, Janvier 1987.

[Shi 90] B. Shirazi, M. Wang, G. Pathak, "Analysis and evaluation of heuristic methods for static task scheduling", J. of Parallel and Distributed Comp., Vol.10, No.3, pp.222-232, Novembre 1990.

[Shi 93] K. G. Shin, Q. Zheng, "Mixed time-constrainted and non-time-constrainted communications in local area networks", IEEE Transaction on Communications, Vol. 41, No. 11, Novembre 1993.

[Sim 92] F. Simonot-Lion, C. Verlinde, "Importance d'un cadre de référence dans la mise en place d'une démarche de développement d'un système

automatisé de production", Actes de conférence Automatisation Industrielle, Vol 1, Montréal, Canada, 1992.

[Sim 95] F. Simonot-Lion, "La démarche de conception des systèmes d'automatisation", Journées d'Etude SAPID, Paris, Mai, 1995.

[Sin 87] J.B. Sinclair, "Efficient computation of optimal assignments for distributed tasks", J. of Parallel and Distributed Computing, Vol.4, pp.342-362, 1987.

[Sta 94a] M. Staroswiecki, M. Bayart, J. Akaichi, "Distribution of Intelligent Automated Production System - a clustering approach", Congrès IFAC, Baden Baden, Septembre 1994.

[Sta 94b] M. Staroswiecki, M. Bayart, "Actionneurs intelligents", Hermès, Paris, 1994.

[Ste 85] C.S. Steele, "Placement of communicating processes on multiprocessor networks", Tech. Rep. 5184:TR:85, California Institute of Technology, Pasadena, Avril 1985.

[Sto 77] H.S.Stone, "Multiprocessor scheduling with the aid of network flow algorithms", IEEE Trans. on Soft. Eng., Vol.SE-3, No.2, pp.85-93, Janvier 1977.

[Sto 78] H. Stone, "Critical Load factors in two-processor distributed systems", IEEE Transactions on Software Engineering, Vol. SE-4, No. 3, Mai 1978.

[Tal 91] E-G.Talbi, P.Bessière, "Parallel genetic algorithms : performances and applications", Int. Conf. on Novel Methods in Optimization, Copenhagen, Denmark, Février 1991.

[Tho 80] J.P. Thomesse, "SYGARE : Une structuration pour la conception d'application temps réel et réparties", Thèse de doctorat, INPL, Octobre 80.

[Tho 89] J.P. Thomesse, "Time critical local area networks. Analysis of needs and characteristics", Contribution à ISO TC 184/SC5/WG2-TCCA, Octobre 89.

[Tho 90] J.P. Thomesse, "Les réseaux locaux à temps critique", AFCET, 1990.

[Tho 91] J.P. Thomesse, P. Lorenz, "Fieldbus, Communications models - Real Time Data Base", ANIA, Milano, 1991.

[Tow 86] D. Towsley, "Allocating programs containing branches and loops within a multiple processor system", IEEE Trans. on Soft. Eng., Vol.SE-12, No.10, pp.1018-1024, Octobre 1986.

[Udi 90] N. Udiavar, G.S. Stiles, "A simple but flexible model for determining optimal task allocation and configuration on a network of transputers", NATUG1: Transputer Research and Applications, Utah, USA, pp.24-32, Avril 1990.

[Ute 90a] FIP Couche d'application MPS, NF C46-602, Union Technique de l'Electricité, Courbevoie, 1990.

[Ute 90b] FIP Couche Liaison de Données, NF C46-603, Union Technique de l'Electricité, Courbevoie, 1990.

[Ute 90c] FIP Couche Physique en bande de base sur paire torsadée blindée, NF C46-604, Union Technique de l'Electricité, Courbevoie, 1990.

[Ver 89] C. Verlinde, E. Georgel, J.P. Thomesse, "A service oriented hierarchical model for the design of control system", CCCT 89, IFAC AFCET Symposium, Septembre, 1989.

[Xu 93] J. Xu, D. L. Parnas, "On Satisfying Timing Constraints in Hard-Real-Time Systems", IEEE Transactions On Software Engineering, Vol. 19, No. 1, Janvier 1993.

# Annexe 1 : Groupe de travail du projet MESR 2033 :

# Laboratoires de recherche et industriels


**Centre de Recherche en Automatique de Nancy-CNRS**

Frédéric Hermann,
Jean-Luc Noizette
Michel Robert

**Centre de Recherche en Informatique de Nancy-CNRS**

Françoise Simonot-Lion
Jean-Pierre Thomesse
Luis Vega-Saens

**Ecole Supérieure d'Ingénieurs en Electronique et Electrotechnique**

Olivier Vennard

**Ecole Polytechnique Fédérale de Lausane**
**Laboratoire d'Informatique Technique**

Jean-Dominique Decotignie
Raja Prasard

**Laboratoire d'Automatique de Grenoble-CNRS**

Jean-Pierre Acquadro
Zdenek Binder
Adolfo Rodriguez

**Laboratoire d'Automatique et de Mécanique Industrielles et Humaines de Valenciennes-CNRS**

Laurent Cauffriez
Jean Defrenne

**Laboratoire d'Automatique et d'Informatiques Industrielles de Lille-CNRS**

Jalel Akaichi
Mireille Bayart
Marcel Staroswiecki

**Laboratoire d'Automatique de Nantes-CNRS**

Denis Creusot
Anne-Marie Deplanche
Jean-Pierre Elloy

**BERTIN**

Pierre Midavaine

**CETIM**

Walid Youssef

**EDF-DER**

Pierre Berthomier
Pierre-Henri Delmaire
Tanguy Le Quenven
Mazen Samaan

# Annexe 2 : Les systèmes de communication

On distingue deux types de système de communication :

- un système de communication qu'on peut qualifier d'externe. Il gère les échanges du système d'automatisation avec ses deux interfaces. Du côté processus, il assure la réception des données issues des capteurs et la transmission des commandes vers les actionneurs. Du côté opérateur, il permet la saisie et la diffusion de diverses informations.
- le système de communication qu'on peut qualifier d'interne qui joue le rôle de lien entre les constituants de l'architecture opérationnelle, sur lesquels on va répartir des atomes de distribution. Ceux-ci seront ainsi capables d'échanger des informations.

Les réseaux locaux industriels constituent les systèmes de communication "internes" les plus utilisés dans les systèmes automatisés de production. Ils sont devenus, en tant que support des flux d'informations entre les constituants, une composante incontournable de l'architecture opérationnelle [Mam 94].

## 1 Topologies des systèmes de communication

La topologie d'un système de communication décrit sa géométrie ou sa configuration physique. C'est la manière dont sont interconnectées les machines de l'architecture matérielle du système d'automatisation. Dans le domaine des réseaux locaux, les configurations les plus utilisées sont [Rol 91]:

- l'étoile : cette configuration est la plus connue et fut la première à voir le jour. Elle est caractérisée par une machine centrale (un micro-ordinateur en général) auquel chaque machine du réseau est reliée. La machine centrale gère et contrôle l'ensemble des autres machines et des liaisons de communication. Dans cette topologie, le transfert de l'information entre machines s'effectue en mode point à point,

- l'anneau : dans cette configuration, le support de transmission relie toutes les machines de manière à former une boucle ou circuit. L'information circule dans un seul sens, le long du support de transmission, et les données sont collectées par les machines branchées sur la liaison en anneau. Dans ce type de topologie, le mode d'exploitation utilisé consiste à allouer des intervalles de temps émission-réception à chaque machine branchée sur l'anneau,
- le bus : dans cette configuration, l'ensemble des machines sont raccordées à un câble d'une certaine longueur constituant une liaison physique commune. Toute information transmise par une machine, transite par le câble pour atteindre les autres machines du système. Chacune de ces dernières examine l'adresse spécifiée dans l'entité informationnelle en cours de transmission pour déterminer si ce message le concerne. La configuration bus est fréquemment utilisée dans les systèmes automatisés de production, pour les facilités d'installation et d'exploitation qu'elle offre.

## 2 Quelques critères de choix des systèmes de communication

Pour assurer la circulation du flux d'information entre les différentes machines l'architecture matérielle du système d'automatisation, la présence de systèmes de communication est indispensable. Ces systèmes de communication doivent garantir généralement :

- des délais de transmission aussi rapides que la criticité des informations circulant entre les différents atomes de distribution l'exige,
- une fiabilité très bonne de communication; ainsi le taux d'erreurs touchant les informations communiquées, doit être très faible,
- une maintenance facile aussi bien pour le câblage que pour la connexion des différentes machines (instruments intelligents ou autres),
- une connexion facile avec d'autres systèmes de communication,
- la gestion des droits d'accès de façon à assurer un niveau de sécurité satisfaisant, ...

## 3 Le modèle de communication OSI

Pour communiquer les machines doivent respecter généralement un modèle de référence. Ce modèle connu sous le nom d'OSI, décompose le protocole (un protocole est un accord sur la façon dont les communications doivent s'effectuer) global définissant les règles de dialogue entre un ensemble d'équipements hétérogènes, en sept couches.

Un système de communication est dit ouvert [Mam 94], lorsqu'il permet la communication entre équipements de types différents, pouvant provenir de constructeurs différents, pourvu que ces équipements respectent les règles de communication dans un environnement OSI. Inversement, un système est dit fermé [Mam 94], lorsqu'il ne permet la communication qu'entre des équipements d'un même type, ou d'un même constructeur, en utilisant des protocoles qui sont la propriété de quelqu'un.

Dans le modèle OSI, la communication des informations est décomposée en sept couches : physique, liaison de données, réseau, transport, session, présentation, et application. Nous décrivons ci-dessous ces couches une à une, un schéma récapitulatif sera également présenté (figure A.1).

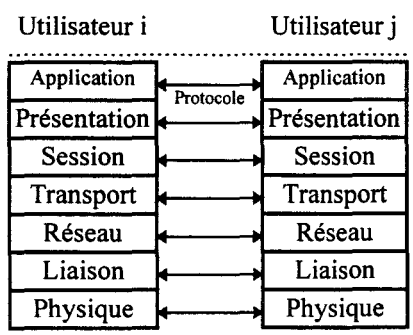



**Figure A.1**

*Architecture du modèle de référence OSI*

**La couche physique**

La couche physique offre les moyens d'établir, de maintenir et de libérer une connexion physique entre deux machines. Elle assure la transmission des bits entres elles.

**La couche liaison**

La couche physique est donc chargée de l'envoi des bits. Cependant les réseaux de communication sont sujets aux erreurs qu'il faut donc détecter et corriger. Le rôle principal de la couche liaison est d'assurer l'acheminement sans erreurs les informations. Dans ce but, elle détecte et corrige les erreurs de transmission et contrôle le flux des informations transmises.

**La couche réseau**

L'expéditeur d'un message n'a généralement pas besoin de savoir où se trouve le destinataire. Il émet son message sur le réseau et le destinataire le récupère. Ce message peut transiter par des stations intermédiaires avant d'arriver à destination. Un réseau long distance comprend un grand nombre de machines, chacune ayant plusieurs liaisons avec d'autres machines. Le choix du meilleur chemin entre deux équipements s'appelle le routage et c'est la tâche principale de la couche réseau. Dans les réseaux locaux, cette couche est normalement vide. Elle peut toutefois exister en ayant pour mission la réalisation d'une passerelle avec un autre réseau.

**La couche transport**

On peut perdre des paquets (un paquet est un ensemble fini de bits) entre l'expéditeur et le destinataire. Si certaines applications préfèrent gérer leur propre récupération des erreurs, d'autres applications préfèrent avoir une connexion fiable. La fonction de la couche transport est de fournir ce service et d'assurer la

fiabilité de la connexion. Cette couche réalise notamment la segmentation de messages en paquets à l'émission et le réassemblage des paquets à la réception.

**La couche session**

La couche session est essentiellement une version enrichie de la couche transport. Elle fournit des dialogues de contrôle pour mémoriser les connexions en cours ainsi que des mécanismes de synchronisation. Elle est employée pour poser des points de reprise lors de gros transferts. Si une panne se produit en cours de transfert, il suffit de retourner au dernier point de reprise au lieu de devoir repartir du point de départ. En pratique, peu d'applications sont intéressées par la couche session et elle est rarement implémentée.

**La couche présentation**

La couche présentation traite les aspects de représentation de l'information afin d'assurer la compatibilité des échanges entre les équipements communicants. Ces derniers peuvent utiliser des codes, des syntaxes et des représentations d'informations qui peuvent être spécifiques à chacun d'eux. C'est le rôle de la couche présentation de masquer les particularités entre machines.

**La couche application**

La couche application offre des moyens divers permettant d'accéder à l'environnement OSI. Elle fournit à l'utilisateur (atome ou opérateur humain) l'ensemble des services liés à la répartition des informations sur différentes machines.

## 4 Les réseaux locaux industriels

La nécessité de faire communiquer un ensemble de traitements situés sur plusieurs machines différentes implantées dans un espace géographique restreint, a entraîné l'introduction d'un nouveau type de système de communication : les réseaux locaux industriels.

Les réseaux locaux peuvent être classifiés en fonction de la nature des échanges selon trois niveaux distincts (niveau zéro, niveau 1, niveau 2), qui répondent chacun à trois catégories de besoins de communication [Lor 94b]. Les différents critères de classification sont :

- la fréquence des informations transmises,
- les contraintes de temps associées aux échanges d'informations,
- les fonctions mises en oeuvre,
- le type et la structure de l'information échangée ainsi que l'environnement dans lequel on se situe.

### 4.1 Le niveau zéro

Au niveau zéro, on retrouve les échanges d'informations nécessitant de respecter de fortes contraintes de temps. On se situe dans le domaine des réseaux de terrain. Le besoin de communication est très déterministe et une grand partie du trafic se fait d'une manière périodique. Les échanges d'information sont souvent de faible taille et ils se font essentiellement entre des capteurs, des actionneurs et des automatismes. L'unité de temps généralement employée est la milliseconde ou la dizaine de millisecondes. Citons, à titre d'exemples, les réseaux de terrain FIP [Ute 90a], [Ute 90b], [Ute 90c], et InterBus_S [Ben 92].

## 4.2 Le niveau un

Au niveau un, l'information est beaucoup plus structurée et les contraintes de temps sont moins strictes que celle du niveau zéro. C'est dans cette partie que l'on retrouve généralement des traitements de téléchargement, de télélecture de programmes entre des automates, des postes opérateurs et des consoles de programmation. Ces traitements peuvent se trouver également au niveau zéro [Lor 94b]. L'unité de temps employée se situe entre la milliseconde et la seconde. Citons, à titre d'exemples, les réseaux mini-MAP [Map 88], et ProfiBus_S [Din 90].

## 4.3 Le niveau deux

Au niveau deux, les informations échangées sont souvent des fichiers et des données volumineuses. On retrouve donc des échanges aléatoires d'informations entre les systèmes de Conception et de Fabrication Assistée par Ordinateur (CFAO), de Gestion de Production Assistée par Ordinateur (GPAO) et de Conception Assistée par Ordinateur (CAO).

Les réseaux locaux sont caractérisés par leurs topologies, leurs modes de transmission des données, et leurs codages de l'information. Les techniques de commutation constituent également un élément essentiel de caractérisation des réseaux locaux. L'unité de temps employée est souvent de l'ordre de la seconde.

Un exemple d'illustration des trois niveaux hiérarchiques peut se représenter de la manière suivante (figure A.2) [Lor 94b].

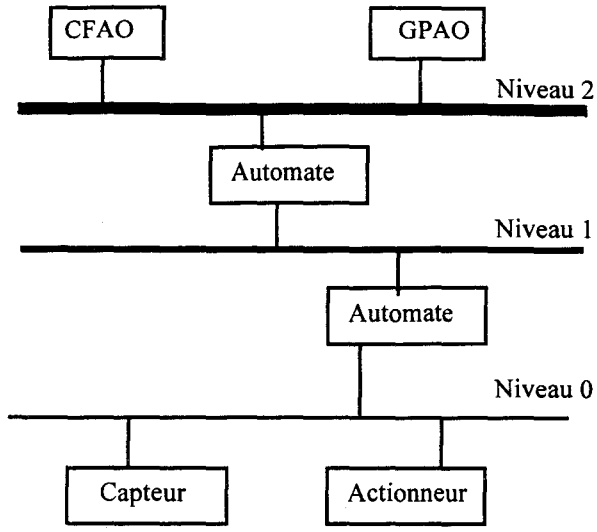



**Figure A.2**

*Représentation des différents niveaux présents dans les réseaux locaux industriels [Lor 94b]*

Les réseaux des niveaux inférieurs sont généralement capables de répondre aux exigences de temps des réseaux de niveaux supérieurs, où les contraintes de temps sont généralement moins fortes [Shi 93]. Par contre, un réseau de niveau supérieur ne peut pas toujours répondre aux exigences temporelles des niveaux inférieurs.

## 4.4 Les réseaux de terrain

La littérature concernant les réseaux de terrains est abondante. La plus grande partie de cette littérature est proposée dans un but commercial, pour satisfaire partiellement des spécifications, et/ou pour fournir des solutions à des problèmes spécifiques dans des délais courts [Dec 93]. Les exemples de ce type de réseau sont nombreux : Bitbus de Intel, Genius de GE-Fanuc, Interbus-S de Phoenix Contact, Bitbus, LON de Echelon, Batibus, ...

Quelques uns des exemples cités ci-dessus sont caractérisés par des restrictions qui rendent difficile leur utilisation dans des domaines autres que ceux pour lesquels ils ont été conçus.

D'autres réseaux de terrain sont des standards nationaux. En effet MIL-STD 1553, FIP, Profibus sont des standards respectivement au Etats unis, en France et en Allemagne.

Les réseaux de terrain permettent de connecter, et d'assurer la communication entre les machines de niveau zéro et les machines de niveau un. Les capteurs, les actionneurs, les interfaces opérateurs, et les instruments de contrôle (automates programmables, robots, les régulateurs) sont des équipements souvent connnectés à l'aide de réseaux de terrain.

## 4.5 Le réseau local FIP

Le réseau FIP est considéré comme un réseau de terrain au niveau zéro. Son rôle principal est de permettre aux capteurs, actionneurs et aux contrôleurs de communiquer [Gal 84]. Il est important de signaler qu'au niveau zéro, la plupart des données échangées sont soumises à des contraintes de temps. Le réseau FIP tient compte de ces contraintes. Il est spécialement conçu pour gérer et pour mettre à jour une base de données temps réel distribuée.

D'après OSI, FIP est caractérisé par une architecture réduite à une couche physique, une couche de liaison, et une couche application.

Deux types de données sont autorisés sur le système de communication FIP :

- les variables : une mesure de température, une commande vers un actionneur, par exemple,
- les messages classiques.

### 4.5.1 Les échanges de variables

Une variable est identifiée par un numéro de variable. Ce numéro est unique. Une variable est produite par un site unique appelé site producteur, et consommée par un ou plusieurs site(s) appelé(s) sites consommateurs(s).

Le contrôle d'accès au médium (centralisé au niveau du réseau FIP), est réalisé par un site "privilégié" appelé "arbitre de bus". Plusieurs sites peuvent être configurés pour jouer le rôle d'un arbitre de bus, mais un seul d'entre eux doit être actif à un instant donné.

L'arbitre de bus exploite une table de scrutation indiquant pour chaque variable sa date de délivrance sur le bus. La délivrance d'une variable est effectuéeen trois étapes :

- l'arbitre de bus diffuse l'identifiant de la variable à transmettre,
- le producteur de la variable correspondant à cet identifiant transmet sa valeur sur le bus,
- chaque consommateur concerné par la variable transmise sur le bus, reçoit sa valeur.

Le réseau FIP utilise le principe de diffusion dans le sens où chaque valeur d'une variable transmise sur le bus n'est pas explicitement adressée à une destination particulière; ce qui veut dire que le producteur de la variable n'a pas d'information sur le nombre et la localisation des consommateurs. C'est la responsabilité de chaque site de se reconnaître comme consommateur ou non.

### 4.5.2 Les échanges de messages

La transmission des messages est différente de la transmission des variables, puisqu'on ne peut pas configurer la destination des messages dès le démarrage de l'application.

Pour transférer un message d'une source à une ou plusieurs destination(s), un identificateur doit être diffusé par l'arbitre de bus et reconnu au préalable par son producteur.

Le producteur de message insère son adresse, celle des destination(s), et le corps du message dans une trame (bloc d'informations, composé et transmis selon un

ensemble de règles). Ensuite et selon l'adresse du ou des destinataires, les sites concernés par la diffusion du message, décideront de recevoir ou non le message en question.

En plus des services précédents, FIP fournit un mécanisme qui permet d'échanger une liste de variables dans une seule transaction. Ce type de service est d'une grande utilité pour les applications pour lesquelles un ensemble de valeurs (température, pression, commandes d'actionneurs, par exemple) est à délivrer dans un temps bien déterminé.

En conclusion, autour du réseau FIP sont connectés des sites qui assurent une ou plusieurs de ces fonctions :

- production (s) de données,
- consommation (s) de données,
- arbitrage du bus.

En utilisant la configuration de démarrage, chaque site reconnaît l'ensemble des identifiants qui sont associés aux objets/messages pour lesquels, ce site est producteur ou consommateur. Ainsi, la communication peut prendre les formes suivantes :

- transfert périodique/apériodique d'une variable d'un producteur vers un ou plusieurs consommateur(s),
- transfert périodique/apériodique d'un message, d'une source vers une destination.
- transfert périodique/ apériodique d'un message, d'une source vers plusieurs destinations.

## 5 Caractéristiques des réseaux locaux industriels

Le tableau suivant est une récapitulation des différentes caractéristiques d'un ensemble de réseaux locaux industriels [Lor 94]. Ces caractéristiques se rapportent aux paramètres suivants :

- la référence normative du réseau,
- la topologie utilisée,
- le support de transmission,
- le codage,
- la longueur maximale supportée par le réseau sans mettre en oeuvre de répéteurs,
- débit du réseau,
- mécanisme utilisé pour accéder au médium de communication,
- nombre maximal de stations qu'il est possible de connecter sur le réseau,
- énumération des couches du modèle OSI utilisées.

| | FIP | PROFIBUS | LONWORKS |
|---|---|---|---|
| **Norme** | NF C46-601 à NF C46-607 | DIN 19245-1 à DIN 19245-2 | Echelon |
| **Topologie** | Bus | Bus | Bus |
| **Support de transmision** | Paire torsadée, Fibre optique | Paire torsadée | Paire torsadée, fibre optique, câble coaxial, fréquence radio, infrarouge. |
| **Codage** | Manchester 2 | NRZ | |
| **Longueur maximale** | 2000 m | 1200 mètres | |
| **Débit** | 31,25 Kbit/s, 1; 2,5; 5 Mbits/s | 9,6;19,2;93,75;187,5 et 500 Kbit/s | 75 Kbit/s à 1,25 Mbit/s |
| **Accès au médium** | Arbitre de bus | Passage de jeton entre stations maîtres, interrogation des esclaves par les stations maîtres | CSMA/CD |
| **Nombre maximal de stations** | 256 | 32 maîtres, 127 esclaves | 32000 |
| **Couches OSI** | 1, 2 et 7 | 1, 2 et 7 | Les 7 couches |

| | BITBUS | FACTOR | FAIS |
|---|---|---|---|
| **Norme** | INTEL | APTOR | Japonaise |
| **Topologie** | Bus | Bus ou étoile | Bus |
| **Support de transmision** | Paire torsadée | Paire torsadée, fibre optique, câble coaxial | Paire torsadée, fibre optique |
| **Codage** | NRZI | | |
| **Longueur maximale** | 13,2 km | 2800 m (CSMA/CD) 10 km (CSMA/CDR) | 500 |
| **Débit** | 62 Kbit/s à 2,4 Mbits/s | 2, 10 et 100 Mbit/s | 5 et 10 Mbit/s |
| **Accès au médium** | Maître/esclave | CSMA/CD et CSMA/DCR | Jeton ISO 8802-4 |
| **Nombre maximal de stations** | 250 | 128 | |
| **Couches OSI** | 1, 2 et 7 | 1 à 4 et 7 | 1, 2 et 7 |

| | LAC | HART | VAN |
|---|---|---|---|
| **Norme** | COMPEX | ROSEMOUNT | ISO 11519-2 |
| **Topologie** | Bus | Bus | Bus |
| **Support de transmision** | Paire torsadée, câble coaxial | Paire torsadée | Paire torsadée, fibre optiqu |
| **Codage** | | Bell 202 | Manchester |
| **Longueur maximale** | 8 km | 3000 m | |
| **Débit** | 50 et 250 Kbauds | 1200 Bauds | 1 Mbit/s |
| **Accès au médium** | CSMA CA/CD | Maître/esclave | Multi-maîtres multi-esclaves |
| **Nombre maximal de stations** | 252 | 15 | |
| **Couches OSI** | | 1, 2 et 7 | 1, 2 et 7 |

| | Data Highway | Can | InterBus-S |
|---|---|---|---|
| **Norme** | ALLEN-BRADLEY | ISO 11519-1 | PHOENIX CONTACT |
| **Topologie** | Bus | Bus | Anneau |
| **Support de transmision** | Paire torsadée | | Paire torsadée, fibre optiqu infrarouge |
| **Codage** | RS-232-C ou RS-422-A | NRZ | RS-485 |
| **Longueur maximale** | 3000 m | 40 m | 400 m |
| **Débit** | 57,6 Kbauds | 125 Kbit/s à 1 Mbit/s | 500 Kbit/s |
| **Accès au médium** | Maître flottant | CSMA/CD | Maitre/esclave |
| **Nombre maximal de stations** | 255 | | 256 |
| **Couches OSI** | 1, 2 et 7 | 1 et 2 | 1, 2 et 7 |

| | EFIWAY | FILBUS |
|---|---|---|
| **Norme** | | |
| **Topologie** | Anneau | Bus |
| **Support de transmision** | Fibre optique | Paire torsadée |
| **Codage** | | NRZI |
| **Longueur maximale** | 52 km | |
| **Débit** | | 375 Kbit/s |
| **Accès au médium** | Accès déterministe par tranches temporelles | Maître/esclave |
| **Nombre maximal de stations** | 64 | 254 |
| **Couches OSI** | 1, 2 et 7 | 1, 2 et 7 |

# Annexe 3 : Généralités sur les graphes

Un graphe G(A, U) est caractérisé par :

- d'un ensemble A dont les éléments sont appelés des sommets.
- d'un ensemble U dont les éléments $u \in U$ sont :
- soit des couples ordonnés de sommets appelés des arcs. Si u=(i,j) est un arc de G, i est l'extrémité initiale de u et j l'extrémité terminale de u. Le graphe est ainsi qualifié d'orienté.
- soit des couples non ordonnés de sommets appelés des arêtes. Le graphe est ainsi qualifié de non orienté.

Un multigraphe est un graphe pour lequel il peut exister plusieurs arêtes entre deux sommets i et j donnés.

Un graphe est dit simple si :

- il est sans boucles,
- il n'y a jamais plus d'une arête entre deux sommets quelconques.

La matrice d'incidence sommets-arêtes de G est une matrice à coefficients 0 ou 1, où chaque ligne i correspond à un sommet i de G, et chaque colonne à une arête u=(i,j) de G.

## Chemin d'un graphe

Un chemin de longueur q est une séquence de q arcs $(u_1, u_2, ..., u_q)$. L'extrémité initiale de $u_i$ doit être l'extrémité terminale de $u_{i-1}$ si $i > 1$, et l'extrémité terminale de $u_i$ doit être l'extrémité initiale de $u_{i+1}$ si $i < q$. Un chemin fermé est un circuit. Un arc peut être vu comme un chemin de longueur 1, et une boucle comme un circuit de longueur 1.

On appelle chemin élémentaire un chemin telle qu'en le parcourant, on ne rencontre pas deux fois le même sommet.

## Chaîne d'un graphe

Une chaîne de longueur q est une séquence de q arêtes ($e_1$, $e_2$, ..., $e_q$), $e_i$ a une extrémité commune avec $e_{i-1}$ si i > 1, et une autre avec $e_{i+1}$ si i < q. Une chaîne fermée est un cycle. Une arête est une chaîne de longueur 1, une boucle est un cycle de longueur 1. On peut considérer des chaînes sur un graphe orienté, en parlant en fait du graphe non orientée associé.

On appelle chaîne élémentaire une chaîne telle qu'en la parcourant, on ne rencontre pas deux fois le même sommet.

## Connexité simple d'un graphe

Un graphe G est connexe s'il existe une chaîne entre toute paire de sommets. Le graphe est donc "d'un seul bloc". Si G n'est pas connexe, on peut identifier plusieurs sous-graphes connexes, maximaux au sens de l'inclusion, appelés composantes connexes. Plus précisément, ces composantes sont les classes d'équivalence de la relation d'équivalence (réflexivité, symétrie, transitivité) binaire entre deux sommets (il existe une chaîne entre deux sommets). La figure 1 décrit des exemples de graphes connexes et non connexes.

Les classes d'équivalence induites sur A par cette relation forment une partition de A en p partitions. Le nombre p de classes d'équivalence distinctes est appelé le nombre de connexité du graphe.

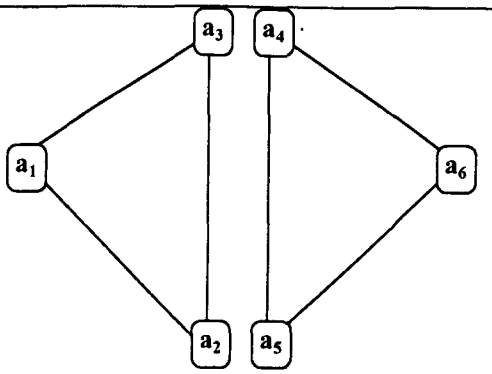



**Figure 1a**

*Graphe non connexe*

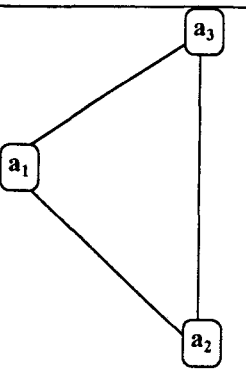



**Figure 1b**

*Graphe connexe*

## Connexité forte d'un graphe

Un graphe orienté G est fortement connexe (figure 2) si pour toute paire de sommets distincts $\{a_i, a_j\}$ il existe un chemin de $a_i$ à $a_j$ et un autre de $a_j$ à $a_i$ : $a_i$ et $a_j$ sont donc sur un circuit. Cette propriété orientée est plus forte que la connexité. Les composantes fortement connexes d'un graphe G sont les classes d'équivalence de la relation binaire entre deux sommets (il existe un circuit entre les deux sommets).

Les classes d'équivalence induites sur A par cette relation forment une partition de A en q partitions. Le nombre q de classes d'équivalence distinctes est appelé le nombre de connexité forte du graphe.

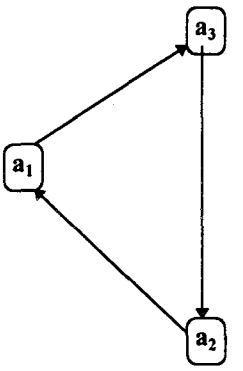



**Figure 2**

*Graphe fortement connexe*

# Annexe 4 : Application de l'algorithme mixte

# Architecture fonctionnelle d'une application de défense aérienne

## 1 Architecture fonctionnelle d'une application de défense aérienne

L'architecture fonctionnelle de l'application de défense aérienne (figure 1) issue de [Ma 82] peut être décrite par un graphe G(A, U). L'ensemble des sommets représente les atomes de distribution, et les arcs représentent les liens de transfert d'information entre ces noeuds. Les arcs devaient être pondérés par les coûts de communication, mais pour simplifier le schéma présenté ci-dessus, nous avons décrit ces coûts dans une matrice associée à G(A, U).

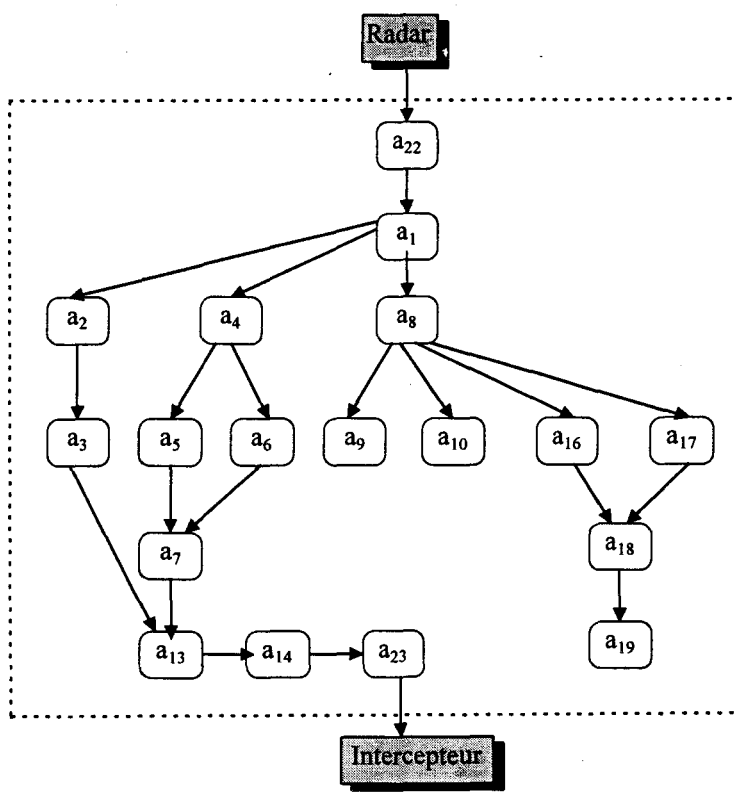



**Figure 1**

*L'application de défense aérienne*

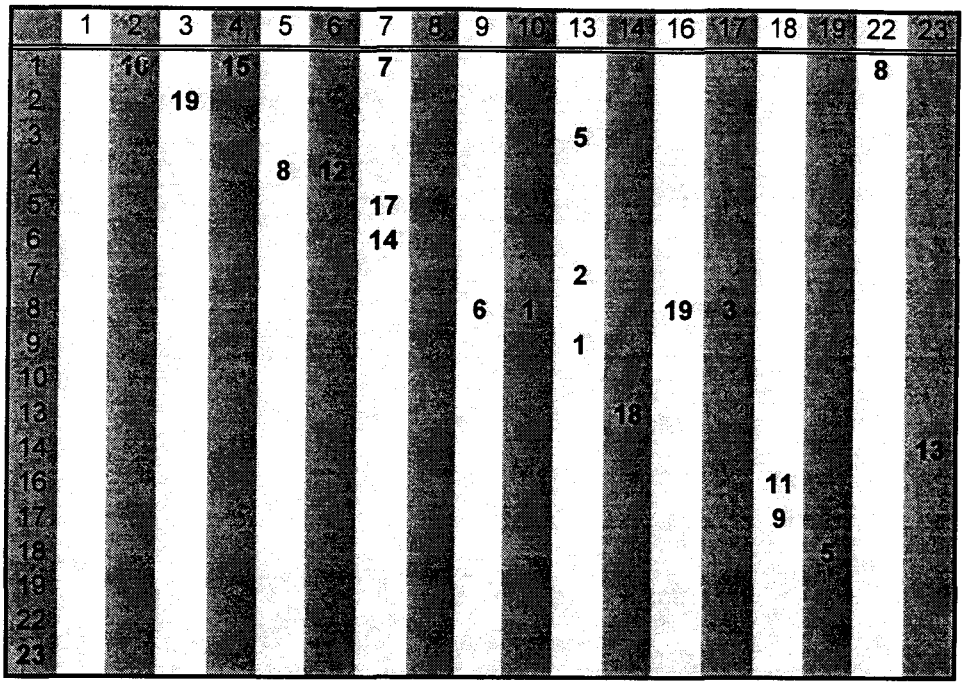


**Figure 2**

*Matrice des coûts*

## 2 Codification de l'algorithme mixte

L'algorithme mixte a été codifié en langage C et implémenté sur un micro-ordinateur DX4 100. Le programme principal est présenté à la fin de cette annexe.

Le programme ayant pour nom JAK peut s'exécuter avec les option suivantes :

- -iFileName1 : FileName1 sera le fichier des données initiales, si cette option n'est pas fournie le fichier de données sera par défaut DATA.INI
- -oFileName2 : FileName2 sera le fichier de sortie, si cette option n'est pas fournie le fichier de sortie sera par défaut DATA.OUT,

- -l[0|1|2|3] : Fixe le niveau de détails dans la trace de l'algorithme, ainsi 0 indiquera le minimum de détails, et 3 le maximum de détails,
- -m[p|a] : Fixe le sens de démarrage de l'algorithme mixte, ainsi avec l'option "p" l'algorithme démarrera par la décomposition, puis entamera l'agrégation, tandis que avec l'option "a" il démarrera par l'agrégation puis entamera la décomposition. L'option par défaut est p.

## 3 Application de JAK à l'application de défense aérienne

### 3.1 Solution initiale

La solution initiale est décrite dans le fichier d'entrée qu'on a appelé **radar.ini**. Ce fichier peut contenir des commentaires identifiés par le symbole #. Il doit contenir la liste des atomes de distribution, le nombre des partitions initiales, la liste des de ces partitions, et finalement la matrice des coûts donnée sous forme symétrique. Les contraintes fournies sont celles relatives aux regroupements et séparations obligatoires.

**Le fichier d'entrée radar.ini**

```
# Fichier des données initiales

# Les lignes de commentaire doivent commencer par #
# La ligne suivante donne les noms de tous les éléments
# Les noms doivent être séparés par une virgule, un espace, une tabulation
# ou un point-virgule.

a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a13,a14,a16,a17,a18,a19,a22,a23

# Les lignes suivantes définissent la partition initiale.
# La première ligne indique le nombre d'ensembles de la partition.
# Chaque ligne suivante définit un ensemble de la partition.

7
a1,a2
a4,a8
a9,a10,a16
a13,a17
a3,a5,a6
a7,a14,a18
a19,a23
```

# Les lignes suivantes définissent les contraintes de séparation
# La première ligne indique le nombre de contraintes

2
a1,a6,a7
a8,a16

# Les lignes suivantes définissent les contraintes de regroupement
# La première ligne indique le nombre de contraintes

2
a3,a5
a9,a10,a16

# Les lignes suivantes définissent la matrice des coûts
# Les éléments sont supposés apparaître dans le même ordre qu'à la première ligne
# N'indiquer que la partie triangulaire supérieure avec la diagonale

```
0 10 0 15 0 0 7 0 0 0 0 0 0 0 0 0 8 0
  0 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0
      0 8 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 17 0 0 0 0 0 0 0 0 0 0 0 0 0
          0 14 0 0 0 0 0 0 0 0 0 0 0 0 0
            0 0 0 0 2 0 0 0 0 0 0 0 0 0 0
              0 6 1 0 0 19 3 0 0 0 0 0 0 0
                0 0 1 0 0 0 0 0 0 0 0 0 0 0
                  0 0 0 0 0 0 0 0 0 0 0 0 0
                    0 18 0 0 0 0 0 0 0 0 0
                      0 0 0 0 0 0 13
                        0 0 11 0 0 0 0
                          0 9 0 0 0 0
                            0 5 0 0 0
                              0 0 0 0
                                0 0 0
                                  0
```

# Fin du fichier

## 3.2 Solution finale

### 3.2.1 R1 puis R2

L'application JAK est lancé de la manière suivante :

JAK -iradar.ini -oradar.a -ma

Ainsi le programme commencera par R1 puis R2.

**Le fichier de sortie radar.a**

```
* Données initiales
   * Ensemble des éléments
   { a23 a22 a19 a18 a17 a16 a14 a13 a10 a9 a8 a7 a6 a5 a4 a3 a2 a1 }
   * Partition initiale
   {
      { a23 a19 }
      { a18 a14 a7 }
      { a6 a5 a3 }
      { a17 a13 }
      { a16 a10 a9 }
      { a8 a4 }
      { a2 a1 }
   }
   * Contraintes de séparation
   {
      { a16 a8 }
      { a7 a6 a1 }
   }
   * Contraintes d'agrégation
   {
      { a16 a10 a9 }
      { a5 a3 }
   }
   * Matrice de coût initiale
   [
   0  10 0  15 0  0  7  0  0  0  0  0  0  0  0  0  8  0
   10 0  19 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  19 0  0  0  0  0  0  0  0  5  0  0  0  0  0  0  0
   15 0  0  0  8  12 0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  8  0  0  17 0  0  0  0  0  0  0  0  0  0  0
   0  0  0  12 0  0  14 0  0  0  0  0  0  0  0  0  0  0
   7  0  0  0  17 14 0  0  0  0  2  0  0  0  0  0  0  0
   0  0  0  0  0  0  0  0  6  1  0  0  19 3  0  0  0  0
   0  0  0  0  0  0  0  6  0  0  1  0  0  0  0  0  0  0
   0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
   0  0  5  0  0  0  2  0  1  0  0  18 0  0  0  0  0  0
   0  0  0  0  0  0  0  0  0  0  18 0  0  0  0  0  0  13
   0  0  0  0  0  0  0  19 0  0  0  0  0  0  0  11 0  0  0
   0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  9  0  0  0
   0  0  0  0  0  0  0  0  0  0  0  0  0  11 9  0  5  0  0
   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  5  0  0  0
   8  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0  0  0  0  0  0  0  13 0  0  0  0  0  0
   ]
* Boucle externe
   * Partition initiale
   {
      { a1 a2 }
      { a4 a8 }
      { a9 a10 a16 }
      { a13 a17 }
      { a3 a5 a6 }
      { a7 a14 a18 }
      { a19 a23 }
   }
   * Coût : 185
   * Boucle d'agrégation - décomposition
```

```
* Partition initiale
{
    { a1 a2 }
    { a4 a8 }
    { a9 a10 a16 }
    { a13 a17 }
    { a3 a5 a6 }
    { a7 a14 a18 }
    { a19 a23 }
}
* Coût : 185
* Agrégation - décomposition
* Agrégation
    * Ensembles agrégés
    { a13 a17 }
    { a7 a14 a18 }
    * Agrégation obtenue
    { a18 a14 a7 a17 a13 }
    * Diminution du coût : 29
    * Nouvelle partition
    {
        { a18 a14 a7 a17 a13 }
        { a23 a19 }
        { a6 a5 a3 }
        { a16 a10 a9 }
        { a8 a4 }
        { a2 a1 }
    }
    * Coût : 156
* Décomposition
    * Ensemble partitionné
    { a18 a14 a7 a17 a13 }
    * Décompositionde cet ensemble
    { a17 a18 }
    { a13 a7 a14 }
    * Augmentation du coût : 0
    * Nouvelle partition
    {
        { a13 a7 a14 }
        { a17 a18 }
        { a1 a2 }
        { a4 a8 }
        { a9 a10 a16 }
        { a3 a5 a6 }
        { a19 a23 }
    }
    * Coût : 156
* Partition obtenue
{
    { a13 a7 a14 }
    { a17 a18 }
    { a1 a2 }
    { a4 a8 }
    { a9 a10 a16 }
    { a3 a5 a6 }
    { a19 a23 }
}
* Coût : 156
* Agrégation - décomposition
* Agrégation
```

```
        * Ensembles agrégés
        { a4 a8 }
        { a3 a5 a6 }
        * Agrégation obtenue
        { a6 a5 a3 a8 a4 }
        * Diminution du coût : 20
        * Nouvelle partition
        {
            { a6 a5 a3 a8 a4 }
            { a23 a19 }
            { a16 a10 a9 }
            { a2 a1 }
            { a18 a17 }
            { a14 a7 a13 }
        }
        * Coût : 136
    * Décomposition
        * Ensemble partitionné
        { a6 a5 a3 a8 a4 }
        * Décompositionde cet ensemble
        { a8 }
        { a4 a3 a5 a6 }
        * Augmentation du coût : 0
        * Nouvelle partition
        {
            { a4 a3 a5 a6 }
            { a8 }
            { a13 a7 a14 }
            { a17 a18 }
            { a1 a2 }
            { a9 a10 a16 }
            { a19 a23 }
        }
        * Coût : 136
    * Partition obtenue
    {
        { a4 a3 a5 a6 }
        { a8 }
        { a13 a7 a14 }
        { a17 a18 }
        { a1 a2 }
        { a9 a10 a16 }
        { a19 a23 }
    }
    * Coût : 136
    * Agrégation - décomposition
    * Agrégation
        * Ensembles agrégés
        { a13 a7 a14 }
        { a19 a23 }
        * Agrégation obtenue
        { a23 a19 a14 a7 a13 }
        * Diminution du coût : 13
        * Nouvelle partition
        {
            { a23 a19 a14 a7 a13 }
            { a16 a10 a9 }
            { a2 a1 }
            { a18 a17 }
            { a8 }
```

```
        { a6 a5 a3 a4 }
    }
    * Coût : 123
* Décomposition
    * Ensemble partitionné
    { a23 a19 a14 a7 a13 }
    * Décompositionde cet ensemble
    { a19 }
    { a13 a7 a14 a23 }
    * Augmentation du coût : 0
    * Nouvelle partition
    {
        { a13 a7 a14 a23 }
        { a19 }
        { a4 a3 a5 a6 }
        { a8 }
        { a17 a18 }
        { a1 a2 }
        { a9 a10 a16 }
    }
    * Coût : 123
* Partition obtenue
{
    { a13 a7 a14 a23 }
    { a19 }
    { a4 a3 a5 a6 }
    { a8 }
    { a17 a18 }
    { a1 a2 }
    { a9 a10 a16 }
}
* Coût : 123
* Agrégation - décomposition
* Agrégation
    * Ensembles agrégés
    { a17 a18 }
    { a9 a10 a16 }
    * Agrégation obtenue
    { a16 a10 a9 a18 a17 }
    * Diminution du coût : 11
    * Nouvelle partition
    {
        { a16 a10 a9 a18 a17 }
        { a2 a1 }
        { a8 }
        { a6 a5 a3 a4 }
        { a19 }
        { a23 a14 a7 a13 }
    }
    * Coût : 112
* Décomposition
    * Ensemble partitionné
    { a23 a14 a7 a13 }
    * Décompositionde cet ensemble
    { a7 }
    { a13 a14 a23 }
    * Augmentation du coût : 2
    * Nouvelle partition
    {
        { a13 a14 a23 }
```

```
        { a7 }
        { a19 }
        { a4 a3 a5 a6 }
        { a8 }
        { a1 a2 }
        { a17 a18 a9 a10 a16 }
    }
    * Coût : 114
* Partition obtenue
{
    { a13 a14 a23 }
    { a7 }
    { a19 }
    { a4 a3 a5 a6 }
    { a8 }
    { a1 a2 }
    { a17 a18 a9 a10 a16 }
}
* Coût : 114
* Agrégation - décomposition
* Agrégation
    * Ensembles agrégés
    { a13 a14 a23 }
    { a4 a3 a5 a6 }
    * Agrégation obtenue
    { a6 a5 a3 a4 a23 a14 a13 }
    * Diminution du coût : 5
    * Nouvelle partition
    {
        { a6 a5 a3 a4 a23 a14 a13 }
        { a16 a10 a9 a18 a17 }
        { a2 a1 }
        { a8 }
        { a19 }
        { a7 }
    }
    * Coût : 109
* Décomposition
    * Ensemble partitionné
    { a6 a5 a3 a4 a23 a14 a13 }
    * Décompositionde cet ensemble
    { a4 a3 a5 a6 }
    { a13 a14 a23 }
    * Augmentation du coût : 5
    * Nouvelle partition
    {
        { a13 a14 a23 }
        { a4 a3 a5 a6 }
        { a7 }
        { a19 }
        { a8 }
        { a1 a2 }
        { a17 a18 a9 a10 a16 }
    }
    * Coût : 114
* Partition obtenue
{
    { a13 a14 a23 }
    { a4 a3 a5 a6 }
    { a7 }
```

{ a19 }
{ a8 }
{ a1 a2 }
{ a17 a18 a9 a10 a16 }
}
* Coût : 114
* Stabilisation sur la partition
{
{ a16 a10 a9 a18 a17 }
{ a2 a1 }
{ a8 }
{ a6 a5 a3 a4 }
{ a19 }
{ a7 }
{ a23 a14 a13 }
}
* Coût : 114
* Boucle de décomposition- agrégation
* Partition initiale
{
{ a13 a14 a23 }
{ a7 }
{ a19 }
{ a4 a3 a5 a6 }
{ a8 }
{ a1 a2 }
{ a17 a18 a9 a10 a16 }
}
* Coût : 114
* Décomposition- agrégation
* Décomposition
* Ensemble partitionné
{ a4 a3 a5 a6 }
* Décompositionde cet ensemble
{ a5 a3 }
{ a6 a4 }
* Augmentation du coût : 8
* Nouvelle partition
{
{ a6 a4 }
{ a5 a3 }
{ a16 a10 a9 a18 a17 }
{ a2 a1 }
{ a8 }
{ a19 }
{ a7 }
{ a23 a14 a13 }
}
* Coût : 122
* Agrégation
* Ensembles agrégés
{ a5 a3 }
{ a2 a1 }
* Agrégation obtenue
{ a1 a2 a3 a5 }
* Diminution du coût : 19
* Nouvelle partition
{
{ a1 a2 a3 a5 }
{ a13 a14 a23 }

{ a7 }
{ a19 }
{ a8 }
{ a17 a18 a9 a10 a16 }
{ a4 a6 }
}
* Coût : 103
* Partition obtenue
{
{ a1 a2 a3 a5 }
{ a13 a14 a23 }
{ a7 }
{ a19 }
{ a8 }
{ a17 a18 a9 a10 a16 }
{ a4 a6 }
}
* Coût : 103
* Décomposition- agrégation
* Décomposition
* Ensemble partitionné
{ a17 a18 a9 a10 a16 }
* Décompositionde cet ensemble
{ a17 }
{ a16 a10 a9 a18 }
* Augmentation du coût : 9
* Nouvelle partition
{
{ a16 a10 a9 a18 }
{ a17 }
{ a6 a4 }
{ a8 }
{ a19 }
{ a7 }
{ a23 a14 a13 }
{ a5 a3 a2 a1 }
}
* Coût : 112
* Agrégation
* Ensembles agrégés
{ a16 a10 a9 a18 }
{ a17 }
* Agrégation obtenue
{ a17 a18 a9 a10 a16 }
* Diminution du coût : 9
* Nouvelle partition
{
{ a17 a18 a9 a10 a16 }
{ a1 a2 a3 a5 }
{ a13 a14 a23 }
{ a7 }
{ a19 }
{ a8 }
{ a4 a6 }
}
* Coût : 103
* Partition obtenue
{
{ a17 a18 a9 a10 a16 }
{ a1 a2 a3 a5 }

```
                { a13 a14 a23 }
                { a7 }
                { a19 }
                { a8 }
                { a4 a6 }
            }
            * Coût : 103
            * Stabilisation sur la partition
            {
                { a6 a4 }
                { a16 a10 a9 a18 a17 }
                { a8 }
                { a19 }
                { a7 }
                { a23 a14 a13 }
                { a5 a3 a2 a1 }
            }
            * Coût : 103
* Boucle externe
    * Partition initiale
    {
        { a1 a2 a3 a5 }
        { a13 a14 a23 }
        { a7 }
        { a19 }
        { a8 }
        { a17 a18 a9 a10 a16 }
        { a4 a6 }
    }
    * Coût : 103
    * Boucle d'agrégation - décomposition
        * Partition initiale
        {
            { a1 a2 a3 a5 }
            { a13 a14 a23 }
            { a7 }
            { a19 }
            { a8 }
            { a17 a18 a9 a10 a16 }
            { a4 a6 }
        }
        * Coût : 103
        * Agrégation - décomposition
        * Agrégation
            * Ensembles agrégés
            { a1 a2 a3 a5 }
            { a13 a14 a23 }
            * Agrégation obtenue
            { a23 a14 a13 a5 a3 a2 a1 }
            * Diminution du coût : 5
            * Nouvelle partition
            {
                { a23 a14 a13 a5 a3 a2 a1 }
                { a6 a4 }
                { a16 a10 a9 a18 a17 }
                { a8 }
                { a19 }
                { a7 }
            }
            * Coût : 98
```

* Décomposition
  * Ensemble partitionné
  { a23 a14 a13 a5 a3 a2 a1 }
  * Décompositionde cet ensemble
  { a13 a14 a23 }
  { a1 a2 a3 a5 }
  * Augmentation du coût : 5
  * Nouvelle partition
  {
    { a1 a2 a3 a5 }
    { a13 a14 a23 }
    { a7 }
    { a19 }
    { a8 }
    { a17 a18 a9 a10 a16 }
    { a4 a6 }
  }
  * Coût : 103
* Partition obtenue
{
  { a1 a2 a3 a5 }
  { a13 a14 a23 }
  { a7 }
  { a19 }
  { a8 }
  { a17 a18 a9 a10 a16 }
  { a4 a6 }
}
* Coût : 103
* Stabilisation sur la partition
{
  { a6 a4 }
  { a16 a10 a9 a18 a17 }
  { a8 }
  { a19 }
  { a7 }
  { a23 a14 a13 }
  { a5 a3 a2 a1 }
}
* Coût : 103
* Boucle de décomposition- agrégation
* Partition initiale
{
  { a1 a2 a3 a5 }
  { a13 a14 a23 }
  { a7 }
  { a19 }
  { a8 }
  { a17 a18 a9 a10 a16 }
  { a4 a6 }
}
* Coût : 103
* Décomposition- agrégation
* Décomposition
  * Ensemble partitionné
  { a17 a18 a9 a10 a16 }
  * Décompositionde cet ensemble
  { a17 }
  { a16 a10 a9 a18 }
  * Augmentation du coût : 9

```
            * Nouvelle partition
            {
                { a16 a10 a9 a18 }
                { a17 }
                { a6 a4 }
                { a8 }
                { a19 }
                { a7 }
                { a23 a14 a13 }
                { a5 a3 a2 a1 }
            }
            * Coût : 112
        * Agrégation
            * Ensembles agrégés
            { a16 a10 a9 a18 }
            { a17 }
            * Agrégation obtenue
            { a17 a18 a9 a10 a16 }
            * Diminution du coût : 9
            * Nouvelle partition
            {
                { a17 a18 a9 a10 a16 }
                { a1 a2 a3 a5 }
                { a13 a14 a23 }
                { a7 }
                { a19 }
                { a8 }
                { a4 a6 }
            }
            * Coût : 103
        * Partition obtenue
        {
            { a17 a18 a9 a10 a16 }
            { a1 a2 a3 a5 }
            { a13 a14 a23 }
            { a7 }
            { a19 }
            { a8 }
            { a4 a6 }
        }
        * Coût : 103
        * Stabilisation sur la partition
        {
            { a6 a4 }
            { a16 a10 a9 a18 a17 }
            { a8 }
            { a19 }
            { a7 }
            { a23 a14 a13 }
            { a5 a3 a2 a1 }
        }
        * Coût : 103
* Stabilisation sur la partition
{
    { a6 a4 }
    { a16 a10 a9 a18 a17 }
    { a8 }
    { a19 }
    { a7 }
    { a23 a14 a13 }
```

```
        { a5 a3 a2 a1 }
    }
    * Coût : 103
* Partition finale
{
    { a6 a4 }
    { a16 a10 a9 a18 a17 }
    { a8 }
    { a19 }
    { a7 }
    { a23 a14 a13 }
    { a5 a3 a2 a1 }
}
```

*** Coût : 103**
*** Nombre de tours de boucle externe : 2**
*** Nombre de tours de boucle décomposition - agrégation : 3**
*** Nombre de tours de boucle agrégation - décomposition : 6**
*** Temps de calcul approximatif (en ms) : 26**

### 3.2.1 R2 puis R1

L'application JAK est lancé de la manière suivante :

JAK -iradar.ini -oradar.p -mp

Ainsi le programme commencera par R2 puis R1.

```
* Données initiales
    * Ensemble des éléments
    { a23 a22 a19 a18 a17 a16 a14 a13 a10 a9 a8 a7 a6 a5 a4 a3 a2 a1 }
    * Partition initiale
    {
        { a23 a19 }
        { a18 a14 a7 }
        { a6 a5 a3 }
        { a17 a13 }
        { a16 a10 a9 }
        { a8 a4 }
        { a2 a1 }
    }
    * Contraintes de séparation
    {
        { a16 a8 }
        { a7 a6 a1 }
    }
    * Contraintes d'agrégation
    {
        { a16 a10 a9 }
        { a5 a3 }
    }
    * Matrice de coût initiale
    [
    0  10 0  15 0  0  7  0  0  0  0  0  0  0  0  0  8  0
    10 0  19 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
    0  19 0  0  0  0  0  0  0  0  5  0  0  0  0  0  0  0
    15 0  0  0  8  12 0  0  0  0  0  0  0  0  0  0  0  0
    0  0  0  8  0  0  17 0  0  0  0  0  0  0  0  0  0  0
    0  0  0  12 0  0  14 0  0  0  0  0  0  0  0  0  0  0
```

```
 7  0  0  0 17 14  0  0  0  0  2  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  6  1  0  0 19  3  0  0  0  0
 0  0  0  0  0  0  0  6  0  0  1  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
 0  0  5  0  0  0  2  0  1  0  0 18  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0 18  0  0  0  0  0  0 13
 0  0  0  0  0  0  0 19  0  0  0  0  0  0 11  0  0  0
 0  0  0  0  0  0  0  3  0  0  0  0  0  0  9  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0 11  9  0  5  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  5  0  0  0
 8  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0 13  0  0  0  0  0  0
 ]
* Boucle externe
   * Partition initiale
   {
      { a1 a2 }
      { a4 a8 }
      { a9 a10 a16 }
      { a13 a17 }
      { a3 a5 a6 }
      { a7 a14 a18 }
      { a19 a23 }
   }
   * Coût : 185
   * Boucle de décomposition- agrégation
      * Partition initiale
      {
         { a1 a2 }
         { a4 a8 }
         { a9 a10 a16 }
         { a13 a17 }
         { a3 a5 a6 }
         { a7 a14 a18 }
         { a19 a23 }
      }
      * Coût : 185
      * Décomposition- agrégation
      * Décomposition
         * Ensemble partitionné
         { a4 a8 }
         * Décompositionde cet ensemble
         { a4 }
         { a8 }
         * Augmentation du coût : 0
         * Nouvelle partition
         {
            { a8 }
            { a4 }
            { a23 a19 }
            { a18 a14 a7 }
            { a6 a5 a3 }
            { a17 a13 }
            { a16 a10 a9 }
            { a2 a1 }
         }
         * Coût : 185
      * Agrégation
         * Ensembles agrégés
         { a18 a14 a7 }
```

{ a17 a13 }
* Agrégation obtenue
{ a13 a17 a7 a14 a18 }
* Diminution du coût : 29
* Nouvelle partition
{
{ a13 a17 a7 a14 a18 }
{ a1 a2 }
{ a9 a10 a16 }
{ a3 a5 a6 }
{ a19 a23 }
{ a4 }
{ a8 }
}
* Coût : 156
* Partition obtenue
{
{ a13 a17 a7 a14 a18 }
{ a1 a2 }
{ a9 a10 a16 }
{ a3 a5 a6 }
{ a19 a23 }
{ a4 }
{ a8 }
}
* Coût : 156
* Décomposition- agrégation
* Décomposition
* Ensemble partitionné
{ a13 a17 a7 a14 a18 }
* Décompositionde cet ensemble
{ a14 a7 a13 }
{ a18 a17 }
* Augmentation du coût : 0
* Nouvelle partition
{
{ a18 a17 }
{ a14 a7 a13 }
{ a8 }
{ a4 }
{ a23 a19 }
{ a6 a5 a3 }
{ a16 a10 a9 }
{ a2 a1 }
}
* Coût : 156
* Agrégation
* Ensembles agrégés
{ a4 }
{ a6 a5 a3 }
* Agrégation obtenue
{ a3 a5 a6 a4 }
* Diminution du coût : 20
* Nouvelle partition
{
{ a3 a5 a6 a4 }
{ a1 a2 }
{ a9 a10 a16 }
{ a19 a23 }
{ a8 }

```
        { a13 a7 a14 }
        { a17 a18 }
    }
    * Coût : 136
* Partition obtenue
{
    { a3 a5 a6 a4 }
    { a1 a2 }
    { a9 a10 a16 }
    { a19 a23 }
    { a8 }
    { a13 a7 a14 }
    { a17 a18 }
}
* Coût : 136
* Décomposition- agrégation
* Décomposition
    * Ensemble partitionné
    { a19 a23 }
    * Décompositionde cet ensemble
    { a19 }
    { a23 }
    * Augmentation du coût : 0
    * Nouvelle partition
    {
        { a23 }
        { a19 }
        { a18 a17 }
        { a14 a7 a13 }
        { a8 }
        { a16 a10 a9 }
        { a2 a1 }
        { a4 a6 a5 a3 }
    }
    * Coût : 136
* Agrégation
    * Ensembles agrégés
    { a23 }
    { a14 a7 a13 }
    * Agrégation obtenue
    { a13 a7 a14 a23 }
    * Diminution du coût : 13
    * Nouvelle partition
    {
        { a13 a7 a14 a23 }
        { a3 a5 a6 a4 }
        { a1 a2 }
        { a9 a10 a16 }
        { a8 }
        { a17 a18 }
        { a19 }
    }
    * Coût : 123
* Partition obtenue
{
    { a13 a7 a14 a23 }
    { a3 a5 a6 a4 }
    { a1 a2 }
    { a9 a10 a16 }
    { a8 }
```

```
    { a17 a18 }
    { a19 }
}
* Coût : 123
* Décomposition- agrégation
* Décomposition
    * Ensemble partitionné
    { a13 a7 a14 a23 }
    * Décompositionde cet ensemble
    { a7 }
    { a23 a14 a13 }
    * Augmentation du coût : 2
    * Nouvelle partition
    {
        { a23 a14 a13 }
        { a7 }
        { a19 }
        { a18 a17 }
        { a8 }
        { a16 a10 a9 }
        { a2 a1 }
        { a4 a6 a5 a3 }
    }
    * Coût : 125
* Agrégation
    * Ensembles agrégés
    { a18 a17 }
    { a16 a10 a9 }
    * Agrégation obtenue
    { a9 a10 a16 a17 a18 }
    * Diminution du coût : 11
    * Nouvelle partition
    {
        { a9 a10 a16 a17 a18 }
        { a3 a5 a6 a4 }
        { a1 a2 }
        { a8 }
        { a19 }
        { a7 }
        { a13 a14 a23 }
    }
    * Coût : 114
* Partition obtenue
{
    { a9 a10 a16 a17 a18 }
    { a3 a5 a6 a4 }
    { a1 a2 }
    { a8 }
    { a19 }
    { a7 }
    { a13 a14 a23 }
}
* Coût : 114
* Décomposition- agrégation
* Décomposition
    * Ensemble partitionné
    { a3 a5 a6 a4 }
    * Décompositionde cet ensemble
    { a5 a3 }
    { a4 a6 }
```

```
        * Augmentation du coût : 8
        * Nouvelle partition
        {
            { a4 a6 }
            { a5 a3 }
            { a23 a14 a13 }
            { a7 }
            { a19 }
            { a8 }
            { a2 a1 }
            { a18 a17 a16 a10 a9 }
        }
        * Coût : 122
    * Agrégation
        * Ensembles agrégés
        { a5 a3 }
        { a2 a1 }
        * Agrégation obtenue
        { a1 a2 a3 a5 }
        * Diminution du coût : 19
        * Nouvelle partition
        {
            { a1 a2 a3 a5 }
            { a9 a10 a16 a17 a18 }
            { a8 }
            { a19 }
            { a7 }
            { a13 a14 a23 }
            { a6 a4 }
        }
        * Coût : 103
* Partition obtenue
{
    { a1 a2 a3 a5 }
    { a9 a10 a16 a17 a18 }
    { a8 }
    { a19 }
    { a7 }
    { a13 a14 a23 }
    { a6 a4 }
}
* Coût : 103
* Décomposition- agrégation
* Décomposition
    * Ensemble partitionné
    { a9 a10 a16 a17 a18 }
    * Décompositionde cet ensemble
    { a17 }
    { a18 a16 a10 a9 }
    * Augmentation du coût : 9
    * Nouvelle partition
    {
        { a18 a16 a10 a9 }
        { a17 }
        { a4 a6 }
        { a23 a14 a13 }
        { a7 }
        { a19 }
        { a8 }
        { a5 a3 a2 a1 }
```

```
        }
        * Coût : 112
    * Agrégation
        * Ensembles agrégés
        { a18 a16 a10 a9 }
        { a17 }
        * Agrégation obtenue
        { a17 a9 a10 a16 a18 }
        * Diminution du coût : 9
        * Nouvelle partition
        {
            { a17 a9 a10 a16 a18 }
            { a1 a2 a3 a5 }
            { a8 }
            { a19 }
            { a7 }
            { a13 a14 a23 }
            { a6 a4 }
        }
        * Coût : 103
    * Partition obtenue
    {
        { a17 a9 a10 a16 a18 }
        { a1 a2 a3 a5 }
        { a8 }
        { a19 }
        { a7 }
        { a13 a14 a23 }
        { a6 a4 }
    }
    * Coût : 103
    * Stabilisation sur la partition
    {
        { a4 a6 }
        { a23 a14 a13 }
        { a7 }
        { a19 }
        { a8 }
        { a18 a17 a16 a10 a9 }
        { a5 a3 a2 a1 }
    }
    * Coût : 103
* Boucle d'agrégation - décomposition
    * Partition initiale
    {
        { a4 a6 }
        { a23 a14 a13 }
        { a7 }
        { a19 }
        { a8 }
        { a18 a17 a16 a10 a9 }
        { a5 a3 a2 a1 }
    }
    * Coût : 103
    * Agrégation - décomposition
    * Agrégation
        * Ensembles agrégés
        { a1 a2 a3 a5 }
        { a13 a14 a23 }
        * Agrégation obtenue
```

```
        { a23 a14 a13 a5 a3.a2 a1 }
        * Diminution du coût : 5
        * Nouvelle partition
        {
            { a23 a14 a13 a5 a3 a2 a1 }
            { a4 a6 }
            { a7 }
            { a19 }
            { a8 }
            { a18 a17 a16 a10 a9 }
        }
        * Coût : 98
    * Décomposition
        * Ensemble partitionné
        { a23 a14 a13 a5 a3 a2 a1 }
        * Décompositionde cet ensemble
        { a13 a14 a23 }
        { a1 a2 a3 a5 }
        * Augmentation du coût : 5
        * Nouvelle partition
        {
            { a1 a2 a3 a5 }
            { a13 a14 a23 }
            { a9 a10 a16 a17 a18 }
            { a8 }
            { a19 }
            { a7 }
            { a6 a4 }
        }
        * Coût : 103
    * Partition obtenue
    {
        { a1 a2 a3 a5 }
        { a13 a14 a23 }
        { a9 a10 a16 a17 a18 }
        { a8 }
        { a19 }
        { a7 }
        { a6 a4 }
    }
    * Coût : 103
    * Stabilisation sur la partition
    {
        { a4 a6 }
        { a23 a14 a13 }
        { a7 }
        { a19 }
        { a8 }
        { a18 a17 a16 a10 a9 }
        { a5 a3 a2 a1 }
    }
    * Coût : 103
* Stabilisation sur la partition
{
    { a4 a6 }
    { a23 a14 a13 }
    { a7 }
    { a19 }
    { a8 }
    { a18 a17 a16 a10 a9 }
```

```
        { a5 a3 a2 a1 }
    }
    * Coût : 103
* Partition finale
{
    { a4 a6 }
    { a23 a14 a13 }
    { a7 }
    { a19 }
    { a8 }
    { a18 a17 a16 a10 a9 }
    { a5 a3 a2 a1 }
}
```

*** Coût : 103**
*** Nombre de tours de boucle externe : 1**
*** Nombre de tours de boucle décomposition- agrégation : 6**
*** Nombre de tours de boucle agrégation - décomposition: 1**
*** Temps de calcul approximatif (en ms) : 18**

Le tableau suivant présente une comparaison des résultats de l'application des algorithmes dans les deux sens (R2 puis R1 et R1 puis R2).

| | Coût | NTBE | NTR1R2 | NTR2R1 | Temps de calcul |
|---|---|---|---|---|---|
| R1 puis R2 | **103** | **1** | **6** | **1** | **18** |
| R2 puis R1 | **103** | **2** | **3** | **6** | **26** |

NTBE : **Nombre de tours de boucle externe**
NTR1R2 : **Nombre de tours de boucle décomposition- agrégation**
NTR2R1 : **Nombre de tours de boucle agrégation - décomposition**

## 3.3 Programme principal de JAK

```
void Mixte(void)
{
    int LOC_n;
  int LOC_n12;
  int LOC_n21;
    uclock_t LOC_Start;
    long int LOC_TimeLength;
    T_Partition LOC_PrevZ;
    T_Partition LOC_NextZ;
    T_Partition LOC_Z_12;
    T_Partition LOC_Z_21;

    LOC_n = LOC_n12 = LOC_n21 = 0;
    LOC_PrevZ = DuplicatePartition(PUB_InitialPartition);

    LOC_Start = uclock();

    if(PUB_TypeAlgo == _2_then_1_)
    {
        do
        {
            ++LOC_n;
```

```
TraceStr("* Boucle externe\n");
IncTab();
{
    TraceStr("* Partition initiale\n");
    TracePartition(LOC_PrevZ);
    TraceStr("* Coût : ");
    TraceInt(CostOfPartition(LOC_PrevZ));
    TraceStr("\n");

    TraceStr("* Boucle de décomposition - agrégation\n");
    IncTab();
    {
        TraceStr("* Partition initiale\n");
        TracePartition(LOC_PrevZ);
        TraceStr("* Coût : ");
        TraceInt(CostOfPartition(LOC_PrevZ));
        TraceStr("\n");

        while(1)
        {
            ++LOC_n12;
            LOC_NextZ = R1oR2(LOC_PrevZ);

            if(EqualPartition(LOC_NextZ,LOC_PrevZ))
                break;

            FreePartition(LOC_PrevZ);
            LOC_PrevZ = LOC_NextZ;
        }

        LOC_Z_12 = DuplicatePartition(LOC_PrevZ);

        TraceStr("* Stabilisation sur la partition\n");
        TracePartition(LOC_Z_12);
        TraceStr("* Coût : ");
        TraceInt(CostOfPartition(LOC_Z_12));
        TraceStr("\n");
    }
    DecTab();

    TraceStr("* Boucle d'agrégation - décomposition\n");
    {
        IncTab();
        TraceStr("* Partition initiale\n");
        TracePartition(LOC_Z_12);
        TraceStr("* Coût : ");
        TraceInt(CostOfPartition(LOC_Z_12));
        TraceStr("\n");

        while(1)
        {
            ++LOC_n21;
            LOC_NextZ = R2oR1(LOC_PrevZ);

            if(EqualPartition(LOC_NextZ,LOC_PrevZ))
                break;

            FreePartition(LOC_PrevZ);
            LOC_PrevZ = LOC_NextZ;
        }
```

```
                    LOC_Z_21 = DuplicatePartition(LOC_PrevZ);

                    TraceStr("* Stabilisation sur la partition\n");
                    TracePartition(LOC_Z_21);
                    TraceStr("* Coût : ");
                    TraceInt(CostOfPartition(LOC_Z_21));
                    TraceStr("\n");
                }
                DecTab();
            }
            DecTab();
        }
        while(!EqualPartition(LOC_Z_12,LOC_Z_21));
}
else
{
    do
    {
        ++LOC_n;
        TraceStr("* Boucle externe\n");
        IncTab();
        {
            TraceStr("* Partition initiale\n");
            TracePartition(LOC_PrevZ);
            TraceStr("* Coût : ");
            TraceInt(CostOfPartition(LOC_PrevZ));
            TraceStr("\n");


            TraceStr("* Boucle d'agrégation - décomposition\n");
            IncTab();
            {
                TraceStr("* Partition initiale\n");
                TracePartition(LOC_PrevZ);
                TraceStr("* Coût : ");
                TraceInt(CostOfPartition(LOC_PrevZ));
                TraceStr("\n");

                while(1)
                {
                    ++LOC_n21;
                    LOC_NextZ = R2oR1(LOC_PrevZ);

                    if(EqualPartition(LOC_NextZ,LOC_PrevZ))
                        break;

                    FreePartition(LOC_PrevZ);
                    LOC_PrevZ = LOC_NextZ;
                }

                LOC_Z_21 = DuplicatePartition(LOC_PrevZ);

                TraceStr("* Stabilisation sur la partition\n");
                TracePartition(LOC_Z_21);
                TraceStr("* Coût : ");
                TraceInt(CostOfPartition(LOC_Z_21));
                TraceStr("\n");
            }
            DecTab();
```

```
                    TraceStr("* Boucle de décomposition - agrégation\n");
                    IncTab();
                    {
                        TraceStr("* Partition initiale\n");
                        TracePartition(LOC_PrevZ);
                        TraceStr("* Coût : ");
                        TraceInt(CostOfPartition(LOC_PrevZ));
                        TraceStr("\n");

                        while(1)
                        {
                            ++LOC_n12;
                            LOC_NextZ = R1oR2(LOC_PrevZ);

                            if(EqualPartition(LOC_NextZ,LOC_PrevZ))
                                break;

                            FreePartition(LOC_PrevZ);
                            LOC_PrevZ = LOC_NextZ;
                        }

                        LOC_Z_12 = DuplicatePartition(LOC_PrevZ);

                        TraceStr("* Stabilisation sur la partition\n");
                        TracePartition(LOC_Z_12);
                        TraceStr("* Coût : ");
                        TraceInt(CostOfPartition(LOC_Z_12));
                        TraceStr("\n");
                    }
                    DecTab();
                }
                DecTab();
            }
            while(!EqualPartition(LOC_Z_12,LOC_Z_21));
        }

        LOC_TimeLength = (int)((uclock() - LOC_Start)/UCLOCKS_PER_MS);

        IncTab();
        TraceStr("* Stabilisation sur la partition\n");
        TracePartition(LOC_Z_12);
        TraceStr("* Coût : ");
        TraceInt(CostOfPartition(LOC_Z_12));
        TraceStr("\n");
        DecTab();

        TraceStr("* Partition finale\n");
        TracePartition(LOC_Z_12);
        TraceStr("* Coût : ");
        TraceInt(CostOfPartition(LOC_Z_12));
        TraceStr("\n");
        TraceStr("* Nombre de tours de boucle externe : ");
        TraceInt(LOC_n);
        TraceStr("\n");
        TraceStr("* Nombre de tours de boucle décomposition - agrégation : ");
        TraceInt(LOC_n12);
        TraceStr("\n");
        TraceStr("* Nombre de tours de boucle agrégation - décomposition : ");
        TraceInt(LOC_n21);
```

```
        TraceStr("\n");
        TraceStr("* Temps de calcul approximatif (en ms) : ");
        TraceInt((unsigned int)LOC_TimeLength);
}
```

# Sommaire

**Présentation générale**
1 Contexte du travail ..... 8
2 Problématique ..... 8
3 Plan de la thèse ..... 10

**Chapitre I : Les traitements du système automatisé de production**

Introduction ..... 14
I.1 Le système automatisé de production ..... 14
I.2 Architectures d'un système automatisé de production ..... 16
I.3 Les machines de l'architecture matérielle ..... 17
I.3.1 Les instruments intelligents ..... 18
I.3.1.1 Les capteurs intelligents ..... 20
I.3.1.2 Les actionneurs intelligents ..... 22
I.3.2 Instruments intelligents : architecture matérielle ou fonctionnelle ..... 23
I.4 Structuration des fonctions du système d'automatisation ..... 24
I.5 Les traitements du système d'automatisation ..... 31
I.5.1 Les aides à la conduite ..... 31
I.5.1.1 Les traitements de surveillance ..... 32
I.5.1.2 Les traitements de pilotage ..... 33
I.5.1.3 Elaboration d'informations complémentaires ..... 33
I.5.2 Les aides à la maintenance ..... 34
I.5.2.1 Les traitements de surveillance ..... 34
I.5.2.2 Les traitements d'ordonnancement ..... 35
I.5.2.3 Elaboration d'informations complémentaires ..... 35
I.5.3 Les aides à la gestion ..... 35
I.5.3.1 Les aides à la gestion de production ..... 36
I.5.3.2 Les traitements de gestion technique ..... 37
I.6 Conclusion ..... 37

**Chapitre II : Modélisation du système d'automatisation par atomes de distribution**

*Introduction* ..... *41*
*II.1 Structuration du système d'automatisation* ..... *41*
*II.2 Modélisation de l'architecture fonctionnelle du système d'automatisation* ..... *44*
*II.2.1 Les modules* ..... *45*
*II.2.2 Les capsules* ..... *47*
*II.2.3 Les atomes de distribution* ..... *49*
*II.2.3.1 Caractérisation des atomes de distribution* ..... *52*
*II.2.3.1.1 L'activité de traitement d'un atome de distribution* ..... *52*
*II.2.3.1.2 L'activité de gestion d'un atome de distribution* ..... *53*
*II.2.4 Caractérisation des variables* ..... *54*
*II.2.4.1 Les variables produites* ..... *54*
*II.2.4.2 Les variables consommées* ..... *54*
*II.2.5 Classification des atomes de distribution* ..... *55*
*II.2.6 Redondance des atomes de distribution* ..... *56*
*II.2.7 Gestion des états d'un atome de distribution* ..... *58*
*II.2.7.1 Vue externe des états d'un atome de distribution* ..... *59*
*II.2.7.2 Vue interne des états d'un atome de distribution* ..... *60*
*II.2.7.2.1 Suspension* ..... *61*
*II.2.7.2.2 Répétition* ..... *62*
*II.2.7.2.3 Contre ordre* ..... *63*
*II.2.8 Relations entre les atomes de distribution* ..... *64*
*II.2.8.1 Relations d'échanges d'information* ..... *66*
*II.2.8.2 Mesure de la relation d'échanges d'information* ..... *67*
*II.3 Ordonnancement des atomes de distribution* ..... *68*
*II.4 Implantation des atomes* ..... *70*

*II.5 Conclusion* ... 72

**Chapitre III : Des stratégies pour répartir le système d'automatisation**

*Introduction* ... 75
*III.1 Les solutions admissibles* ... 75
*III.2 Classification des stratégies de répartition* ... 76
*III.3 Les critères de répartition* ... 78
*III.3.1 Le coût d'exécution d'un atome de distribution* ... 79
*III.3.2 Les coûts de communication entre atomes de distribution* ... 81
*III.3.3 Le coût d'interférence entre atomes de distribution* ... 84
*III.4 Les contraintes* ... 85
*III.4.1 Les contraintes logiques* ... 86
*III.4.2 Les contraintes physiques* ... 87
*III.5 Fonctions coûts et architecture du système d'automatisation* ... 88
*III.6 Les algorithmes de répartition* ... 89
*III.6.1 Les algorithmes itératifs de recherche locale* ... 90
*III.6.2 Recherche d'un homomorphisme faible entre graphes* ... 92
*III.6.3 Partition des coupes minimales d'un graphe* ... 95
*III.6.5 Les algorithmes génétiques* ... 97
*III.6.6 Autres approches utilisées* ... 100
*III.7 Conclusion* ... 101

**Chapitre IV : La répartition du système d'automatisation : Une approche de classification**

*Introduction* ... 104
*IV.1 Indépendance, hiérarchisation des atomes de distribution* ... 105
*IV.2 Répartition du système d'automatisation* ... 109
*IV.2.1 Les inf-demi-treillis* ... 110
*Proposition* ... 112
*IV.2.2 Les sup-demi-treillis* ... 112
*Proposition* ... 113
*Démonstration* ... 114
*IV.3 Détermination d'une fonction coût* ... 114
*IV.3.1 Définition de la fonction coût* ... 114
*IV.3.2 Les contraintes* ... 117
*IV.4 L'approche de classification* ... 118
*IV.4.1 L'algorithme ascendant (R1)* ... 119
*Complexité de l'algorithme R1* ... 120
*IV.4.2 L'algorithme de décomposition (R2)* ... 121
*Complexité de l'algorithme R2* ... 123
*IV.4.3 L'algorithme mixte* ... 125
*Proposition (i)* ... 126
*Démonstration* ... 126
*Proposition (ii)* ... 127
*Démonstration* ... 127
*Complexité de l'algorithme mixte* ... 128
*IV.5 Illustration* ... 130
*IV.6 Conclusion* ... 134

**Conclusion générale** ... **137**
**Bibliographie** ... **141**
**Annexe 1 : Groupe de travail du projet MESR 2033** ... **150**
**Annexe 2 : Les systèmes de communication** ... **152**
**Annexe 3 : Généralités sur les graphes** ... **167**
**Annexe 4 : Architecture fonctionnelle d'une application de défense aérienne** ... **170**