République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique
Laboratoire de Commande des Processus

Mémoire de Magister

Spécialité : **Automatique**

Option : Automatique des systèmes industriels

Présenté par

BOUGCHICHE Fazia

Ingénieur d'Etat en Automatique de l'UMMTO

Thème

Contribution à l'ordonnancement temps-réel sur des ressources identiques en parallèle

Soutenu publiquement le 18/06/2007devant le jury composé par :

M^r^ M.S BOUCHERIT	**Professeur à l'ENP…….....**	**Président**
M^r^ M. BAKALEM	**Docteur à l'ENP..……...….....**	**Rapporteur**
M^r^ M. TADJINE	**Professeur à l'ENP……...……...**	**Rapporteur**
M^r^ F. BOUDJEMAA	**Professeur à l'ENP…….....**	**Examinateur**
M^r^ B. HEMICI	**Chargé de cours à l'ENP……..…..**	**Examinateur**
M^r^ H. SAARI	**Maître de conférence INSM/Bousmaïl ...**	**Examinateur**

Laboratoire de Commande des Processus, ENP
10, Avenue Hassen Badi, BP. 182, 16200 El Harrach, Alger, Algérie.

Remerciements

Je tiens à remercier Messieurs M. BAKALEM et M. TADJINE pour leur disponibilité tout au long de ce travail, pour leurs précieux conseils et orientations ainsi que la mise à ma disposition de l'espace et du matériel nécessaire à la réalisation de ce mémoire.

J'exprime ma profonde gratitude à Monsieur M.S. BOUCHRIT pour ses conseils et son aide précieuse durant l'élaboration de ce mémoire.

J'exprime, aussi, ma profonde gratitude à Monsieur N. BRAHIMI (Nantes), Monsieur A. DERBALA (Blida) et A. GHARBI (Tunisie) pour leurs précieux conseils et la mise à ma disposition de toute la documentation nécessaire à la réalisation de ce travail.

Je n'oublierai pas de remercier tous les membres du Laboratoire de Commande des Processus.

Je remercie tout particulièrement les membres du jury qui ont accepté de juger ce travail.

Enfin, je remercie tous mes amis et collègues qui m'ont soutenus et tous ceux qui ont contribué à la réalisation de ce mémoire.

Je dédie ce modeste travail à celle qui m'a soutenu tout au long de sa réalisation. A celle qui m'a tout donné sans rien attendre en retour.

A Ma Famille.

Table

Des

Matières

Tables des matières

Figures

Tableaux

Introduction

Générale

Du fait de l'évolution économique et technologique, les systèmes de production industrielle se sont considérablement diversifiés et compliqués. Les études d'ordonnancement, commencées au début des années 50 avec les travaux de Johnson [Joh 54], de Jackson [Jac 55] et de Smith [Smi 56], sont particulièrement fécondes, du fait de la diversité des problèmes d'ordonnancement d'une part, et du fait des nombreux domaines d'applications potentielles d'autre part (production, informatique, administration, etc.) [Car 82].

Selon l'organisation de l'atelier, nous distinguons différents problèmes d'ordonnancement.

Cette étude traite un problème d'ordonnancement temps-réel sur deux ressources en parallèle. Ces dernières, dans ce cas, sont identiques. Plusieurs ressources sont candidates pour l'exécution d'une tâche ne nécessitant qu'une seule, représente le principe des ressources identiques parallèles. Dans l'ordonnancement en temps-réel, l'action et l'intervention sur le système se font dès l'évolution et le changement qui ont lieu.

L'ordonnancement s'effectue en deux étapes : l'ordonnancement statique, correspond à l'ordonnancement des tâches prévisionnelles (connues a priori) et l'ordonnancement dynamique consistant à utiliser le séquencement précédent et inclure les nouvelles tâches, dès leur apparition (en temps-réel), suivant des règles adéquates.

Le problème à considérer est la minimisation de la durée totale d'achèvement, dite aussi Makespan, sous des contraintes liées aux ressources, aux intervalles temporels et aux tâches elles-mêmes ainsi que le nombre de tâches aléatoires ordonnancées. L'ordonnancement des tâches aléatoires nécessite le décalage des tâches programmables dans le programme de leur exécution.

Dans cette étude, la préemption est interdite.

La stratégie, à adopter, est de considérer les deux ordonnancements (statique et dynamique) individuellement. Elle consiste à ordonnancer les tâches prévisionnelles (ordonnancement statique). Utiliser ce dernier comme base d'un nouvel ordonnancement en plaçant les tâches aléatoires dès leur apparition. Cette affectation des tâches aux ressources se fait grâce à la règle d'ordonnancement choisie. Ensuite, dans le but de minimiser la durée totale d'ordonnancement (Makespan), en considérant une seule ressource, réordonnancer les tâches placées précédemment et trouver la bonne séquence à déterminer.

Ce présent mémoire est composé de quatre chapitres (cf. Table des matières).

Le premier chapitre est une présentation de l'ordonnancement et sa place dans l'entreprise ou plus précisément dans un système de production.

Nous y présentons la production et sa gestion, les différentes décisions intervenant dans un système de gestion sans oublier la relation entre la gestion de production et les différentes fonctions de l'entreprise. Les généralités sur l'ordonnancement sont présentées entamées en définissant l'ordonnancement puis son problème. Les critères de performance et la classification des problèmes d'ordonnancement sont introduits dans ce chapitre. Dans la septième section, nous présentons des méthodes de résolution des problèmes d'ordonnancement statique et dynamique. La première sous section présente les méthodes exactes et les méthodes approchées pour résoudre le problème d'ordonnancement statique. La deuxième sous section est dédiée aux méthodes de résolution des problèmes d'ordonnancement dynamique.

Le second chapitre traite le problème d'ordonnancement sur des ressources identiques parallèles en temps-réel. Ce chapitre est entamé en présentant le problème général d'ordonnancement sur des ressources identiques parallèles, puis, les problèmes d'ordonnancement temps-réel. La complexité du problème d'ordonnancement temps-réel sur des ressources identiques en parallèle est introduite au niveau de la quatrième section. La cinquième section est un recueil d'un certain nombre de travaux de recherche dans le domaine de l'ordonnancement statique et dynamique. La sixième section est dédiée à la présentation d'une démarche de résolution du problème qui nous intéresse.

Le troisième chapitre est consacré à l'une des approches de résolution des problèmes d'ordonnancement temps-réel sur des ressources identiques en parallèle. Chaque ressource dispose d'une file d'attente des tâches programmables. Une file d'attente des tâches aléatoires est commune à toutes les ressources. Pour la résolution de ce problème, deux algorithmes d'ordonnancement en temps-réel sont proposés, un algorithme d'ordonnancement au plus tôt et un autre au plus tard.

Le chapitre quatre présente une autre approche du problème d'ordonnancement temps-réel sur des ressources identiques parallèles. Pour chaque ressource, il existe une file d'attente des tâches programmables et une autre pour les tâches aléatoires. Deux situations sont à distinguer : le cas d'affectation des tâches aléatoires aux files d'attente des tâches aléatoires en les positionnant directement dès leur apparition et le cas d'affectation des tâches aléatoires aux files d'attente un certain temps d'attente après leur arrivée.

Ce travail est achevé par une conclusion générale suivie des annexes et des références bibliographiques.

Chapitre I :

L'Ordonnancement

I.1 Introduction

La performance industrielle s'est complexifiée. A côté de la mesure traditionnelle de productivité allant dans le sens de gestions tayloriennes[1], d'autres formes de performance se sont progressivement imposées, en réponse aux évolutions technologiques des systèmes industriels[2]. La performance est dans ce nouveau contexte induite par une compétitivité non plus seulement basée sur les coûts, mais aussi la qualité et surtout les délais. Aussi, son appréhension ne peut plus se restreindre à une minimisation des coûts et une augmentation du volume de production.

De ce fait, l'ordonnancement ne consiste plus uniquement à produire en un temps raisonnable mais doit tenir compte de phases importantes, comme la livraison et l'installation du produit pour satisfaire les besoins d'un client.

Dans ce chapitre, nous nous intéressons à la présentation de l'ordonnancement et sa place dans l'entreprise ou plus précisément dans un système de production.

Nous présentons, dans la deuxième section, la production et la gestion de production ainsi que les différentes décisions intervenant dans un système de gestion. La troisième section est dédiée aux généralités sur l'ordonnancement en l'entamant par la présentation de certains concepts et définitions.

La quatrième section présente les organisations d'ateliers. La cinquième section est dédiée à l'énumération des critères de performance. La classification des problèmes d'ordonnancement est présentée dans la sixième section.

La septième section, divisée en deux sous sections, est consacrée à la présentation des différentes méthodes de résolution des problèmes d'ordonnancement. La première sous section présente les méthodes de résolution (exactes et approchées) des problèmes d'ordonnancement statique. La deuxième sous section est réservée à la présentation des méthodes de résolution des problèmes d'ordonnancement dynamique.

I.2 Production et gestion de production

I.2.1 La production

Il n'y a pas si longtemps, la production pouvait s'étudier indépendamment du reste de l'entreprise [Afn 88]. Le marché, [Ben 98], était en situation de pénurie, il suffisait de disposer d'une marchandise (quelles qu'en soient les caractéristiques) pour être sûr de la

[1] Division du travail, spécification des tâches, minimisation de l'utilisation des ressources.
[2] Intégration, concepts productiques, importance des activités hors production…

vendre. Tel n'est plus le cas de nos jours où, pour vendre un produit, il faut trouver quelqu'un pour qui ce produit correspond à un besoin et qui est prêt à en payer le prix.

La production est caractérisée par sa nature, ses modes et ses types*.

Dans une économie de marché, tout chef d'entreprise souhaitant développer son système de production est amené à agir dans plusieurs directions, aussi bien à moyen terme qu'à long terme. Il recherche à [Hen 99] :

- Minimiser les risques de l'entreprise,
- Optimiser les stocks,
- Diminuer le poids des en-cours,
- Diminuer le coût de revient des produits,
- Diminuer les délais,
- …, etc.

Certains objectifs vont dans le même sens, sans avoir toutefois pour optimum la même solution. D'autres, par contre, sont antagonistes.

I.2.2 La gestion de production

Depuis quelques décennies, pour vendre, il faut d'abord trouver qui a besoin du produit, qui veut l'acheter et à quel prix [Gal 94]. Actuellement, toute entreprise se doit d'être plus performante et d'avoir un niveau supérieur sur le plan technique et économique, par rapport aux entreprises concurrentes.

La gestion de production a pour objet la recherche d'une organisation efficace dans l'espace et dans le temps de toutes les activités relatives à la production afin d'atteindre les objectifs de l'entreprise [Gia 88]. L'amélioration de la gestion de production est à la base de toute recherche dans le domaine de la productivité à accroître [Afn 88]. La productivité est la clé de la survie et du développement des entreprises industrielles.

Selon Bénassy, dans [Ben 98], "La gestion de production, ça n'est pas difficile, c'est seulement compliqué", expliquant que la diversité des entreprises et des positions vis-à-vis du marché, que la complexité des traitements et que la fréquence des aléas de tous types sont les conditions contemporaines qui assurent un niveau de complexité toujours croissant de la gestion de production.

Pour Javel, [Jav 97] et [Dup 97a], la gestion de production consiste à produire en temps voulu, les quantités demandées par les clients dans des conditions de coût de revient et de

* Voir Annexe I.

qualité déterminées en optimisant les ressources de l'entreprise de façon à assurer sa pérennité, son développement et sa compétitivité.

Nous distinguons deux types de gestion de production : la gestion de production sur commande et la gestion de production sur stock [Hen 99] et [Gia 88][3].

Un troisième type de gestion de la production existe, c'est un compromis entre les deux précédents types de gestion, appelé « gestion de production mixte ». On le retrouve souvent dans le cas où les délais de livraison sont inférieurs à la durée du processus de fabrication [Hen 99].

I.2.3 Le système de production

I.2.3.1 Définition

Un système de production est un ensemble de ressources réalisant une activité de production.

La transformation de ressources s'effectue par une succession d'opérations qui utilisent des ressources (machines et opérateurs) et modifient les matières premières ou composants entrant dans le système de production afin de créer les produits finis sortant de ce système et destinés à être consommés par des clients. Les modifications peuvent porter sur la forme du produit, sa structure, son apparence, etc. La transformation subie par les produits leur procure de la valeur ajoutée. Les ressources appartenant au système de production mobilisées pour réaliser l'activité de production peuvent être des machines, des opérateurs, de l'énergie, des informations, des outillages,..., etc.

Une des fonctions importantes du système de production est la fabrication elle-même du produit fini, mais son bon déroulement nécessite la mise en œuvre de fonctions additionnelles telles que les achats des composants et matières premières, la distribution du produit fini, la gestion de la qualité des composants et du produit ou la maintenance des différentes ressources interviennent aussi de manière importante. Classer des objets et des systèmes oblige à une réflexion sur leurs invariants, leurs spécificités et leurs cas d'usage. La classification est ainsi une étape importante de la compréhension d'un système.

I.2.3.2 Décomposition du système de production

Les systèmes de production peuvent être des systèmes très complexes et difficiles à gérer au vu de toutes leurs composantes fonctionnelles (fabrication, achat, distribution,

[3] Voir Annexe II.

maintenance,...). Plusieurs approches ont été envisagées dans le but de mieux comprendre leur fonctionnement et de mieux les appréhender.

Dans la littérature, différentes décompositions du système de production sont proposées.

Mélèse, dans [Mél 85], propose une décomposition en deux sous systèmes : le système physique de production (ressources humaines et matérielles) et le système de gestion destiné à piloter le système physique.

Doumeinghts & al., dans [Dou 83] et [Our 01], proposent une décomposition structurée en fonction de la nature des flux qui la traversent : système physique (flux de matières, humains, énergie,…), système de décision permet le pilotage du système physique (flux de décisions) et le système d'informations représentant l'ensemble des informations en stock ou en circulation dans le système (flux d'informations). La même décomposition est proposée dans [Le M 74] et [Le M 77]. La figure I.1 illustre cette décomposition. Elle est une schématisation générale des interactions existantes entre les différents sous systèmes, production et pilotage.

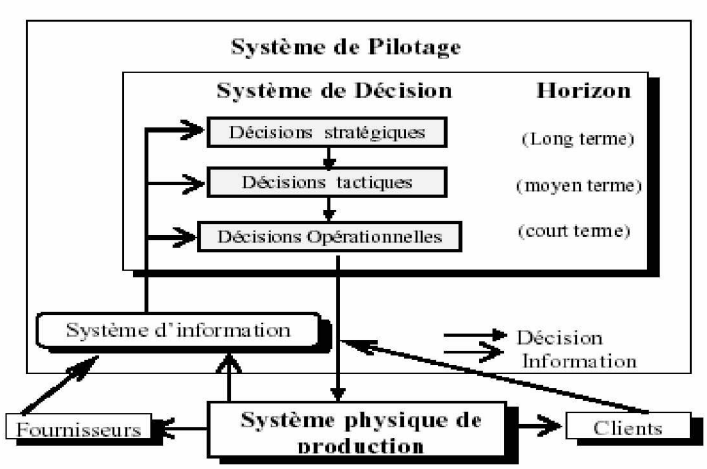


Figure I.1 : Vision systémique d'un système de production [Our 01].

Le système de pilotage est destiné à commander le système physique de production : des données récupérées par le sous système informationnel sont traitées par le sous système décisionnel afin de définir les décisions qui garantissent le bon fonctionnement du système physique compte tenu des multiples contraintes internes (ressources disponibles) ou externes (contraintes des clients et fournisseurs) à l'entreprise. Etant donné que le système de production est dynamique, les commandes sortant du système de pilotage se présentent sous forme d'une suite de commandes appelées contrôles.

Le système de décision contrôle le système physique de production. Il en coordonne et organise les activités en prenant des décisions basées sur les données transmises par le système d'information.

Le rôle du système d'information est de collecter, stocker, traiter et transmettre des informations. Il intervient à l'interface entre les systèmes de décision et de production et à l'intérieur même du système de décision, pour la gestion des informations utilisées lors de prises de décision, et du système physique de production, pour la création et le stockage d'informations de suivi par exemple. L'association des parties des systèmes de décision et d'information concernant uniquement la production, constitue le système de gestion de production.

Le système physique de production transforme les matières premières ou composants en produits finis. Il est constitué de ressources humaines et physiques. Ses activités sont déclenchées et vérifiées par le système de gestion de production.

I.2.3.3 Le pilotage des systèmes de production

Pour mener à bien la fonction pilotage décrite précédemment dans un système de production classique [Sac 02], le gestionnaire doit s'assurer, entre autres, de la bonne exécution des sous fonctions suivantes [Art 97] et [Hen 99] : la planification, la programmation, l'ordonnancement, la conduite et la commande.

I.2.4 Relation de la gestion de production avec les fonctions de l'entreprise

Les systèmes de production sont, par nature, complexes [Bux 89] et [Ben 98]. Il est par ailleurs nécessaire de tenir compte des demandes réelles ou prévues à long et à moyen termes, afin de pouvoir utiliser au mieux les ressources disponibles. Ces facteurs conduisent à une complexité telle qu'il est impossible de résoudre le problème d'ordonnancement globalement, d'autant plus que les événements aléatoires endogènes (pannes de machines, grèves,...) et exogènes (commandes imprévues et prioritaires,...) qui interviennent constamment nécessitent de recalculer fréquemment l'ordonnancement. C'est pourquoi, les chercheurs étudient depuis des dizaines d'années des structures de gestion hiérarchisée [Axs 81] et [Ger 87]. Dans une telle structure, la gestion de production est décomposée en plusieurs niveaux. Chaque niveau a son propre modèle constitué d'entités avec leurs attributs, des critères, des contraintes et des liens entre les entités.

La gestion de production présente des relations avec plusieurs fonctions différentes au sein de l'entreprise.

On trouve généralement la planification, les approvisionnements, les achats, la gestion des ressources humaines et techniques. L'une des méthodes de gestion de production permettant de prendre en compte la complexité des produits (nomenclatures) et des organisations des industries manufacturières est la MRP II (Manufacturing Resource Planning : Planification des Ressources de Production). Elle respecte notamment la forte hiérarchisation des prises de décision dans les entreprises. Les différentes étapes qui composent la MRP II sont présentées sur la figure I.2 :

Au sommet de l'enchaînement présenté sur la figure I.2, le plan industriel et commercial (PIC) intègre mois par mois et par famille de produits les prévisions de vente des produits et les objectifs de production, sur un horizon assez long, de l'ordre d'une année. Les prévisions commerciales du PIC sont ensuite détaillées sur chaque produit et réévaluées sur un horizon plus court (de l'ordre de quelques mois) dans le plan directeur de production (PDP). Celui-ci recense toutes les commandes fermes et prévisionnelles et en déduit un échéancier des quantités à fabriquer par produit et par période. On détermine à ce niveau, si les capacités globales de l'entreprise en hommes et en machines critiques vont être suffisantes ou s'il est nécessaire d'investir dans de nouveaux matériels ou d'embaucher davantage de personnel.

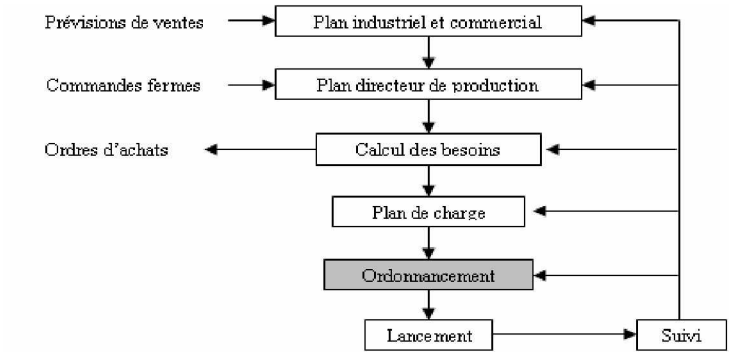


Figure I.2 : Les fonctions de la MRP II [Let 01].

A partir de cet échéancier, le calcul des besoins détermine les quantités de chacun des composants des produits à fabriquer et les dates de fabrication, en se basant sur la nomenclature des produits et les délais de fabrication. A l'issue du calcul des besoins, on dispose des quantités de composants élémentaires à fabriquer (ordres de fabrication) et de matières premières et de composants à acheter (ordres d'achat), accompagnées de leurs dates de besoin. Le jalonnement de ces ordres de fabrication permet de déterminer la charge par machine et par période. On peut donc vérifier si les commandes sont réalisables ou pas (s'il y a suffisamment de ressources pour assurer la production des commandes), et prendre d'éventuelles mesures correctives telles que des heures supplémentaires pour les opérateurs, la

sous-traitance d'une partie du travail ou le lissage de la charge. Lorsque l'adéquation charge / capacité est faite, les opérations sont ordonnancées.

L'ordonnancement correspond à l'affectation des opérations sur les ressources[4]. Cette fonction de gestion de production étant l'objet principal de notre étude, elle est détaillée dans la section suivante. Pour le lancement, les ordres de fabrication sont transmis à l'atelier pour que les pièces soient fabriquées.

La méthode MRP II est adaptée à la gestion des organisations complexes car elle permet de décomposer les décisions du niveau stratégique (long terme, PIC et PDP) au niveau opérationnel (court terme, ordonnancement et lancement), en passant par des décisions tactiques (moyen terme, calcul des besoins et plan de charge). Cette décomposition permet de gérer la complexité des données et des décisions. La MRP II permet en outre de faire face, dans une certaine mesure, à l'incertitude des données gérées. La décomposition temporelle des décisions conduit, à long et moyen terme, à utiliser des données globales, par famille, où l'incertitude peut être prise en compte par l'intermédiaire de marges de sécurité, d'outils statistiques, etc. Toutefois, à court terme et notamment au niveau de l'ordonnancement, alors qu'il n'est plus possible de travailler sur des données globales, l'incertitude qui caractérise individuellement ces données est laissée de côté [Let 01].

I.3 Généralités sur l'ordonnancement

I.3.1 Introduction

Le plan de charge permet de vérifier si la charge occasionnée dans l'atelier par les commandes n'est pas supérieure à la capacité des ressources de l'atelier. Dans le cas contraire, des réajustements de charge (lissage) ou de capacité (sous-traitance, heures ou équipes supplémentaires) peuvent être faits. Les données sont transmises à la fonction ordonnancement.

Bien que globalement l'adéquation charge / capacité soit vérifiée sur un horizon donné, on se retrouve au niveau du lancement avec plusieurs ordres de fabrication (OF) présentant diverses contraintes telles que les dates d'achèvement, les dates de livraison, la non préemption,…, etc. faisant qu'il est impossible de les traiter en même temps. A ce niveau, un séquencement des ordres de fabrication sur les ressources s'impose. Nous parlons, dans ce cas là, d'ordonnancement.

[4] Ressources techniques en général, mais de plus en plus on se soucie de l'affectation des ressources humaines.

I.3.2 Définitions et concepts

L'ordonnancement est la programmation dans le temps de l'exécution d'une série de tâches (ou activités, opérations) sur un ensemble de ressources physiques (humaines et techniques), en cherchant à optimiser certains critères, financiers ou technologiques, et en respectant les contraintes de fabrication et d'organisation [Got 93] et [Esq 99]. Les ordres de fabrication, suggérés par le calcul des besoins, représentent chacun une requête pour fabriquer une quantité déterminée de pièces pour une date donnée. Ils constituent les données d'entrée de l'ordonnancement et permettent de définir, au moyen des gammes de fabrication, l'ensemble des tâches que la fonction ordonnancement doit planifier.

Dans la littérature, nous distinguons un certain nombre de définitions d'un problème d'ordonnancement dont nous citerons quelques unes :

- *Un problème d'ordonnancement consiste à affecter des tâches aux ressources et à décider de leur répartition dans le temps, de manière à optimiser un critère ou à trouver un compromis entre plusieurs critères* [Car 87].
- *Programmation coordonnée d'une ou plusieurs ressources pour réaliser un ensemble de tâches ou d'opérations dans un laps de temps donné* [Gia 88].
- *Ordonnancer un ensemble de tâches, c'est programmer leur exécution en leur allouant les ressources requises et en fixant leurs dates de début* [Got 93].
- *Ordonnancer la production, c'est planifier les dates de lancement et de fin des opérations ainsi que l'affectation de ces opérations aux différents postes de production susceptibles de les exécuter. On classe généralement la fonction d'ordonnancement parmi les activités de planification à très court terme, quoique dans le cas de l'ordonnancement de projets elle puisse parfois tenir compte d'un horizon de plusieurs années.* [Dup 97 b].
- *Le problème d'ordonnancement a pour objet de définir, prévoir et coordonner l'ensemble des ressources physiques et humaines nécessaires à la fabrication. La solution de ce problème devra permettre d'optimiser l'utilisation des ressources tout en respectant les délais de réalisation fixés. Cette solution doit donc répondre aux questions suivantes : Qui ? Quoi ? Quand ? Combien ?* [Jav 97]. *La réponse sera obtenue lorsqu'on saura quelle est la séquence de passage des travaux et les machines sur lesquelles ils doivent passer, tout en optimisant un ou plusieurs critères donnés* [Hen 99].
- *L'ordonnancement consiste, en fonction des prévisions de commandes clients et de disponibilité des ressources, à:*
    1. *déterminer le calendrier prévisionnel de fabrication ;*

2. *distribuer les documents nécessaires à la bonne exécution des fabrications ;*
3. *suivre l'exécution des fabrications.*

Autrement dit, il décrit l'exécution des tâches et l'allocation des ressources au cours du temps et vise à satisfaire un ou plusieurs objectifs [Esq 99].

- Ø *L'ordonnancement est une fonction permettant de déterminer les priorités de passage des produits et leur affectation sur les différentes machines* [Hen 99].
- Ø *Le problème d'ordonnancement consiste à organiser dans le temps la réalisation des tâches, compte tenu des contraintes temporelles (délais, contraintes d'enchaînement, etc.) et de contraintes portant sur l'utilisation et la disponibilité des ressources requises par la tâche* [Lop 01].

I.3.3 Problématique de l'ordonnancement

L'environnement actuel des entreprises est caractérisé par des marchés soumis à une forte concurrence et sur lesquels les exigences et les attentes des clients deviennent de plus en plus fortes en termes de qualité, de coût et de délais de mise à disposition. Cette évolution se renforce par le développement rapide des nouvelles technologies de l'information et de la communication qui permettent une relation directe entre les entreprises (Business to Business) et entre les entreprises et leurs clients (Business to Customer). Dans un tel contexte, la performance de l'entreprise se construit selon deux dimensions [Lop 01] :

- Ø ***Une dimension technologique :*** qui vise à développer les performances intrinsèques des produits mis sur le marché afin de satisfaire aux exigences de qualité et de réduction du coût de possession des produits. L'innovation technologique joue ici un rôle important et peut constituer un élément différenciateur déterminant pour la pénétration et le développement d'un marché. Il faut relever à ce niveau que l'évolution technologique rapide des produits et les exigences de personnalisation de ces produits qu'attendent les marchés conduisent les entreprises à abandonner souvent les productions de masse pour s'orienter vers des productions de petites ou moyennes séries, sinon à la commande. Ceci leur impose donc d'avoir des systèmes de production flexibles et évolutifs, capables de s'adapter aux exigences et aux besoins des marchés rapidement et efficacement.
- Ø ***Une dimension organisationnelle :*** qui vise à développer la performance en terme de durée des cycles de fabrication, respects des dates de livraison prévues, gestion des stocks et des en-cours, adaptation et réactivité face aux variations du carnet commercial,… Cette dimension joue un rôle de plus en plus important, dans la

> mesure où les marchés sont de plus en plus versatiles et évolutifs et exigent des temps de réponse des entreprises de plus en plus courts. Il faut donc disposer de méthodes et d'outils de plus en plus performants pour l'organisation et la conduite de la production.

L'organisation de l'entreprise repose sur plusieurs fonctions[5] où l'ordonnancement en est essentiel. Il n'y a pas si longtemps, l'ordonnancement consistait à fixer les échéances à relativement court terme et affecter les tâches dans le temps et aux moyens de production.

Suivant le niveau auquel s'effectue l'ordonnancement, la difficulté du problème est différente. En effet, au niveau du moyen et du long terme, l'ordonnancement doit être fait à capacité infinie, alors qu'au niveau de l'atelier (court et très court terme) il est fait à capacité finie en tenant compte de nombreuses contraintes telles que le partage de ressources, les délais de livraison,… . En théorie de la complexité, ce deuxième type constitue un problème NP-Difficile [Fox 83]. Il faut également remarquer que le problème de prise de décision associé au problème d'ordonnancement appartient à la classe des problèmes combinatoires [Lop 01]. Aucun algorithme ne peut fournir la solution mathématiquement optimale en un temps raisonnable à de tels types de problèmes malgré la puissance de plus en plus grande des moyens de calculs. Toutefois, les entreprises disposent de moyens limités pour établir de bons ordonnancements. Leurs soucis est d'aboutir à une meilleure solution que celle proposée en un temps relativement court et satisfaisant un certain nombre de critères.

A l'heure actuelle, le contexte dans lequel la fonction ordonnancement s'insère est le siège de changements profonds. La volonté d'un rapprochement des agents influant sur les coûts, les délais et la quantité des produits ou projets, remet en cause la structure et le fonctionnement des organisations fondées sur la division du travail et l'efficacité de la spécialisation.

Ces changements ont deux conséquences principales. D'une part, l'approche tendant à l'automatisation complète des décisions d'ordonnancement est critique, car elle prive les décideurs de l'interactivité nécessaire à l'intégration des facteurs qui ne sont pas pris en compte dans les modèles. D'autre part, la prise de décisions est vue comme résultante d'un processus de décision plus dynamique et plus collectif [Esq 99].

Les contraintes économiques actuelles, l'évolution technologique et la variété des produits imposent à tout système de production la réactivité et le respect des délais. Tout retard dans la production et la livraison est pénalisé.

[5] Planification, ordonnancement, lancement, suivi,…

Une gestion efficace d'un système de production réside dans la meilleure planification et notamment dans le bon ordonnancement des tâches à exécuter. Ce dernier représente l'un des principaux problèmes des systèmes de production et des gérants. Le résoudre est d'un intérêt capital.

La réactivité d'un système de production se traduit par sa réponse rapide et efficace face aux situations imprévues et urgentes, en les intégrant au mieux dans le programme de production. Autrement dit, l'apparition de tâches imprévues au cours de l'exécution d'une commande prévisionnelle nécessite le traitement de ces tâches aléatoires en les incluant dans le programme d'exécution de celles connues a priori.

Cette réactivité est la caractéristique principale de l'ordonnancement en temps-réel. Ce dernier représente la réponse du système dès qu'un changement se produit tout en préservant l'objectif de production préfixé (il répond aux aléas et aux changements en s'adaptant à chaque évolution du système).

I.3.4 Les classes d'ordonnancement

Les approches classiques de gestion de production dissocient l'ordonnancement de projet et l'ordonnancement des fabrications[6] [Gaz 03].

I.3.5 Eléments fondamentaux d'un problème d'ordonnancement d'ateliers

Les éléments fondamentaux d'un problème d'ordonnancement d'ateliers représentent les tâches, les opérations, les ressources et les contraintes[*].

I.3.6 Propriétés des ordonnancements

Des sous-ensembles d'ordonnancement particuliers[7] sont ici cités dont certains présentent des propriétés de dominance vis-à-vis de tout critère[8] régulier (minimisation de la durée, de la somme des retards,…) : Ordonnancement actif, semi-actif, admissible, sans retard et au plus tôt / au plus tard.

[6] Voir Annexe I.
[*] Voir Annexe I.
[7] Voir Annexe I.
[8] On appelle critère toute règle permettant de classer des ordres de fabrication ou des opérations à planifier à une étape précise des différentes méthodes de calcul. Il peut être temporel, potentiel ou structurel.

I.3.7 Objectifs de l'ordonnancement

Le traitement de l'ordonnancement dans la littérature s'est tout d'abord orienté vers une optimisation monocritère. L'environnement manufacturier évoluant rapidement et la concurrence devenant de plus en plus acharnée, les objectifs des entreprises se sont diversifiés et le processus d'ordonnancement est devenu de plus en plus multicritère. Les critères que doit satisfaire un ordonnancement sont variés.

D'une manière générale, on distingue plusieurs classes d'objectifs concernant un ordonnancement [Esq 99] (les objectifs liés au temps, les objectifs liés aux ressources, les objectifs liés au coût, les objectifs liés à une énergie ou un débit).

Les objectifs à satisfaire au niveau de l'ordonnancement sont issus des objectifs globaux de l'entreprise, par décomposition. Cette décomposition conduit à une structure d'objectifs qui permet de gérer les contradictions et les compromis [Far 90] et [Gra 98].

Chaque entreprise a ses propres critères, dépendants de sa politique, de son passé, de ses problèmes particuliers,... . Même si on y retrouve certains points communs, ces objectifs sont trop dépendants de la compagnie, de l'environnement manufacturier ou tout simplement de la personnalité et des habitudes du responsable de la production pour pouvoir être définitivement fixés dans les méthodes de résolution de problèmes d'ordonnancement.

I.4 Les organisations d'ateliers

Dans les problèmes d'ordonnancement d'ateliers, les ressources sont des machines ne pouvant réaliser qu'une tâche (ou opération). D'autre part, chaque travail concerne une entité physique indivisible appelée produit, ou lot lorsque plusieurs produits identiques sont groupés. Une entité ne peut pas se retrouver simultanément en deux lieux distincts, un même travail ne peut être exécuté qu'à raison d'une opération à la fois, sur une des machines.

I.4.1 Les différents types d'organisation

On peut regrouper les organisations en trois grandes classes comme suit[9] :

- Les organisations à ressource unique.
- Les organisations à ressources multiples.
- Les organisations multi lignes parallèles.

[9] Voir Annexe I.

I.4.2 Relations entre organisations

La figure ci-dessous (figure I.3) illustre schématiquement les différences et les ressemblances qui existent entre les différentes organisations de production. Un arc reliant une organisation O_1 à une autre organisation O_2 peut être interprété comme suit : O_2 est un cas particulier de la première organisation O_1 [Gaz 03].

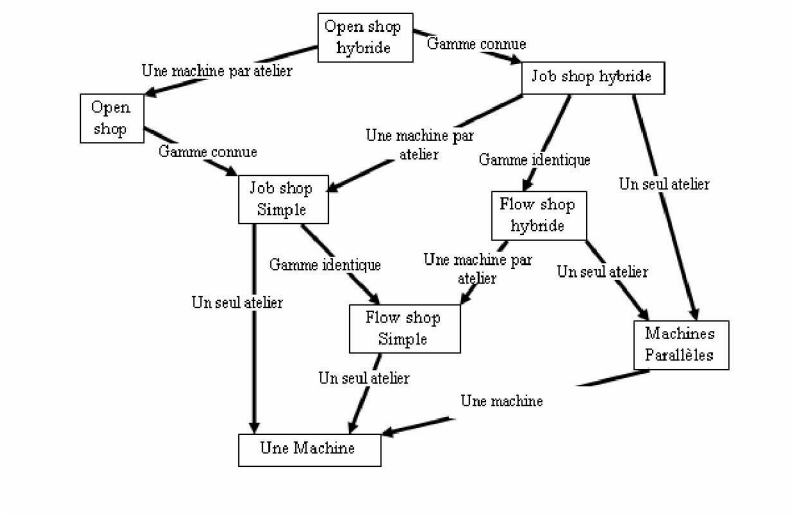


Figure I.3 : Schéma montrant les liens entres les diverses organisations.

I.5 Les critères de performances et leur évaluation

De nombreuses définitions des indicateurs de performance ont été données. La plus couramment admise, de nos jours, est celle donnée par l'Association Française de Gestion Industrielle dans [Afg 92] : « Un indicateur de performance est une donnée quantifiée qui mesure l'efficacité et/ou l'efficience de tout ou partie d'un processus ou d'un système (réel ou simulé), par rapport à une norme, un plan ou un objectif, déterminé et accepté dans le cadre d'une stratégie d'entreprise. »

I.5.1 Définitions

Nous définissons, dans cette sous section, les variables intervenant le plus souvent dans l'expression de la fonction objectif :

C_i : la date d'achèvement ou de terminaison de la tâche i (Completion time) ;

L_i : le retard algébrique de la tâche i (Lateness), $L_i = C_i - d_i$, (d_i : due date) ;

T_i : le plus grand retard absolu de la tâche i (Tardiness), $T_i = \max(0, L_i)$;

U_i : l'indicateur de retard, $U_i = 0$ si $L_i \leq 0$ et 1 sinon ;

F_i : le temps de présence ou de séjours (Flow time) de la tâche i dans l'atelier, $F_i = C_i - r_i$, (r_i : release date) ;

W_i : le temps d'attente de la tâche i (Waiting time), $W_i = C_i - r_i - \sum p_{ij}$;

E_i : l'avance absolue (Earlyness), $E_i = -\min(0, L_i)$.

I.5.2 Classification des critères de performance

Nous pouvons regrouper les critères de performances suivant les quatre points ci-dessous[10] :

- Les dates de fin d'exécution notées C_i.
- Les dates de livraison notées d_i.
- Les volumes des en-cours ($\overline{C}$).
- L'utilisation efficace des ressources (C_{max}).

I.5.3 Propriétés des critères

Un critère est dit *régulier* quand sa valeur décroît si et seulement si l'un au moins des C_i décroît (les autres restent fixes) [Our 01] et [Gaz 03].

Un critère en fonction des variables C_i est dit *régulier* si l'on peut le dégrader en avançant l'exécution d'une tâche. C'est pour la minimisation du temps total d'exécution, du plus grand retard, etc. [Esq 99].

Un sous-ensemble de solutions est dit dominant pour l'optimisation d'un critère donné, s'il contient au moins un optimum pour ce critère [Esq 99].

I.5.4 Relation entre les critères

Nous nous intéressons dans cette sous section aux diverses relations d'équivalence et de similitude qui peuvent exister entre les critères d'optimisation. En effet, une solution optimale pour l'un ou l'autre des problèmes pour un critère donné peut s'avérer optimale pour un autre critère. La figure I.4 donne les liens entre certains critères, nous lisons au début, qu'une solution optimale pour le critère L_{max} est aussi optimale pour le critère C_{max} (avec $d_i = 0$). Autrement dit, une méthode qui optimise L_{max} optimise C_{max}, la méthode qui optimise le critère $\sum W_i C_i$ optimise aussi le critère $\sum C_i$, etc. [Pin 95] et [Esq 99].

[10] Voir Annexe I.

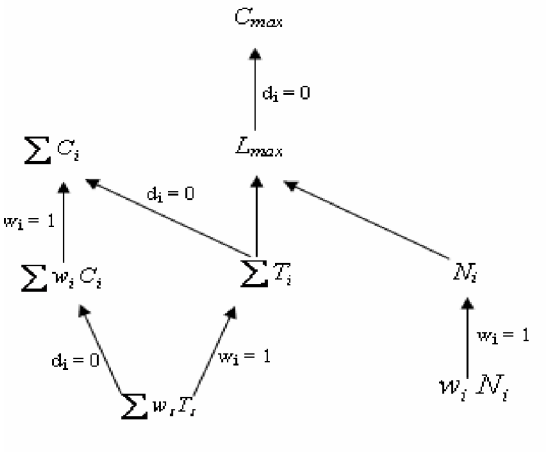


Figure I.4: Les relations entre les critères d'optimisation.

I.6 Classification des problèmes d'ordonnancement

Plusieurs classifications ont été proposées depuis la définition du premier problème d'ordonnancement. Certains auteurs utilisent parfois d'autres classifications bien spécifiques. Plus les problèmes se diversifient plus nous aurons besoin d'actualiser nos notations tout en gardant un souci d'homogénéité des classifications existantes. Les classifications[11] les plus connues sont :

- Classification de Conway, Maxwell et Miller (1967) [Con 67]
- Classification de Graham & al. (1979)
- Classification de Lawler & al. étendue par Vignier & al. (1995)[Vig 95]

La version originale de la classification de Conway, Maxwell et Miller (A/B/C/D) ne peut être utilisée pour caractériser les problèmes hybrides ou d'ateliers avec plus d'une ressource. Elle a été étendue par Lenstra en 1976 en rajoutant à la notation originale un cinquième paramètre en éclatant le troisième (C' et C'') pour inclure les contraintes.

L'utilisation des indices, dans la classification de Lawler & al. étendue par Vignier & al., ne facilite pas la tâche tant pour l'écriture que pour la lecture. Plus le nombre d'ateliers augmente, plus l'écriture sera fastidieuse, difficile à lire et prendra plus de place.

Dans ce travail, nous nous intéressons à la classification de Graham & al.

[11] Voir Annexe I.

I.7 Les approches de résolution

Nous traitons des approches pour les ordonnancements statiques et déterministes. Ces approches utilisent des algorithmes pour définir le meilleur ordonnancement sans tenir compte des perturbations qui peuvent intervenir. Par contre, les approches d'ordonnancement temps-réel régénèrent de nouveaux ordonnancements pour faire face aux aléas (arrivée aléatoire des tâches, pannes de machines,…) [Our 01].

Les problèmes d'ordonnancement font partie intégrante de la famille des problèmes d'optimisation combinatoire [Per 03]. L'optimisation combinatoire occupe une place très importante en recherche opérationnelle, en mathématiques discrètes et en informatique. Son importance se justifie d'une part par la grande difficulté des problèmes d'optimisation [Pap 82] et d'autre part par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire [Rib 94]. Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont généralement difficiles à résoudre. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-Difficiles et ne possèdent donc pas à ce jour de solution algorithmique efficace valable pour toutes les données [Gar 79]. Etant donnée l'importance de ces problèmes, de nombreuses méthodes de résolution ont été développées en recherche opérationnelle (RO) et en intelligence artificielles (IA). Ces méthodes peuvent être classées sommairement en deux grandes catégories : les méthodes exactes (complètes[12]) qui garantissent la complétude de la résolution et les méthodes approchées (incomplètes) qui perdent la complétude de la résolution pour gagner en efficacité [Gaz 03].

I.7.1 Ordonnancement statique

Un problème d'ordonnancement prévisionnel est un problème qui consiste à optimiser une fonction objectif sur un ensemble de configurations, le problème ainsi défini est un problème d'optimisation combinatoire. Pour résoudre de tels problèmes, un algorithme simple consiste à essayer toutes les configurations possibles pour choisir celle qui optimise la fonction objectif. Néanmoins, cet algorithme ne peut être utilisé dans la majorité des cas du fait que le nombre de configurations devient très important avec l'augmentation de la taille du problème. Par conséquent, le temps de calcul devient aussi très important (par exemple, pour

[12] Une méthode est complète si quel que soit le problème abordé, on peut prouver en un temps raisonnable que le problème admet ou non des solutions.

le cas d'un problème d'ordonnancement à $n = 5$ tâches et $m = 5$ machines, le nombre de configurations est $(n !)^m = 2.5\ 10^{10}$ [Our 01].

Les problèmes d'optimisation combinatoire qui ne peuvent être résolus par des algorithmes polynomiaux sont dits NP-Difficiles. La mise en œuvre des méthodes heuristiques (algorithmes approchés) permet d'obtenir une solution satisfaisante dans un temps de calcul raisonnable.

Cette sous section est réservée pour la présentation de ces différentes méthodes.

I.7.1.1 Les méthodes exactes

Le principe essentiel d'une méthode exacte consiste généralement à énumérer, souvent de manière implicite, l'ensemble des solutions de l'espace de recherche. Pour améliorer l'énumération des solutions, une telle méthode dispose de techniques pour détecter le plus tôt possible les échecs (calculs de bornes) et d'heuristiques spécifiques pour orienter les différents choix. Parmi les méthodes exactes, nous trouvons entre autres : des procédure par séparation et évaluation et la résolution de modèles mathématiques en nombres entiers. Le temps de calcul de ces méthodes est exponentiel ce qui explique qu'elles ne sont utilisables que sur des problèmes de petites tailles.

I.7.1.1.1 Les méthodes de potentiels

Ces méthodes sont utilisées pour résoudre les problèmes d'ordonnancement de n tâches liées par des contraintes de succession et de localisation temporelle, avec comme objectif la minimisation du délai des tâches (la date de fin de toutes les tâches, la plus grande date de fin C_{max} (makespan)) [Sac 02].

Ces méthodes supposent que la capacité des ressources est illimitée. Elles sont fondées sur les graphes valués et sont connues sous les appellations : méthodes de potentiels-tâches de Bernard Roy et la méthode potentiel/étape (CPM/ PERT).

Les algorithmes de résolution correspondants sont des algorithmes polynomiaux issus de la programmation dynamique.

Ces méthodes sont adaptées surtout pour la gestion de projet et sont très peu extensibles aux problèmes manufacturiers.

I.7.1.1.2 Les procédures par séparation et évaluation (Branch & Bound)

Les procédures d'optimisation par séparation et évaluation (PSE ou B&B) sont des procédures d'exploration par énumération implicite de l'ensemble des solutions. Ce sont des méthodes exactes particulièrement bien adaptées pour résoudre un problème d'ordonnancement.

La méthode de séparation et évaluation est basée sur une énumération implicite et intelligente de l'ensemble des solutions réalisables. Pour cela, la séparation consiste à décomposer le problème initial en plusieurs sous problèmes qui sont à leurs tours décomposables. Ce processus peut se visualiser sous forme d'un arbre d'énumération. Pour chaque sous problème (nœud de l'arbre), la procédure d'évaluation calcule (dans le cas d'un problème de minimisation) une borne inférieure de la solution obtenue à partir de ce sous problème [Hao 99] et [Lop 01].

Les procédures par séparation et évaluation « PSE » reposent sur quatre composantes essentielles [Lop 01] :

- La technique de séparation.
- La méthode d'évaluation.
- La méthode de sondage.
- La méthode de sélection ou stratégie d'exploration.

Deux grandes stratégies existent. Dans une stratégie meilleure d'abord, on sélectionne puis on sépare le sous problème « pendant » ayant la meilleure évaluation, la méthode arborescente ainsi développée est appelée PSEP (procédure par séparation et évaluation progressive). Dans une stratégie de type profondeur d'abord puis retour arrière, les sommets pendants sont gérés selon une pile : à chaque étape on dépile un problème pendant, on l'évalue, on le sonde et on empile les sous problèmes issus de sa séparation, la procédure résultante est une PSES (procédure par séparation et évaluation séquentielle).

Pour des problèmes de taille importante, il faut craindre d'avoir à manipuler des arborescences conséquentes. Pour limiter la taille de l'arborescence, on peut chercher à raffiner les calculs des bornes inférieures. On peut aussi appliquer des résultats de dominance [Car 88]. En plus du calcul d'une borne inférieure, on peut enfin calculer, pour chaque sommet, une borne supérieure du critère, à l'aide d'une heuristique. Le but est d'interrompre l'exploration d'un sommet lorsque sa borne est plus grande que la meilleure borne supérieure déjà trouvée jusque là.

I.7.1.1.3 La programmation dynamique

Introduite par Bellman dans les années 50 [Bel 57], la programmation dynamique décompose un problème de dimension n en n problèmes de dimension 1. Le système est alors constitué de n étapes que l'on résout séquentiellement, le passage d'une étape à une autre se faisant à partir des lois d'évolution du système et d'une décision [Che 77]. Le principe d'optimalité de Bellman est basé sur l'existence d'une équation récursive permettant de décrire la valeur optimale du critère à une étape en fonction de sa valeur à l'étape précédente. Ainsi, pour appliquer la programmation dynamique à un problème combinatoire, le calcul du critère pour un sous-ensemble de taille k nécessite la connaissance de ce critère pour chaque sous-ensemble de taille k-1 et porte le nombre de sous-ensembles considérés à 2 n (où n est le nombre d'éléments considérés dans le problème). La programmation dynamique est ainsi de complexité exponentielle. Pourtant pour les problèmes NP-Difficiles au sens faible, pour lesquels il existe des algorithmes de résolution pseudo-polynomiaux, il est souvent possible de construire un algorithme de programmation dynamique pseudo-polynomial, pouvant alors être utilisé pour des problèmes de tailles raisonnables.

I.7.1.1.4 La programmation linéaire

Un programme linéaire modélise un problème d'optimisation dans lequel un critère et des contraintes sont des fonctions linéaires des variables. Pour traiter un programme linéaire en variables continues, les deux types d'algorithmes les plus importants sont la méthode du simplexe et la méthode des points intérieurs. En pratique, la méthode du simplexe est performante, bien que de complexité théorique exponentielle. Le meilleur exemple d'une méthode de points intérieurs est l'algorithme de Karmarkar qui résout les programmes linéaires en un temps polynomial [Lop 01].

I.7.1.2 Les méthodes heuristiques ou méthodes approchées

Les méthodes approchées constituent une alternative très intéressante pour traiter les problèmes d'optimisation de grande taille si l'optimalité n'est pas primordiale. En effet, ces méthodes sont utilisées depuis longtemps par de nombreux praticiens [Hao 99].

Une méthode approchée est une méthode qui ne garantit pas l'obtention d'une solution optimale mais qui s'exécute, en général, plus rapidement qu'une méthode exacte.

Parmi ces méthodes, nous distinguons les heuristiques dédiées à un type de problèmes et les métaheuristiques.

Une heuristique est une méthode, conçue pour un problème d'optimisation donné, qui produit une solution non nécessairement optimale lorsque nous lui fournissons une instance de ce problème. Par contre, une métaheuristique est définie de manière similaire, mais à un niveau d'abstraction plus élevé. Le terme « métaheuristique » a été initialement utilisé par F. Glover pour distinguer la méthode Tabou des heuristiques spécifiques [Glo 86], [Glo 89] et [Glo 90].

Les performances d'une méthode approchée sont liées à la qualité de la solution produite ainsi qu'au temps de calcul nécessaire pour l'obtenir. Ces deux critères étant de nature opposée, il s'agit de trouver un compromis, au cas par cas, sur la base des spécificités du problème d'optimisation considéré.

Les cinq classes principales d'heuristiques sont, les approches constructives, les approches de recherche locale, les approches par décomposition, celles dites par relaxation et changement de modèle, et celles considérées comme évolutives. Ces méthodes étant suffisamment générales pour être appliquées à plusieurs catégories de problèmes d'optimisation combinatoire.

I.7.1.2.1 Les algorithmes de liste

Ces algorithmes sont dédiés aux problèmes d'ordonnancement et construisent une solution relativement à une liste de priorité donnée.

Ces algorithmes consistent, dans une première phase, à définir une liste qui donnera l'ordre de prise en compte des éléments. Cette liste construite a priori, à partir d'un critère bien défini, n'est pas remise en cause au cours de l'ordonnancement. La deuxième phase de l'algorithme se réduit à considérer les éléments (processeurs ou tâches) dans l'ordre de la liste pour construire l'ordonnancement.

Une revue détaillée des règles classiques couramment utilisées en ordonnancement est proposée par Panwalkar et Iskander [Pan 77].

La liste des critères utilisés pour le choix des tâches à planifier en priorité est très longue. Nous allons ainsi présenter une liste qui regroupe un ensemble significatif et majoritairement utilisé dans les heuristiques. Ces règles sont, dans la plupart des cas, utilisées afin de construire des solutions initiales.

Ø EDD (Earliest Due Date): ce critère cherche à démarrer les tâches dont la fin au plus tard (ou date échue) est la plus proche afin de maximiser nos chances de ne pas augmenter la durée de l'ordonnancement.

Ø SPT (Shortest Processing Time) ou SIO (Shortest Imminent Operation): elle cherche à planifier le plus tôt possible les tâches dont la durée opératoire est la plus petite. Nous pouvons imaginer facilement qu'une tâche à durée courte permettant de libérer les ressources plus rapidement et ainsi d'autoriser leur utilisation par un autre ensemble de tâches plus rapidement.

Ø LPT (Longest Processing Time): pour des instances de problèmes particuliers, il est possible qu'il soit plus intéressant de retarder les tâches les plus courtes. En effet, ces dernières ont plus de chance de disposer de marges plus importantes vis-à-vis des tâches de longue durée.

Ø GRU (Greatest Resource Utilization) : nous sélectionnons l'ensemble des tâches qui utilisent le maximum de ressources.

Ø GRD (Greatest Resource Demand): il est en fait une combinaison des critères GRU et LPT dans le sens où nous lancerons les tâches que nous qualifierons de « goulet » vis-à-vis, à la fois, de leurs utilisations en ressources et de leurs durées.

Ø LST (Latest Start Time): ce critère est similaire au critère EDD. La seule différence est qu'il s'appuie sur les dates de début au lieu des dates de fin.

Ø MTS (Most Total Successors): l'idée est que si nous planifions une tâche possédant plusieurs successeurs alors nous retardons moins le projet étant donné que nous nous offrons la possibilité de planifier le plus tôt possible le plus de tâches possibles.

I.7.1.2.2 Les approches constructives

Les méthodes constructives produisent des solutions admissibles de la forme $s=(x_1, x_2,...x_n)$ en partant d'une solution initiale vide *s*[0] et en insérant, à chaque étape k $(k=1,2,...,n)$, une composante $x_{o(k)}$ $(o(k)\in\{1,2,...,n\}/\{o(1),o(2),...,o(k-1)\})$ dans la solution partielle courante s[k-1]. La décision qui est prise en étape donnée porte l'indice de la composante à insérer puis sur la valeur à donner à cette dernière. Cette décision n'est jamais remise en question par la suite. La représentation vectorielle $s=(x_1, x_2,...x_n)$ convient très bien pour les solutions d'un problème d'affectation général. Les positions du vecteur correspondent aux objets alors que chaque composante x_i définit la ressource qui est affectée à l'objet « i » [Lop 01].

I.7.1.2.3 Les approches de recherche locale

Ces méthodes également appelées méthodes de recherche de voisinage, sont des algorithmes itératifs explorant l'espace X en se déplaçant pas à pas d'une solution à une autre.

Une méthode de ce type commence à partir d'une solution $s_0 \in X$ choisie arbitrairement ou alors obtenue par le biais d'une méthode constructive.

Le passage d'une solution admissible à une autre se fait sur la base d'un ensemble de modifications élémentaires Δ qu'il s'agit de définir suivant les cas. Une solution « s' » obtenue à partir de « s » en effectuant une modification (ou aussi mouvement) $\delta \in \Delta$ est notée $(s' = s \oplus \delta)$.

Le voisinage N(s) d'une solution $s \in X$ est défini comme l'ensemble des solutions admissibles atteignables depuis s en effectuant des modifications élémentaires δ. Plus formellement, nous pouvons écrire : $N(s) = \{s' \in X / \exists \delta \in \Delta : s' = s \oplus \delta\}$.

Un tel processus d'exploration est interrompu lorsqu'un ou plusieurs critères d'arrêt sont satisfaits.

Parmi ces approches, nous citons la méthode de descente ([Lop 01] et [Gaz 03]), le recuit simulé x ([Met 53], [Kir 83], [Haj 88], [Chu 96] et [Hao 99]) et la recherche tabou ([Glo 86] [Glo 89] et [Glo 90] et [Han 86]).

I.7.1.2.4 Les approches par décomposition

D'une manière générale, il existe plusieurs stratégies de décomposition des problèmes d'ordonnancement. Ces décompositions peuvent être les suivantes [Hen 99][13] :

- Décomposition hiérarchique,
- Décomposition structurelle,
- Décomposition spatiale,
- Décomposition temporelle,
- Décomposition de l'ensemble des solutions.

Portmann, en 1987, présentait des méthodes de décomposition spatiale, temporelle et spatiale & temporelle à la fois pour résoudre des problèmes d'ordonnancement. Plusieurs expérimentations ont été effectuées. Aucune méthode de décomposition présentée ne domine nettement les autres. Des différences de temps de calcul sont à noter. Les résultats ont montré

[13] Voir le glossaire.

l'intérêt de ces approches et leur efficacité par rapport au critère étudié : la somme des retards [Sac 02].

I.7.1.2.5 Les approches par relaxation et changement de modèle

Les approches par relaxation et changement de modèle sont des méthodes où nous modifions le modèle que nous avons à résoudre. Le relâchement de contraintes est la stratégie la plus utilisée dans ces approches. Ce fait conduit à des solutions qui sont souvent non réalisables.

Les méthodes de relaxation sont plus utilisées pour déterminer une bonne borne inférieure (pour augmenter l'efficacité des méthodes de séparation et d'évaluation progressive) que pour calculer une solution du problème d'ordonnancement. La relaxation lagrangienne est la plus connue des méthodes de relaxation [Sac 02].

Dans la relaxation lagrangienne, la contrainte est souvent traduite dans la fonction objectif par un coefficient de pénalité. Le principe de la relaxation est très simple : il consiste à considérer le problème sans certaines de ses contraintes de façon à l'alléger pour le résoudre plus facilement. Parfois, le changement de modèle peut guider vers d'autres formes d'approches de résolution des problèmes d'ordonnancement. En effet, le rapprochement des problèmes d'ordonnancement à d'autres problèmes de la combinatoire peut s'avérer fructueux.

I.7.1.2.6 Les approches évolutives

Ce sont des méthodes qui utilisent des représentations de connaissances et des résolutions liées à l'intelligence artificielle, nous pouvons citer les systèmes experts, les réseaux de neurones, les algorithmes génétiques et la propagation par contrainte, les systèmes intelligents à base de connaissances, les colonies de fourmis (Ants systems), … [Gaz 03].

La plupart de ces méthodes s'inspirent de phénomènes concrets et naturels. Les systèmes experts prennent leur similitude de la réflexion humaine. Les algorithmes génétiques sont des méthodes de plus en plus présentes dans le domaine d'ordonnancement s'inspirant de la génétique [Chu 96], [Mit 96], [Por 96] et [Hen 99]. Le principe des réseaux de neurones est tiré du principe des neurones des êtres vivants, etc.

I.7.2 Ordonnancement dynamique

Un système d'ordonnancement temps-réel est un système qui inclut une méthode qui permet à chaque fois qu'une décision d'affectation doit être prise de proposer une solution en temps-réel tout en tenant compte de l'état présent du système et des objectifs de production. Nous distinguons plusieurs approches d'ordonnancement réactif :

I.7.2.1 La gestion des files d'attente

Afin de fixer les décisions à prendre pour la construction de l'ordonnancement temps-réel, cette approche utilise des règles de priorité. Ces règles permettent, dès la libération de la ressource de sélectionner dans la file d'attente la prochaine tâche à traiter. Nous notons que ce choix dépend des paramètres qui caractérisent les tâches (durées opératoires, dates échues,…) [Our 01]. Aussi, ces règles peuvent être [Bla 82] :

- Locales, si elles ne considèrent que les informations des opérations de la file d'attente considérée,
- Globales, si elles se basent sur une connaissance complète du système de production,
- Statiques, si l'indice de priorité associé à une opération ne varie pas avec le temps,
- Dynamiques, si l'indice de priorité varie avec le temps.

Les algorithmes d'ordonnancements à priorités les plus courants sont les suivants :

- **EDF** (*Earliest Deadline First*) : une tâche est d'autant plus prioritaire que sa date d'échéance absolue est proche de la date courante. Le cas particulier où les tâches sont synchrones est parfois appelé **EDD** (*Earliest Due Date*), ou algorithme de Jackson.
- **LLF** (*Least Laxity First*) : les tâches sont d'autant plus prioritaires que leur laxité est faible à la date courante.
- **RM** (*Rate Monotonic*) : une tâche est d'autant plus prioritaire que sa période est petite.
- **DM** (*Deadline Monotonic*) : une tâche est d'autant plus prioritaire que son échéance relative est petite.

DM équivaut à RM quand l'échéance relative est égale à la période.

Pour l'ordonnancement à priorité dynamique, les démonstrations utilisent le fait que si nous disposons d'un ordonnancement faisable, alors, intervertir deux tâches pour respecter

l'ordre d'ordonnancement EDF ou LLF conserve la faisabilité de l'ordonnancement. Le même type de démonstration est utilisé avec les priorités des tâches pour démontrer l'optimalité de RM/DM.

I.7.2.2 Ensembles d'ordonnancements admissibles

Partant du constat que le problème d'ordonnancement est par essence dynamique, le laboratoire LAAS a développé depuis 1976 une approche d'ordonnancement temps-réel basée sur la méthode d'analyse sous contraintes et dont est issu le logiciel ORABAID. Le problème consiste à caractériser l'ensemble des solutions admissibles (compatibles avec les contraintes du modèle), qui permet en temps-réel d'expliciter des degrés de liberté qui sont exploitées pour réagir aux diverses situations imprévues ([Rou 98a] et [Rou 98b]) [Our 01].

I.7.2.3 Simulation

Le pilotage des systèmes de production consiste à résoudre le problème de l'affectation des tâches à réaliser aux différentes ressources, en minimisant une fonction de coût sujette à un jeu de contraintes. Le choix d'un outil de pilotage (algorithmes ou heuristiques permettant de résoudre de manière exacte sinon acceptable les problèmes de planification et d'ordonnancement) adapté à un cas industriel spécifique est très souvent difficile. En effet, la grande variété des caractéristiques propres au système physique étudié rend complexe la résolution de problèmes d'ordonnancement par une démarche analytique. Cette approche donne lieu à des outils intéressants pour la résolution de cas simples mais, face à des problèmes de taille industrielle, il devient important de développer des méthodes plus subtiles.

Il apparaît, en effet, que les outils développés ne peuvent prendre en compte la totalité des caractéristiques du problème considéré telles que certaines contraintes de fabrication ou certains aléas.

L'utilisation de techniques de simulation devient dès lors pertinente afin de tester le comportement des heuristiques de résolution sur un modèle dynamique de l'entreprise. La simulation est l'un des plus puissants outils d'analyse des systèmes complexes. Aujourd'hui, elle est devenue indispensable pour résoudre les problèmes d'optimisation des flux physiques ou des flux d'informations dans les systèmes de production manufacturiers [Dra 98].

La simulation permet de répondre à la question : Qu'obtiendra-t-on si l'on fait ceci ?mais ne permet pas de répondre à la question : Que faut-il faire pour obtenir cela ? [Dra 98].

Bakalem ([Bak 96]), dans [Our 01], présente la simulation en tant qu'une approche expérimentale s'adaptant particulièrement aux systèmes complexes du fait que les problèmes posés ne sont pas aisément résolus par les modèles analytiques. Considérée comme un outil efficace d'aide à la conception, la simulation modélise un système afin d'analyser son comportement dynamique sur des exemples réels ou sur des exemples générés aléatoirement, et apporte une connaissance sur leurs performances. C'est une méthode qui ne cherche pas à optimiser le système étudié mais plutôt à atteindre des objectifs préétablis

La simulation a beaucoup d'objectifs, tels que l'évolution des performances d'un système donné est au niveau de la structure physique, soit au niveau de la structure, soit au niveau du système de décision.

I.7.2.3.1 Etapes et critères dans la réalisation d'une simulation

L'intégration et l'automatisation de plus en plus poussée des systèmes de production a rendu leur phase de conception critique. Par ailleurs, les volumes des investissements font que les marges d'erreur sont très faibles dans la mise en place de ces systèmes. Dans ce contexte, la simulation informatique révèle toute son utilité pour évaluer les performances prévisionnelles de ces ateliers. La simulation consiste à construire un modèle d'un système réel et à conduire des expériences sur ce modèle afin de comprendre le comportement de ce système et d'en améliorer les performances. Elle se focalise sur une formalisation et une recherche de solutions en utilisant des méthodes par tâtonnement [Dav 94]. En effet, le processus de simulation est un processus itératif basé encore sur l'expérience individuelle. C'est hors de la partie proprement informatique qu'une étude de simulation requiert le plus de temps et de savoir-faire.

De manière relativement simplifiée, le processus de simulation peut être présenté selon la figure I.5. La modélisation fournit un modèle conceptuel (modèle de connaissance), la programmation fournit un modèle exécutable (modèle d'action), la simulation ou l'expérimentation fournit les résultats obtenus du modèle, l'analyse des résultats permet d'évaluer le processus à modéliser [Hab 01].

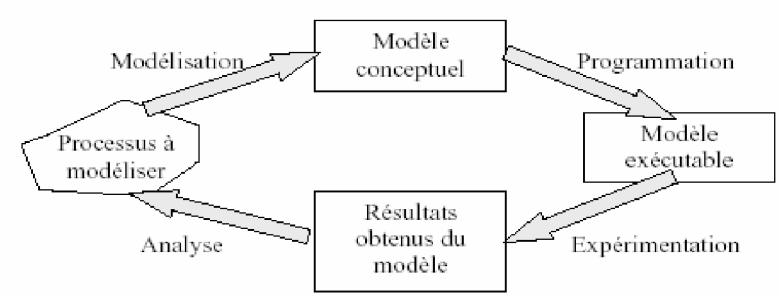


Figure I.5: Processus simplifié de simulation.

C'est hors de la partie proprement informatique qu'une étude de simulation requiert le plus de temps et de savoir-faire. De manière plus détaillée, le processus de simulation peut être éclaté en dix étapes [Pri 86] (figure I.6).

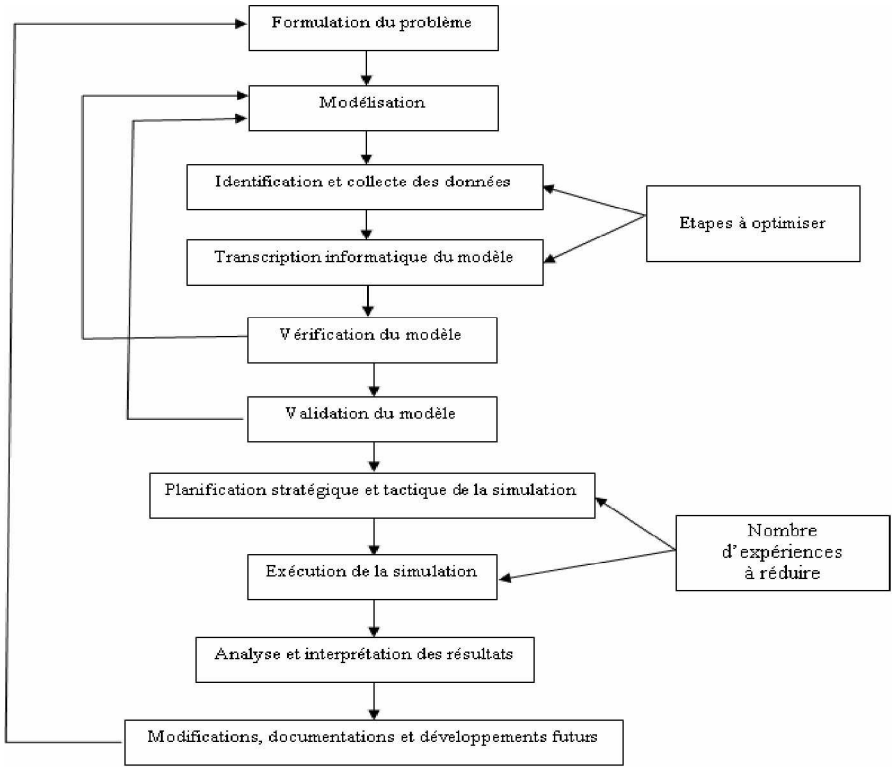


Figure I.6: Processus de simulation selon Pritsker [Hab 01].

I.7.2.3.1.1 Analyse

La première phase, l'analyse du problème, cherche à construire un modèle qui soit le plus simple possible, tout en étant cohérent avec les objectifs de l'étude. C'est une phase difficile, où il faut trouver un compromis entre le prescripteur de l'étude, qui souhaite que l'ensemble des constituants du système soient finement représentés, et le concepteur du

modèle qui cherche toujours la simplification afin de garder la maîtrise de ce dernier. Un principe important, lors de la modélisation en vue d'une simulation est le *principe de parcimonie* [Cas 04] : « *Il n'est pas nécessaire de faire appel à un modèle de granularité élevée pour résoudre un problème s'il s'avère qu'un modèle doté d'une granularité inférieure est capable de lui trouver une solution.* »

La modélisation systémique doit aider à comprendre les objectifs de la simulation et permettre le dialogue entre le demandeur de l'étude et le spécialiste de la simulation. Cette phase comprend :

- § L'identification du problème.
- § La définition des objectifs (cahier des charges).
- § La collecte des données / analyse des données.
- § La rédaction d'un dossier d'analyse fonctionnelle.
- § La validation.

I.7.2.3.1.2 Modélisation

La construction du modèle, comprend la modélisation logico-mathématique qui peut être facilitée par un outil graphique et la programmation proprement dite. Cette phase doit conduire à un modèle facilement modifiable. Cette phase comprend :

- § La modélisation logico-mathématique :
  - le choix des entités, des attributs.
  - la modélisation de l'évolution du système.
- § L'identification des paramètres.
- § La modélisation initiale.
- § Le programme de simulation.
- § La validation et les tests.

I.7.2.3.1.3 Simulation (exploitation du modèle)

Nous avons alors un modèle qui fonctionne. La phase de validation qui intervient alors doit permettre de s'assurer de la cohérence entre le comportement du modèle de simulation et le système réel qu'il est censé représenter. Si ce système existe déjà, cette phase pourra se faire en comparant une trace du comportement du système réel avec celui du modèle de simulation. Lorsque, et c'est souvent le cas, le système étudié n'existe pas encore, cette validation est beaucoup plus délicate. Cette phase comprend :

- § L'expérimentation (jeux de simulation).
- § L'analyse des résultats.
- § L'interprétation des résultats.

Avant de commencer la simulation d'un système, nous devons fixer les critères pour pouvoir obtenir de bons résultats. Quelques critères d'un bon modèle de simulation sont :

- § La connaissance du but de la simulation.
- § La compréhension parfaite du système.
- § Le modèle évolutif (démarrer simple ==> complexe).
- § Le modèle complet sur les caractéristiques principales (décrit les phénomènes principaux avec exactitude).
- § Le modèle flexible, facile à modifier et à mettre à jour.
- § Le modèle solide, qui reste valide face à un grand nombre de situations.
- § Le modèle facile à mettre au point.
- § Le modèle qui donne des résultats de façon claire.

I.7.2.3.2 Concepts de simulation des systèmes de production [Dra 98]

La simulation se divise en deux modes très distincts : celui des simulations discrètes et celui des simulations continues. Les termes de discret et continu s'appliquent en fait à la nature des flux examinés et, par extension, à la nature des modèles. Dans les simulations discrètes, les flux essentiels que l'on examine sont composés d'éléments isolables que l'on peut dénombrer et identifier individuellement. Ces éléments sont couramment appelés « entités ».

Dans les simulations continues, il n'est pas possible d'isoler quoi que ce soit dans le flux étudié, parce qu'il est continu. Le regard que l'on porte sur les systèmes peut être soit statique, soit dynamique. Le regard statique prend pour base un instantané du système ou bien une approximation. En revanche, l'approche dynamique, parce qu'elle tient compte de l'évolution dans le temps des paramètres et des états va rendre avec plus de justesse les phénomènes réels.

Certains outils de résolution du problème ne permettent qu'une approche déterministe, où les mêmes causes produisent immanquablement les mêmes effets, avec une exactitude arithmétique parfaite, infiniment reproductible. La simulation se démarque en capacité à prendre en considération l'aspect stochastique des phénomènes aléatoires.

Il existe plusieurs méthodes d'analyse des systèmes de production (figure I.7) :

§ Les méthodes analytiques (réseaux de files d'attente, réseaux de Petri, chaînes de Markov).

§ La simulation (SIMAN/ARENA, Sirphyco, Cadence, QNAP, Automod).

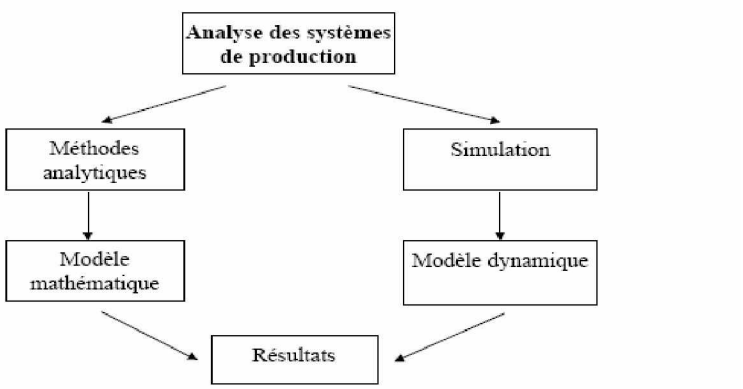


Figure I.7: Analyse des systèmes de production

Le problème étudié, dans ce mémoire, tient compte de contraintes liées aux ressources ainsi que des aléas. Les aléas considérés représentent l'apparition de tâches imprévues au cours de l'exécution de la production qu'il est impératif d'insérer dans le programme d'exécution des tâches programmées au préalable, cela dans leur intervalle d'exécution.

I.8 Conclusion

Dans ce chapitre, nous avons défini et présenté l'ordonnancement en général. Nous avons introduit la fonction production et sa gestion. Les éléments d'un problème d'ordonnancement, les organisations d'ateliers et la classification des critères de performance ont été présentés.

Les approches de résolution des problèmes d'ordonnancement sur des ressources en parallèle et en temps-réel ont été présentées.

Le chapitre suivant sera consacré à l'énumération d'un certain nombre de travaux de recherche dédiés à l'ordonnancement sur des ressources en parallèle et l'ordonnancement dynamique. De même, une démarche de résolution du problème d'ordonnancement temps-réel sur deux ressources identiques en parallèle sera présentée.

Chapitre II :

L'Ordonnancement Temps-Réel Sur Des Ressources Identiques En Parallèle

II.1 Introduction

L'environnement à ressources parallèles représente un atelier multi ressources, dont toutes les ressources peuvent exécuter une tâche ne nécessitant qu'une. Il existe trois types de ressources en parallèle suivant leurs performances : identiques, uniformes et indépendantes (cf. Annexe I).

Les ressources considérées dans cette étude sont identiques. Elles sont disjonctives, elles n'exécutent qu'une tâche à la fois.

Un problème d'ordonnancement à ressources parallèles met en œuvre m ressources notées m_1,…, m_m et n tâches (ou travaux) notées 1, 2, 3,…. A chaque instant, une ressource exécute au plus une tâche à la fois et chaque tâche est exécutée au plus par une ressource à la fois avec un temps d'exécution noté p_i.

L'objectif de ce chapitre est de présenter :

- Ø Un état de l'art sur les algorithmes d'ordonnancement des tâches dans un environnement à ressources parallèles.
- Ø Un état de l'art sur l'ordonnancement en temps-réel.
- Ø Une démarche de résolution du problème d'ordonnancement en temps-réel sur deux ressources identiques en parallèle.

Nous présentons dans la deuxième section de ce chapitre, le problème général d'ordonnancement sur des ressources identiques en parallèle. Le problème d'ordonnancement temps-réel est présenté dans la troisième section. La quatrième section concerne la complexité du problème d'ordonnancement temps-réel sur des ressources identiques en parallèle.

La cinquième section présente un certain nombre de résultats obtenus dans les recherches sur l'ordonnancement sur des ressources identiques parallèles et en temps-réel.

Le problème d'ordonnancement temps-réel sur deux ressources identiques en parallèle est présenté dans la sixième section. Cette présentation se fera à travers les caractéristiques du problème et une démarche de résolution.

II.2 Les problèmes d'ordonnancement sur des ressources identiques en parallèle

Dans cette section, nous présentons le problème général de l'ordonnancement sur des ressources identiques en parallèle, qui nous intéresse dans cette étude.

II.2.1 Définition

Un environnement de ressources identiques parallèles représente un atelier de production (ou service) où les différentes ressources peuvent effectuer la même tâche qui peut être traitée sur une seule.

II.2.2 Caractéristiques générales

Les caractéristiques de base dans une organisation à ressources identiques parallèles sont les suivantes :

- Toutes les tâches peuvent être exécutées sur n'importe quelle ressource.
- Les tâches sont indépendantes entre elles.
- Toutes les tâches et toutes les ressources sont disponibles à l'instant zéro.
- Le temps de transport entre les ressources est négligeable.
- Les temps de préparation des ressources et des tâches sont négligeables.
- Les tâches peuvent observer des temps d'attente avant d'être exécutées sur une ressource.
- Les capacités de stockage des files d'attente sont illimitées.
- Les pannes ne sont pas considérées dans ce contexte.

II.2.3 Formulation du problème

Dans cette section, nous proposons une modélisation du problème d'ordonnancement sur des ressources en parallèle.

Un modèle de programme linéaire, [Pin 95], est donné dans le cas de la préemption des tâches.

n : nombre de tâches.

m : nombre de ressources.

d_i : date de livraison ou date échue de la tâche i.

p_i : la durée d'exécution de la tâche i.

Le problème considéré est formulé comme suit :

Minimiser C_{max}

Sous les contraintes suivantes :

$$\sum_{j=1}^{m} x_{ij} = p_i \qquad pour\ i = 1, \text{h}, n$$

$$\sum_{j=1}^{m} x_{ij} \le C_{\max} \qquad pour\ i = 1, \text{h}, n$$

$$\sum x_{ij} \le C_{\max} \qquad pour\ j = 1, \text{h}, m$$

$$x_{ij} \ge 0 \qquad pour\ j = 1, \text{h}, m \quad i = 1, \text{h}, n$$

La variable x_{ij} représente la durée totale d'exécution de la tâche i sur la ressource j.

Le premier ensemble de contraintes assure que chaque tâche i a besoin d'une durée de traitement p_i.

Le second ensemble de contraintes assure que la somme des durées de traitement sur toutes les ressources, de chaque tâche i est inférieure ou égale à la durée totale C_{max} (Makespan).

Le troisième ensemble de contraintes assure que la somme des durées opératoires sur chaque ressource est inférieure ou égale à la durée totale.

Puisque le C_{max} est une variable de décision, et n'est pas un élément du vecteur de ressource du programme linéaire, le deuxième et le troisième ensemble de contraintes peuvent être réécrits comme suit :

$$C_{\max} - \sum_{j=1}^{m} x_{ij} \ge 0 \qquad i = 1, \text{h}, n$$

$$C_{\max} - \sum_{i=1}^{n} x_{ij} \ge 0 \qquad j = 1, \text{h}, m$$

Ce programme linéaire se résout en un temps polynomial, mais la solution du PL ne précise pas l'ordonnancement final optimal, car la solution obtenue spécifie uniquement la quantité de temps de traitement de la tâche i sur la ressource j. Cependant, avec cette information l'ordonnancement final peut être facilement construit [Sac 02].

Pour ce même cas (préemptif), l'algorithme de Mc Naughton, [Mc N 59], est connu pour être optimal. Il consiste à calculer la durée totale de production optimale $C^*_{\max} = \max\left\{\max_i p_i, \frac{1}{m}\sum_i p_i\right\}$, puis, d'ordonnancer les tâches une à une en complétant chaque machine jusqu'au C_{max}.

Comme le problème non préemptif est connu pour être un problème NP-Difficile, et qu'il n'existe pas d'algorithme optimal pour le résoudre, des heuristiques permettant de le résoudre sont proposées par la communauté scientifique telle que la règle LPT, [Gra 66], et bien d'autres. Dans le cas des contraintes de précédence entre les tâches et des durées opératoires unitaires ($p_i=1$), l'algorithme de Hu, [Hu 61], est optimal.

II.3 Les problèmes d'ordonnancement temps-réel

Cette section est consacrée à la présentation du problème d'ordonnancement temps-réel.

II.3.1 Introduction

Les notions de temps-réel, agir et réagir en temps-réel, travailler en temps-réel ont pris une plus grande ampleur avec l'évolution technologique due au développement de l'informatique d'où l'importante place qu'ont acquise les systèmes temps-réel. Ces derniers recouvrent un spectre d'applications très étendu qui va du simple contrôle de systèmes de commande, au contrôle du trafic aérien, des télécommunications, en passant par les systèmes de défense militaire, installations nucléaires et bien d'autres. Les systèmes temps-réel sont des systèmes de traitement dont la validité est conditionnée non seulement par la correction des résultats mais surtout par les dates auxquelles ceux-ci sont délivrés. En effet, ces systèmes sont essentiellement caractérisés par des contraintes de temps sur les actions à entreprendre, qu'il faut respecter de manière plus ou moins critique.

Grand nombre d'applications temps-réel nécessitent, pour leur exécution, des systèmes très stricts d'une fiabilité incontestable car la moindre erreur dans le respect des contraintes temporelles peut entraîner des situations à conséquences graves. Les systèmes supportant ce genre d'applications doivent offrir un maximum de garantie[12] et de déterminisme[13].

Un moyen efficace d'imposer le respect des contraintes temporelles est l'ordonnancement. Généralement, par ordonnancement temps-réel, on entend l'ordonnancement des tâches afin de pouvoir les exécuter dans les délais spécifiés par l'application. Toutefois, la notion d'ordonnancement peut également intervenir à d'autres niveaux (communications, allocation, mémoire) afin de garantir l'exécution des tâches.

Avec les demandes incessantes en applications temps-réel plus souples, telles que les communications multimédias, les systèmes multi robots coopératifs ou autonomes, les bases

[12] La garantie des systèmes temps-réel est assurée par le respect des contraintes temporelles lors de l'exécution de l'application.

[13] Le déterminisme d'un système représente la capacité, de ce dernier, à reproduire un comportement identique dans des conditions identiques.

de données temps-réel que nous trouvons dans les systèmes de radars, la production, etc., la nouvelle génération de systèmes temps-réel est dynamique et plus flexible[14]. Ces systèmes plus souples sont à notre avis adaptés aux besoins de classes plus larges d'applications.

Un système informatique temps-réel est un système qui est étroitement lié à son environnement. La propriété du temps-réel est attribuée si l'exactitude du système est déterminée par les dates auxquelles les résultats d'exécution sont disponibles.

La performance d'un système de production est désormais plurielle et multidimensionnelle. Elle doit être évaluée globalement et sur l'ensemble du cycle de vie du système et des produits réalisés. Elle intègre non seulement les notions de coût, de délai, de qualité mais également de flexibilité, de réactivité, de proactivité, de robustesse[15] et de valeurs[16]. En outre, les facteurs humains et sociaux, longtemps négligés, en sont dorénavant des paramètres prépondérants. Il en résulte un fort besoin de méthodologie et d'outils pouvant aider les décideurs à mieux appréhender la notion de performance et à l'évaluer lors de la conception, l'exploitation et la reconfiguration d'un système de production.

II.3.2 Description du problème d'ordonnancement à ressources identiques parallèles en temps-réel

Le problème posé consiste à ordonnancer sur plusieurs ressources un certain nombre de tâches arrivant aléatoirement. Dans cette section, nous présentons les caractéristiques et les paramètres de ce problème traité dans cette étude.

Il faut ordonnancer un ensemble P de tâches programmables et un autre ensemble A de tâches aléatoires.

La tâche $P_i \in P$ est caractérisée par :

- r_i : la date de disponibilité,
- p_i : la durée opératoire,
- d_i : la date échue,
- [ri di] : l'intervalle d'exécution de la tâche.
- Toutes les tâches sont supposées connues à l'instant zéro et que la relation :

$$d_i - r_i \geq p_i \qquad i = 1 \text{ h } n$$

est toujours satisfaite.

[14] Un système de production est dit flexible s'il présente une forte capacité d'adaptation à différentes situations.
[15] Voir Annexe II.
[16] Valeurs liées à la satisfaction du client.

La tâche $A_i \in A$ est caractérisée par :

- r_j : la date de disponibilité,
- p_j : la durée opératoire,
- d_j : la date échue.

Ø Processus d'arrivée

La variable aléatoire considérée X, prend comme valeur le temps séparant l'occurrence de deux tâches aléatoires consécutives et suit une loi exponentielle de paramètre s. Sa densité s'écrit :

f (t) = $\frac{1}{s}\exp\left(-\frac{1}{s}t\right)$ pour tout t≥ 0 et s > 0 (s :est un paramètre du problème).

Ø Les durées opératoires

Les durées opératoires sont aussi aléatoires et sont modélisées par une loi de probabilité dont la variable aléatoire est notée Z, la moyenne μ et l'écart type σ . Les valeurs prises par Z donnent les temps opératoires. Dans notre étude, nous supposons que les durées opératoires suivent une loi triangulaire (Min, Mode, Max), sa densité de probabilité s'écrit :

$$g(t)=\begin{cases}\dfrac{2(t-Min)}{(Mode-Min)(Max-Min)} & pour\ \ Min \le t \le Mode \\ \dfrac{2(Max-t)}{(Max-Mode)(Max-Min)} & pour\ Mode \le t \le Max \\ 0 & \sin on\end{cases}$$

Min, donne une estimation de la durée de la tâche en dessous de laquelle il est impossible de descendre.

Max, donne une estimation maximale.

Mode, désigne la valeur la plus fréquente ou celle qui a la plus grande probabilité d'apparition.

Ø Dates échues

La variable aléatoire donnant les valeurs des dates échues s'écrit $D = X + Ct$ (Ct : un second paramètre du problème), d'où :

$r_j = r_{j-1} + x$, x : la valeur prise par la variable aléatoire X, ($r_0 = 0$).

$d_j = r_{j-1} + d = r_j + Ct$, d : la valeur prise par la variable aléatoire D.

Nous supposons que les tâches aléatoires et programmables ne peuvent être interrompues et ne peuvent être réalisées simultanément sur la même ressource. Les hypothèses suivantes sont à respecter :

- Il existe un ordonnancement admissible des tâches programmables permettant à la ressource disjonctive d'exécuter toutes les tâches dans leurs intervalles respectifs.
- Une tâche aléatoire est rejetée s'il n'est pas possible de l'exécuter dans son délai.
- La préemption des tâches est interdite.

II.4 Complexité du problème

La qualité d'un ordonnancement, voire sa validité, dépend de nombreux paramètres tant au niveau de la structure de l'atelier que de la ressource. Il est à noter que la recherche d'un ordonnancement optimal, le plus efficace, est en général un problème NP-Difficile. Dans ce cas, nous recherchons des heuristiques.

Ourari, dans [Our 01], a montré que les problèmes d'optimisation combinatoire, dans le cas d'une ressource, sont des problèmes NP-Difficiles.

Derbala, dans [Der 01], considère que dans le cas d'une ressource unique, le problème du Makespan ne se pose pas. Le Makespan est égal à une constante pour toute séquence de n tâches données. Dans le cas de ressources multiples, le problème du Makespan n'est pas trivial.

Le problème $Pm//C_{max}$ est un problème NP-Difficile. Il n'est pas facile à résoudre puisque même le simple cas d'ordonnancement sur deux ressources a été prouvé être NP-Difficile [Kar 72], [Pin 95] et [Mok 04].

Lenstra & al., dans [Len 77], ont montré que le problème d'ordonnancement sur des ressources en parallèle sous contraintes des dates de disponibilité est un problème NP-Difficile pour tout critère.

Du fait que le problème $1/r_i/L_{max}$, par exemple, est NP-Difficile au sens fort implique que les problèmes $1/r_i/\sum U_i$ *et* $1/r_i/\sum T_i$ sont aussi NP-Difficiles au sens fort [Pin 95].

Babu & al., dans [Bab 04], ont montré que le problème $1/r_i/\sum w_i U_i$ est NP-Difficile au sens fort.

Yugma, dans [Yug 04], mentionne que Brucker & al. (1977) ont montré que le problème $1//\sum w_i T_i$ est NP-Difficile au sens fort.

Ce qui permet de conclure que le problème $Pm//C_{max}$ est NP-Difficile de même que pour le problème $Pm/r_i/C_{max}$.

En supposant que la préemption est interdite, nous pouvons avancer que le problème en temps-réel non préemptif $Pm / r_i / C_{max}$ est un problème NP-Difficile.

Etant donné que le problème, considéré dans cette étude, consiste à ordonnancer en temps-réel des tâches intervenant aléatoirement sur deux ressources identiques en parallèle, et que, le problème d'ordonnancement $Pm / r_i / C_{max}$ est supposé NP-Difficile, nous pouvons avancer que le problème $P2 / r_i / C_{max}$ est NP-Difficile.

II.5 Ordonnancement statique et ordonnancement dynamique

Cette section est réservée à la présentation d'un certain nombre de travaux concernant l'ordonnancement sur des ressources en parallèle et en temps-réel.

II.5.1 Ordonnancement statique

De nombreux travaux ont été effectués sur l'ordonnancement sur des ressources en parallèle. Un certain nombre d'ouvrages consacrés à l'ordonnancement, parmi eux [Got 93], [Law 93], [Pin 95] et [Chu 96], en citent quelques uns.

II.5.1.1 Le problème $Pm//C_{max}$

Dans cette sous section, nous rapportons un grand nombre de ces réalisations, qui nous paraissent les plus pertinentes.

Le problème consistant à ordonnancer un ensemble de tâches sur un ensemble de ressources identiques en parallèle, sans contraintes supplémentaires en minimisant la durée totale (Makespan), est NP-Difficile étant donné que le problème à deux ressources ($P2//C_{max}$) a été prouvé être NP-Difficile pour une réduction du problème de bipartitionnnenment par Karp [Kar 72].

Etant donné qu'il n'est pas possible de trouver un algorithme polynomial permettant de résoudre ce type de problèmes, des heuristiques de résolution sont proposées dans la littérature. Une classe de ces heuristiques est composée d'algorithmes de liste où les tâches sont affectées aux ressources en suivant l'ordre d'une liste construite à partir d'un certain critère.

L'algorithme List Scheduling [Gra 66] appliqué au problème $Pm//C_{max}$ a une performance[17] égale à $(2 - 1/m)$. L'heuristique LPT (Longest Processing Time) classe les

17 Performance d'une heuristique : $R_H = Z^H / Z^*$, où Z^H est la valeur du critère de la solution donnée par l'heuristique et Z^* est la valeur du critère pour une solution optimale développée.

tâches par ordre décroissant des durées opératoires. De complexité *O (n log n)*, elle a une performance $R_{LPT} = 4/3 - 1/3m$ [Gra 66]. Coffman, dans [Cof 76], propose une autre performance pour cette règle prenant en compte le nombre *k* de tâches assignées à une ressource dont la dernière tâche termine l'ordonnancement : $R_{LPT}(k) = 1 + 1/k - 1/ k m$. La meilleure performance de cette heuristique est obtenue lorsque le nombre de tâches augmente.

Plus tard, Ibarra et Kim [Iba 77] ont prouvé que $R_{LPT} = 1 + 2(m -1) / n$ pour $n \geq 2(m - 1)\ \pi$ et $\pi = max (pi) / min (pi)$. D'autres algorithmes de liste ont été développés.

Nous citons également l'algorithme MULTIFIT issu de l'analogie entre le problème $P//C_{max}$ et le problème du sac à dos [Cof 78]. Cette heuristique détermine une valeur C comme capacité maximale des ressources puis considère les ressources une à une et leur affecte des tâches en suivant la règle FFD (First Fit Decreasing). FFD affecte la tâche la plus longue respectant la capacité restante de la ressource. L'algorithme vérifie la faisabilité de la solution trouvée pour la capacité C donnée, et recherche la valeur de C minimale pour laquelle une solution réalisable existe.

D'autre part, un algorithme de programmation dynamique a, également, été proposé pour $P//C_{max}$ par Rothkopf [Rot 66]. Il permet de trouver la solution optimale en un temps $O (n C^m)$ et ne peut être utilisé que pour de petites valeurs de m et de C (borne supérieure du C_{max}).

En relaxant la contrainte de non préemption, le problème devient facile et peut être résolu optimalement en un temps polynomial, en appliquant la méthode de Mc Naughton [Mc N 59]. La complexité de cet algorithme est *O (n)*.

Considérons maintenant le cas où il existe des contraintes de précédence entre les tâches. Ullman [Ull 76] a tout d'abord prouvé que la restriction de ce problème aux tâches unitaires était NP-Difficile.

L'utilisation des algorithmes de liste, comme pour le problème $P//C_{max}$, peut générer des ordonnancements de comportements non attendus. En effet, les anomalies découvertes par Graham [Gra 66], montre que la longueur d'un ordonnancement construit à partir d'algorithmes de liste peut augmenter si, par exemple, le nombre de processeurs augmente, si les durées opératoires diminuent, si les contraintes de précédence sont relaxées ou encore si la liste de priorité change. Il existe cependant des cas particuliers pour lesquels une résolution polynomiale est envisageable. En effet, dans le cas *P/intree*, $p_i = 1/C_{max}$ (ou *P/out tree*, $p_i = 1/C_{max}$), Hu [Hu 61] a proposé un algorithme polynomial basé sur la notion de niveau de tâches. De même, le cas $P2/prec; pi = 1/C_{max}$ est un autre cas particulier

pour lequel Coffman et Graham [Cof 72] ont proposé un algorithme en $O(n^2)$ utilisant l'algorithme de Hu.

Considérons maintenant le problème de l'ordonnancement sur ressources parallèles identiques avec des contraintes de précédence entre les tâches et l'autorisation de préemption. Dans le cas général, ce problème est NP-Difficile [Ull 76]. Néanmoins, deux cas particuliers *(P2/pmtn; prec/*C_{max}*, P/pmtn; forest/*C_{max}*)* sont résolus par l'algorithme de Muntz et Coffman [Mun 70] basé sur une réduction des chemins critiques en partageant les processeurs entre les tâches de plus haut niveau. Le niveau d'une tâche i étant composé du temps nécessaire pour compléter la tâche i, plus la somme des temps d'exécution des tâches situées sur le plus long chemin entre i et une tâche terminale. Ainsi, le niveau d'une tâche en cours d'exécution décroît.

II.5.1.2 Le problème *Pm//*$\sum C_i$

Considérons la recherche d'un ordonnancement d'un ensemble de tâches sur un ensemble de ressources identiques parallèles minimisant la somme des dates de fin d'exécution des tâches. La généralisation de l'algorithme SPT (Shortest Processing Time), développé pour le problème *1//*$\sum C_i$, est optimal pour le cas de plusieurs ressources en parallèle. Cet algorithme de liste classe les tâches par ordre croissant de leur durée d'exécution et les affecte, dans cet ordre, à la première ressource libre. La complexité de cet algorithme est O (*n log n*) pour la construction de la liste.

En considérant des tâches ayant des contraintes de précédence, le problème devient rapidement difficile. En effet, comme l'indiquent Lenstra et Rinnoy Kan [Len 78], le problème restreint à deux ressources avec des durées opératoires unitaires (*P2/prec; pi = 1/*$\sum C_i$) est NP-Difficile. Seul le cas *P/outtree; pi = 1/*$\sum C_i$ peut être résolu polynomialement par une adaptation de l'algorithme de Hu développé pour *P/outtree; pi = 1/*C_{max}.

Mc Naughton [Mc N 59] ayant montré que la préemption n'apportait rien pour la minimisation de la somme des dates de fin d'exécution sur des ressources parallèles identiques, la résolution du problème *P/pmtn/*$\sum C_i$ est la même que pour *P//*$\sum C_i$. Dans le cas de la somme des dates de fin d'exécution pondérées, Brucker montre que pour le problème *P/pmtn/*$\sum w_i\ C_i$ **,** il existe un ordonnancement optimal non préemptif. Ce résultat est affirmé par Du & al. [Bru 99] et [Bru 01]. Ils ont montré que le résultat reste vrai si les

contraintes de précédence sont de la forme chaîne (*Pm/ chaîne, pmtn/* $\sum C_i$). Le problème *P2//* $\sum w_i C_i$ étant NP-Difficile, il en est, alors, de même pour *P2/pmtn/* $\sum w_i C_i$.

II.5.1.3 Les autres problèmes

D'autres contraintes, telles que les temps de préparation, les dates de disponibilité des tâches, le splitting des tâches,…, etc., ont fait objets de plusieurs études. Parmi ces dernières, nous citons les travaux de Yalaoui et Chu [Yal 99] consacrés au problème *Pm/Split,* S_{sd}, r_{sd}/C_{max} où les auteurs ont proposé une heuristique se décomposant en deux parties. Outre ceux-là, d'autres se sont intéressés à ce problème tels que Seraphini [Ser 96], Lenstra & al.[Len 90], Allahverdi & al. [All 99], Riotteau & al. [Rio 01] et bien d'autres.

Gharbi et Haouari ont consacré la plupart de leurs travaux de recherche, [Gha 02], [Gha 04], [Gha 05] et [Hao 04], au problème *Pm/* r_j ,q_j/C_{max} où ils ont repris les travaux de Carlier [Car 87] et [Car 98] et Gusfield [Gus 84] basés sur l'algorithme de Jackson. Des méthodes basées sur des parcours tronqués d'arborescence on été proposées par Tercinet [Ter 04].

II.5.2 Ordonnancement dynamique

Cette sous section est réservée à l'énumération de certains résultats concernant l'ordonnancement en temps-réel.

II.5.2.1 Les problèmes d'ordonnancement temps-réel liés à la nature des tâches

Liu et Layland [Liu 73] ont proposé un modèle de tâches répétitives[18]. Ce modèle a été amélioré en relaxant certaines contraintes [Leu 80], [Bar 90] et [Leh 90].

Deux algorithmes d'ordonnancement des tâches périodiques ont été proposés par Liu et Layland : RM (Rate Monotonic), ou "ordonnancement monotone par taux" et EDF (Earliest Deadline First), ou "échéance la plus courte d'abord".

Ces algorithmes sont les plus couramment utilisés au sein de la communauté temps-réel. Le modèle de tâches qu'ils utilisent est contraignant. Toutefois, de nombreuses extensions ont été proposées ([Car 94], [Sta 95], [Her 97] et [Riv 98]). Citons, à titre d'exemple, le support des contraintes de précédence [Bla 76] et [Che 90], ou encore la possibilité de partager des

[18] Les tâches répétitives sont des tâches faisant l'objet de plusieurs activations successives.

ressources entre les tâches [Sha 90] ainsi que d'autres contraintes [Buc 93], [Sak 95] et [Han 96].

F. Cottet & al. proposent une solution pour supprimer la variation sur le délai (ou gigue) entre deux fins d'activation successives d'une tâche [Cot 98]. La solution consiste à modifier les dates d'exécution au plus tôt et les échéances.

G. Coulson & al. définissent de nouvelles conditions d'ordonnançabilité avec la règle EDF sur des tâches dites "isochrones" ou "périodiques avec gigue" [Cou 97].

Han & al. proposent un modèle où l'activation des tâches n'est plus déterminée par une période mais par la fin d'exécution de l'activation précédente [Han 96]. Ce nouveau modèle de tâches n'est pas optimal s'il est ordonnancé par EDF. Leboucher propose un algorithme d'ordonnancement optimal basé sur EDF pour ce modèle : c'est l'algorithme de la dernière goutte ou LSD (Last Single Drop) [Leb 98]. L'algorithme consiste à retarder la fin d'exécution d'une activation jusqu'à son échéance. Cet algorithme est toutefois difficile à implanter. Dans la solution présentée par Han & al., on ajoute un délai fixe entre la fin d'exécution d'une activation donnée et le début de l'activation suivante.

Des méthodes travaillant sur les tâches éligibles[19] sont couramment utilisées dans les gros systèmes multiprocesseurs. Comme la seule information disponible est celle relative aux tâches pouvant être exécutées, l'unique préoccupation est de déterminer quelle tâche associer à quel processeur pour que tous les processeurs travaillent et terminent leurs travaux plus ou moins simultanément. Ces méthodes servent surtout à éviter que des processeurs restent inactifs alors que des tâches éligibles sont disponibles. Dandamundi [Dan 95] a proposé un système de queues hiérarchiques qui permet de répartir les tâches éligibles de la queue centralisée vers l'ensemble des queues distribuées d'un système de manière équitable. Lorsqu'un processeur génère une tâche, celle-ci est soit envoyée vers un autre processeur (aléatoirement, selon un algorithme donné [Ros 95],...), soit mémorisée dans la queue locale. Dans ce cas, une méthode de diffusion est implémentée pour transférer les tâches d'une queue à l'autre. Nous pouvons citer, par exemple, la méthode du gradient [Lin 87], celle de la diffusion initiée par le récepteur [Wil 93] ou celle qui consiste à aller "voler" du travail dans la queue d'un autre processeur [Blu 94].

Enfin, Jeffay & al. proposent un modèle appelé les "cadences d'exécution" (ou RBE pour Rate Based Execution) [Jef 99]. Les tâches d'une application multimédia y sont contraintes par une échéance et sont définies par une loi d'arrivée plus générale que celle

[19] Tâches prêtes à être exécutées.

initialement décrite par Liu et Layland. Une tâche peut être activée x fois pendant une période de temps y. Aucune hypothèse particulière n'est faite sur la distribution des x activations sur la période de temps y. Le modèle est proche (bien que plus général) du modèle LBAP (Linear Bounded Arrival Process) utilisé par Anderson [And 93]. Les cadences d'exécution ont été implantées dans YARTOS, un système d'exploitation pour les applications de vidéoconférence [Jef 91].

II.5.2.2 Les problèmes d'ordonnancement temps-réel liés aux graphes des tâches

Certains travaux ont été effectués dans le cas de l'ordonnancement temps-réel sur plusieurs processeurs. Nous trouvons des algorithmes statiques calculant au moment de la compilation sur quel processeur chaque tâche devra s'exécuter et à quel moment elle devra être initiée. Ces méthodes ont pour avantage d'effectuer une analyse globale et approfondie du graphe de tâche pour aboutir à une solution optimale ou presque optimale. Le problème général est en effet NP-Complet [Sar 93].

Des méthodes dites locales n'utilisent que l'information locale du graphe de tâche. Nous pouvons citer par exemple l'algorithme HFN (Heavy Node First) [Shi 90]. Une autre classe de méthodes utilise la longueur (pondérée ou non) du chemin séparant un noeud du graphe du noeud de sortie. Les tâches prioritaires sont celles ayant le plus long chemin. L'algorithme de base est appelé CPM (Critical Path Method), Shirazi l'a adapté en pondérant la longueur en fonction du nombre de successeurs (WL ou Weighted Length algorithm) [Shi 90]. Enfin, une troisième catégorie d'algorithmes classe les noeuds selon qu'ils appartiennent ou non au chemin critique. A titre d'exemple, Parker, [Par 97], traite un à un les noeuds du chemin critique en fonction de leur distance au noeud d'entrée (Decisive Path Algorithm ou DPS). Pour chaque noeud ayant des noeuds parents hors du chemin critique, les parents de ce noeud sont récursivement ajoutés dans la queue. Une fois les tâches ordonnées, elles sont allouées au processeur en donnant la priorité à celui qui aura terminé son travail le plus tôt.

D'autres algorithmes concernant les graphes sont disponibles ([Kim 88], [Joh 93], [Yan 94], [Ham 95], [Shu 96], [Ha 97] et [Wu 97]).

II.5.2.3 Les problèmes d'ordonnancement temps-réel liés à la production

Dans la littérature réservée à la recherche dans le domaine qui nous intéresse (production), un certain nombre de résultats est disponible pour piloter un système de production flexible réagissant aux aléas.

[Mon 98] dans son étude s'est intéressé à une période de 10 mn, et a appliqué les algorithmes génétiques pour la résolution du problème d'ordonnancement temps-réel.

Les travaux de Holloway et Nelson [Gia 88] proposent une technique d'ordonnancement intermittent adaptatif dont le principe est « on génère périodiquement un ordonnancement sur la base d'une approche de type statique certain puis on l'adapte par des règles de priorité pour tenir compte des arrivées de nouvelles tâches pendant la période qui sépare deux régénérations successives ».

Ainsi, [Our 01], le problème d'ordonnancement régénéré périodiquement est modélisé par un programme linéaire en nombre entiers (modèle statique) et résolu par les procédures de décomposition. A la base de la solution trouvée, nous effectuons une adaptation automatique à toute modification intervenant dans le système productif (particulièrement à l'arrivée de nouvelles tâches et aux variations constatées des temps opératoires) par l'utilisation de règles locales d'affectations. Ils créent, alors, deux listes de priorités, la première est générée lors du calcul centralisé de l'ordonnancement, et la seconde est régénérée localement lors des arrivées des nouvelles tâches. Lorsqu'une ressource se libère, l'arbitrage entre deux tâches qui ont la plus forte priorité dans chacune des deux listes, se fait sur la base de la plus faible marge disponible (différence entre le temps restant avant la date de livraison et le cumul des temps opératoires restant à faire).

De là, nous aboutissons à un nouvel ordonnancement prévisionnel ayant intégré la nouvelle tâche apparue.

Comme dans le cas des systèmes informatiques, l'approche la plus utilisée est la gestion des files d'attente basée sur des règles de priorité.

Plusieurs règles d'ordonnancement sont possibles. Ces règles appelées aussi règles de priorité, ou dispatching rules dans la littérature anglo-saxone, ont donné matière à de nombreux travaux de recherche pendant ces dix dernières années et spécialement pour résoudre les problèmes d'ordonnancement dynamiques des systèmes de type Job Shop [Our 01].

Blackstone & al., [Bla 82], définissent ces règles par « règles utilisées pour sélectionner la prochaine tâche à traiter parmi les tâches en attente de la ressource ».

Dans [Yan 97], la règle d'ordonnancement est représentée par l'expression logique suivante :

IF *conditions dans ensemble C sont vérifiées* ***THEN***

les actions dans ensemble A_1 *sont prises* ***ELSE*** A_2

Un grand nombre de règles d'ordonnancement est disponible dans la littérature. Panwalker et Iskander, dans [Pan 77], en rapportent plus de 113 règles et on en propose toujours dans les travaux de recherche. La question qui se pose toujours est : existe-t-il une règle d'ordonnancement meilleure qu'une autre ? Le choix de la règle peut se faire en fonction de ses performances par rapport à un critère d'évaluation ou de son coût de mise en œuvre [Meb 95] et [Our 01].

On définit une règle de priorité comme étant un algorithme qui calcule un nombre ou index *'PN_j'* de priorité pour chaque tâche *j* en attente dans la file de la ressource. La tâche qui a la priorité la plus élevée est choisie pour être exécutée en premier. Le calcul peut être basé sur une des caractéristiques suivantes [Meb 95] :

- Ø Date d'arrivée dans la file t_j
- Ø Durée opératoire de la tâche p_j
- Ø Date échue de la tâche d_j
- Ø Date échue du job correspondant d_j^*
- Ø Nombre de tâches restant du job n_j

L'ordonnancement est souvent construit selon la valeur minimale du nombre *'PN_j'*. On distingue ainsi deux types de règles [Our 01] :

* Les règles simples :

- √ First In First Out (FIFO): $PN_j = t_j$
- √ Shortest Processing Time (SPT) : $PN_j = p_j$
- √ Earliest Due Date (EDD) : $PN_j = d_j$
- √ Remaining Operation (RO) $PN_j = n_j$
- √ Slack Time (ST) : $PN_j = d_j - p_j - t_{now}$, t_{now} est la date courante
- √ Critical Ratio (CR) : $PN_j = d_j - t_{now} / d_j - r_j$

* Les règles combinées :

- √ Modified Operation Date (MOD) : $PN_j = max\ (p_j + t_{now}, d_j)$
- √ Combinaison entre SL et SPT (TSL/SPT) [Meb 95] :

$$PN_j = max\ (d_j^*\ ,\ d_j^* - t_{now} - t_j^*/t_j^*)$$

L'efficacité de la règle dépend de la mesure de performance associée au critère d'évaluation. Un grand nombre de ces règles est disponible dans la littérature, 53 mesures de performance sont proposées dans [Kir 84].

Ourari, [Our 01], a traité le problème d'ordonnancement temps-réel dans le cas d'une seule ressource. Elle a proposé deux algorithmes d'ordonnancement basés sur les règles de priorité ainsi qu'une approche basée sur l'insertion de la tâche aléatoire dès son arrivée.

Notre travail constitue la suite logique des travaux de Ourari [Our 01]. Ses résultats sont adaptés au cas de ressources identiques en parallèle.

D'autres travaux ont été effectués sur une ressource. Citons à titre d'exemple, l'insertion d'une tâche en temps-réel [Dur 04]. Yugma et Dupont, [Yug 04], ont étudié un problème en ligne de sélection et d'ordonnancement de tâches sur une ressource où ils ont proposé une méthode itérative consistant à fixer un certain nombre de paramètres et à résoudre sur une période donnée un problème statique.

II.6 Ordonnancement temps-réel sur deux ressources identiques en parallèle

Très peu de résultats ont été obtenus au niveau de la résolution optimale des problèmes concernant les organisations à ressources multiples (ressources parallèles, Flow Shop simple, Flow Shop hybride, Job Shop simple, Job Shop hybride ainsi que les organisations Open Shop simple et Open Shop hybride). Les études théoriques ont surtout porté sur la recherche de résultats approchés avec une certaine garantie de performance en temps et en qualité. La plupart des tentatives de résolution et de recherche d'algorithmes approchés sont basées sur l'algorithme polynomial de Johnson [Joh 54] qui résout, de manière optimale, le problème Flow Shop avec deux machines.

L'algorithme de Mc Naughton [Mc N 59] résout de manière optimale les problèmes à ressources identiques parallèles (dans le cas préemptif). En effet, la plupart des problèmes sont NP-Complets si ce ne sont NP-Difficiles [Sac 02].

Il est prouvé, [Mok 83], que pour l'ordonnancement dynamique, dans le cas multi ressources (multi processeurs), aucun algorithme d'ordonnancement ne peut être optimal sans avoir connaissance complète à l'avance de toutes les échéances, de tous les temps d'exécution, de toutes les dates de démarrage des tâches. En pratique, des heuristiques sont utilisées pour résoudre ces problèmes.

II.6.1 Résolution du problème

La démarche de résolution s'effectue en deux étapes. La première étape consiste à résoudre le problème concernant les tâches programmables. Il est connu sous le nom d'ordonnancement statique. La seconde, se base sur l'ordonnancement obtenu au niveau de

l'étape précédente en incluant les nouvelles tâches imprévues. C'est l'ordonnancement dynamique.

Cette résolution doit respecter certaines contraintes en optimisant un critère précis.

Dans cette étude, le critère à optimiser est la durée totale de l'ordonnancement, le Makespan, sous des contraintes liées aux ressources, aux intervalles temporels et aux tâches elles-mêmes ainsi que le nombre de tâches aléatoires ordonnancées (exécutées).

La résolution du problème d'ordonnancement dynamique sur des ressources identiques en parallèle est un problème d'affectation des tâches à exécuter sur chaque ressource à un emplacement adéquat.

II.6.1.1 Présentation du problème

Il existe n tâches programmables à exécuter sur deux ressources identiques en parallèle, soit P l'ensemble de ces tâches, en plus d'un certain nombre de tâches imprévues à exécuter. Soit A, l'ensemble des tâches aléatoires.

La tâche $P_i \in P$ est caractérisée par : r_i, p_i et d_i.

La tâche $A_i \in A$ est caractérisée par : r_j , p_j et d_j.

II.6.1.2 Hypothèses

- Les tâches programmables sont toutes traitées dans leurs intervalles temporels.
- Les temps inter opératoires sont nuls.
- La préemption est interdite.
- Les temps morts sont minimisés.
- Toute tâche aléatoire induisant, lors de son exécution, un retard sur une tâche programmable est rejetée.

II.6.2 Ordonnancement statique

L'ordonnancement statique concerne le séquencement des tâches connues a priori sur les ressources.

II.6.2.1 Affectation des tâches aux ressources

La démarche à suivre est de répartir les n tâches prévisionnelles sur les deux ressources selon une règle d'ordonnancement permettant d'exécuter les tâches dans leurs intervalles temporels à l'intérieur d'une plage de temps la plus faible possible. La limite supérieure de

cette plage représente la borne supérieure qui doit être minimisée. Minimiser cette borne permet de minimiser la durée totale d'ordonnancement.

La règle d'ordonnancement choisie est basée sur l'ordre croissant des dates de disponibilité des tâches en respectant les délais.

Affecter les tâches aux ressources dans l'ordre croissant de leurs dates de disponibilité, en respectant leurs délais. De là, considérer la plus grande durée autant que borne supérieure.

$$BS = C_{\max} \tag{II.1}$$

Une fois la répartition achevée, deux problèmes d'ordonnancement à une ressource sont à résoudre. De là, considérer une ressource et trouver la séquence admissible PA[20] pour l'exécution des tâches prévisionnelles qui ne sera plus modifiée (ultérieurement). La même procédure est à appliquer à l'autre ressource.

A noter que pour une séquence admissible, il est possible de déduire une infinité d'ordonnancements admissibles[21]. D'où, la nécessité de déterminer des ordonnancements admissibles extrêmes[22] permettant d'exécuter les tâches aléatoires en les intégrant dans la séquence des tâches programmables.

Toute tâche prévisionnelle disponible et non exécutée par une ressource est en attente d'exécution et stockée dans une file appelée File TP.

II.6.2.2 Définition de la séquence admissible

En considérant une seule ressource à chaque fois.

Une solution du problème est donnée par $t_i, i = 1 \text{ i } n,$ les dates réelles de début des tâches programmables.

Elle est dite admissible car elle satisfait les contraintes de fenêtres temporelles et d'utilisation disjonctive de la ressource.

Les contraintes temporelles sont données par :

$$\begin{aligned} t_i &\geq r_i \quad , \quad i = 1, \text{h} , n \\ t_i &\leq d_i - p_i \, , \quad i = 1, \text{h} , n \end{aligned} \tag{II.2}$$

[20] La séquence admissible est l'ordre de traitement des tâches sur les ressources dans leurs intervalles temporels.

[21] Un ordonnancement est dit admissible s'il respecte toutes les contraintes du problème (dates limites, précédence,…).

[22] Les ordonnancements admissibles extrêmes sont les ordonnancements au plus tôt et au plu tard qui représentent la date de début au plus tôt et au plus tard de chaque tâche programmable sur chaque ressource.

Proposition

L'ordre des tâches défini par la séquence admissible $PA = \{P_1, P_2, \text{h}, P_n\}$ dépend des différents paramètres caractérisant les tâches programmables.

La règle de priorité est utilisée pour deux tâches consécutives, P_m et P_{m+1}, laquelle des deux sera exécutée en premier.

Soit $C_{i,}$ la date de fin d'exécution prévue pour la tâche P_i ($C_0 = 0$).

En considérant les fenêtres temporelles des tâches P_m et P_{m+1} et la date C_{i-1}, soient M et M' les marges libres définies respectivement par les formules suivantes quand P_{m+1} est ordonnancée au plu tard et au plus tôt :

$$M = d_{m+1} - p_{m+1} - \max(C_{m-1}, r_m) \quad \text{(II.3)}$$

$$M' = d_m - (\max(C_{m-1}, r_{m+1}) + p_{m+1}) \quad \text{(II.4)}$$

D'après l'hypothèse que toutes les tâches programmables sont exécutées dans leurs intervalles temporels par la ressource disjonctive, la relation $p_m \leq \max(M, M')$ est vérifiée pour toute tâche $P_i \in P$.

L'idée serait de placer la tâche P_m dans la marge libre maximale. Il faut, alors, chercher dans quel cas M ¤ M'.

A noter que si P_m et P_{m+1} sont permutées, la position des autres tâches n'est pas modifiée.

Remarque

Si la relation $p_m \leq \max(M, M')$ n'est pas vérifiée pour une tâche P_m donnée, ceci est équivalent à suspendre l'hypothèse. Ce cas n'est pas à prendre en considération.

$$\begin{aligned} M > M' &\Leftrightarrow d_{m+1} - p_{m+1} - \max(C_{m-1}, r_m) > d_m - (\max(C_{m-1}, r_{m+1}) + p_{m+1}) \\ &\Leftrightarrow d_{m+1} + \max(C_{m-1}, r_{m+1}) > d_m + \max(C_{m-1}, r_m) \end{aligned} \quad \text{(II.5)}$$

Selon les valeurs de d_m et d_{m+1}, deux cas sont à distinguer : $d_m < d_{m+1}$ et $d_m \geq d_{m+1}$. Les figures suivantes illustrent chacun de ces cas :

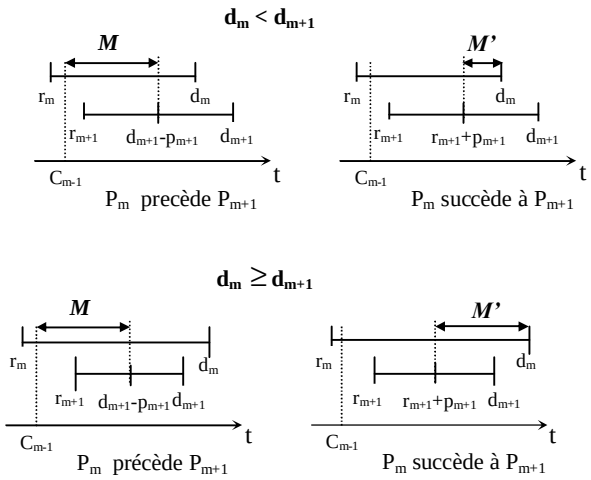


Figure II.1: les différents cas d'ordonnancement.

1er cas : $d_m < d_{m+1}$

$r_{m+1} > r_m$ vérifiée.

si $C_{m-1} < r_m$ alors la relation (II.5) s'écrit :

$d_{m+1} + r_{m+1} > d_m + r_m$: relation vraie.

- si $r_m < C_{m-1} < r_{m+1}$ alors la relation (II.5) s'écrit :

$d_{m+1} + r_{m+1} > d_m + C_{m-1}$: relation vraie.

- si $C_{m-1} > r_{m+1}$ alors la relation (II.5) s'écrit :

$d_{m+1} > d_m$: relation vraie.

Conclusion : quand $d_{m+1} > d_m$, les tâches P_m et P_{m+1} sont ordonnancées selon l'ordre croissant de leurs dates de disponibilité.

2nd cas : $d_m \geq d_{m+1}$

- si $C_{m-1} < r_{m+1}$ alors la relation (II.5) s'écrit :

$$d_{m+1} + r_{m+1} > d_m + \max(C_{m-1}, r_m) \quad (*)$$

Conclusion : les tâches P_m et P_{m+1} sont permutées si et seulement si la relation (*) est fausse.

- si $C_{m-1} \geq r_{m+1}$ alors la relation (II.5) s'écrit :

  $d_{m+1} > d_m$: relation fausse.

Conclusion

Il faut permuter les tâches P_m et P_{m+1}, elles doivent être ordonnancées dans l'ordre croissant de leurs dates échues.

Exemple

Il y a 20 tâches à exécuter sur deux ressources. Le tableau II.1 fournit les données caractérisant ces tâches :

i	1	2	3	4	5	6	7	8	9	10	11	12
r_i	0	1	3	5	6	7	9	10	14	16	19	20
p_i	2	3	4	6	3	5	3	2	6	4	3	5
d_i	6	8	32	20	19	25	20	30	28	35	37	38

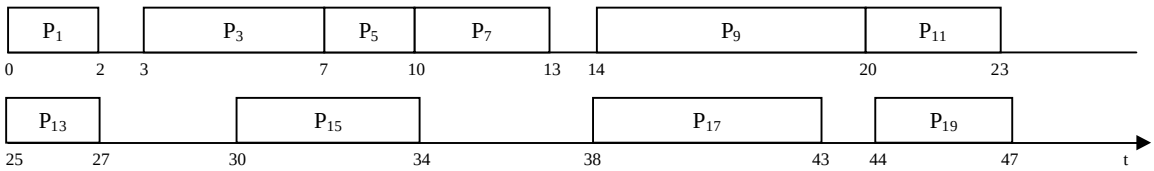
i	13	14	15	16	17	18	19	20
r_i	25	28	30	33	38	41	44	49
p_i	2	3	4	4	5	6	3	2
d_i	43	50	45	43	46	49	55	58

Tableau II.1 : Caractéristiques des tâches programmables.

a/ Affectation des tâches aux ressources

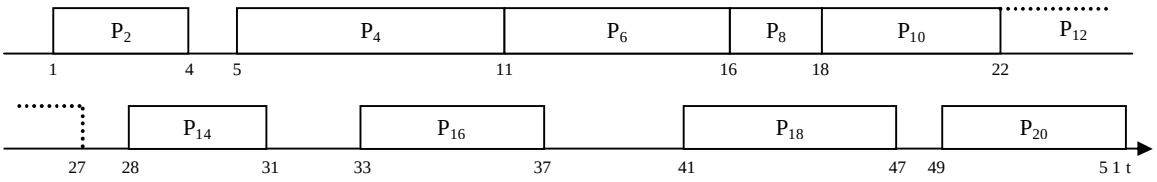
Ø *Répartir les tâches sur les ressources en respectant la borne et les délais*

La première ressource



(a)

La deuxième ressource



(b)

Figures II.2. (a) et (b) : Affectation des tâches aux ressources.

A noter que suivant cette répartition les délais sont satisfaits et BS = C_{max} = 51.

Remarque

Dans cet exemple le nombre de tâches affectées à chaque ressource est égal à 10 ce qui correspond à n/2 (20/2 = 10). C'est une simple coïncidence. Pour d'autres caractéristiques des tâches, une autre répartition sera donnée.

Etude des priorités

Pour cette étude, nous nous limiterons à la présentation de quelques priorités d'exécution entre les tâches programmables sur chacune des ressources afin d'illustrer le principe.

Ressource 1

Etude de la priorité entre P_1 et P_3

$C_0 = 0$, d_3 ¤ d_1 alors c'est le premier cas à considérer.

Conclusion : la tâche P_1 est ordonnancée avant la tâche P_3.

Etude de la priorité entre P_3 et P_5

$C_1 = 2$, d_3 ® d_5 alors c'est le deuxième cas à considérer.

C_1 ® r_5 alors la relation $d_5 + r_5$ ¤ $d_3 + \max(C_1, r_3)$ est fausse.

Conclusion : les tâches P_3 et P_5 doivent être permutées : ordonnancer la tâche P_5 puis P_3. A noter que P_3 est disponible à t = 3 et P_5 n'apparaît qu'à t = 6, comme il faut exécuter P_5 avant P_3 de ce fait il y a un temps mort. De là, par commodité, exécuter P_3 puis P_5.

Remarque

Possibilité de conserver le temps mort pour une éventuelle insertion d'une tâche aléatoire ultérieurement.

Etude de la priorité entre P_7 et P_9

$C_5 = 10$, d_7 ® d_9 Le premier cas est à considérer.

Comme $r_7 < C_5 < r_9$, nous avons la relation $d_9 + r_9 > d_7 + C_5$ vraie.

Conclusion : les tâches P_7 et P_9 sont ordonnancées dans l'ordre de leur apparition.

Etude de la priorité entre P_{15} et P_{17}

$C_{13} = 27$, d_5 ® d_7 alors c'est le premier cas à considérer.

C_{13} ® r_{15}, la relation $d_{17} + r_{17}$ ¤ $d_{15} + \max(C_{13}, r_{15})$ est vraie.

Conclusion : les tâches P_{15} et P_{17} sont ordonnancées dans cet ordre.

Résultat

La séquence admissible pour la première ressource est *PA1 = {P_1, P_3, P_5, P_7, P_9, P_{11}, P_{13}, P_{15}, P_{17}, P_{19}}.*

Ressource 2

Etude de la priorité entre P_2 et P_4

$C_0 = 0$, $d_2 < d_4$ le premier cas est à considérer.

$C_0 < r_2$, la relation $d_4 + r_4 > d_2 + \max(C_0, r_2)$ est vraie.

Conclusion : la tâche P_2 est ordonnancée avant la tâche P_4.

Etude de la priorité entre P_6 et P_8

$C_4 = 11$, $d_6 < d_8$ considérer le premier cas.

$C_4 > r_8$, la relation $d_8 > d_6$ reste toujours vraie.

Conclusion : ordonnancer la tâche P_8 après la tâche P_6.

Etude de la priorité entre P_{14} et P_{16}

$C_{12} = 27$, $d_{14} > d_{16}$ le deuxième cas à considérer.

$C_{12} < r_{14}$, la relation $d_{16} + r_{16} > d_{14} + \max(C_{12}, r_3)$ est fausse.

Conclusion : les tâches P_{14} et P_{16} doivent être permutées : ordonnancer la tâche P_{16} puis P_{14}. A noter que P_{14} est disponible à t = 28 et P_{16} n'apparaît qu'à t = 33, comme il faut exécuter P_{14} avant P_{16} de ce fait il y a un temps mort. De là, par commodité, exécuter P_{14} puis P_{16}.

Etude de la priorité entre P_{18} et P_{20}

$C_{16} = 37$, $d_{18} < d_{20}$ le premier cas est à considérer.

$C_{14} < r_{16}$, la relation $d_{20} + r_{20} > d_{18} + \max(C_{16}, r_{18})$ est vraie.

Conclusion : les tâches P_{18} et P_{20} sont ordonnancées dans l'ordre de leur disponibilité.

Résultat

La séquence admissible pour la deuxième ressource est *PA2= {P_2, P_4, P_6, P_8, P_{10}, P_{12}, P_{14}, P_{16}, P_{18}, P_{20}}.*

Pour le détail sur l'efficacité de l'approche proposée, voir [Our 01].

II.6.2.3 Définition des ordonnancements admissibles extrêmes

Cette sous section présente les ordonnancements admissibles au plus tôt et au plus tard.

II.6.2.3.1 Ordonnancement admissible au plus tôt

A partir de la séquence admissible définie par $PA = \{P_1, P_2, \text{ i } P_n\}$ et en appliquant les formules citées ci-dessous, les dates de début au plus tôt de chaque tâche P_i sont données. Soit ta_i cette date.

$$ta_i = \begin{cases} r_1 & i = 1 \\ \max\left(ta_{i-1} + p_{i-1}, r_i\right) & i = 2 \text{ i } n \end{cases} \qquad \text{(II.6)}$$

II.6.2.3.2 Ordonnancement admissible au plus tard

Partant de la date échue de la dernière tâche rangée dans PA, les formules suivantes donnent les dates de début au plus tard de chaque tâche. Soit tb_i cette date.

$$tb_i = \begin{cases} d_n - p_n & i = n \\ \min\left(d_i - p_i, tb_{i-1} - p_i\right) & i = n-1, \text{ i } ,1 \end{cases} \qquad \text{(II.7)}$$

Exemple de calcul des ordonnancements admissibles extrêmes :

Soit le tableau de l'exemple précédent.

Ressource 1

$ta_1 = 0$	$tb_1 = 4$
$ta_3 = 3$	$tb_3 = 10$
$ta_5 = 7$	$tb_5 = 14$
$ta_7 = 10$	$tb_7 = 17$
$ta_9 = 14$	$tb_9 = 22$
$ta_{11} = 20$	$tb_{11} = 32$
$ta_{13} = 25$	$tb_{13} = 35$
$ta_{15} = 30$	$tb_{15} = 37$
$ta_{17} = 38$	$tb_{17} = 41$
$ta_{19} = 44$	$tb_{19} = 52$

Ressource 2

$ta_2 = 1$	$tb_2 = 5$
$ta_4 = 5$	$tb_4 = 14$
$ta_6 = 11$	$tb_6 = 20$
$ta_8 = 16$	$tb_8 = 25$
$ta_{10} = 18$	$tb_{10} = 27$
$ta_{12} = 22$	$tb_{12} = 31$
$ta_{14} = 28$	$tb_{14} = 36$
$ta_{16} = 33$	$tb_{16} = 39$
$ta_{18} = 41$	$tb_{18} = 43$
$ta_{20} = 49$	$tb_{20} = 56$

II.6.3 Ordonnancement dynamique

Le principe est d'utiliser le programme d'exécution de l'ordonnancement statique et d'y insérer les tâches aléatoires parmi les tâches programmables tout en respectant l'objectif de production. De ce fait, partant de PA un nouvel ordonnancement incluant les tâches aléatoires est construit.

La question qui se pose est quelle est la tâche, programmable ou aléatoire, à exécuter dès la disponibilité d'une ressource.

Dans le cas de l'existence d'une tâche aléatoire dans une file d'attente, le raisonnement est simple et consiste à l'ordonnancer dès que des conditions, fixées au préalable, sont satisfaites, sinon elle est stockée dans cette file. Dans le cas contraire, lorsque plusieurs tâches

aléatoires sont en attente, il faut trouver une règle de sélection pour donner la priorité d'exécution à telle tâche sur la première ressource disponible plutôt qu'à une autre, qui sera exécutée ultérieurement sur la prochaine ressource qui se libérera. Entre temps, les autres tâches aléatoires sont en attente d'exécution dans leur file.

A préciser que toute tâche aléatoire doit être traitée dans son intervalle temporel, sinon elle est rejetée.

La résolution de ce problème consiste, alors, à trouver une règle d'ordonnancement donnant la priorité d'exécution à la tâche programmable ou à la tâche aléatoire.

Soit t_i, la date de début réel de la tâche i et n le nombre de tâches dans PA.

Les contraintes temporelles liées aux tâches programmables s'écrivent : [Our 01]

$t_i \geq ta_i \quad i = 1, \ldots, n$: L'exécution de la tâche i ne peut commencer avant la date ta_i ;

$t_i \leq tb_i \quad i = 1, \ldots, n$: La tâche i doit être exécutée avant la date tb_i ;

$t_i + p_i \leq t_{i+1} \quad i = 1, \ldots, n$: L'exécution de la tâche *i +1* ne peut commencer avant la fin d'exécution de la tâche *i*.

Les caractéristiques des tâches aléatoires sont connues a priori. Les contraintes temporelles liées aux tâches aléatoires s'écrivent :

$t_j + p_j \leq d_j$: L'exécution de la tâche j ne peut dépasser sa date échue ;

$t_j + p_j \leq tb_i$: La tâche j doit être exécutée avant la date tb_i.

A_j immédiatement successeur à P_i.

Dans ce type d'ordonnancement, suivant l'instant de traitement de la tâche aléatoire quatre cas, illustrés par le tableau ci-après, sont à distinguer.

Approche	Caractéristiques
I	Exécution de la tâche aléatoire dès son apparition
II	Rejet de la tâche aléatoire à l'instant de sa disponibilité
III	Traitement de la tâche imprévue, un temps d'attente après sa disponibilité
IV	Rejet de la tâche aléatoire un temps d'attente après sa date d'arrivée

Tableau II.2 : Classification des différentes approches.

Dans les deux premiers cas, I et II, la tâche aléatoire est traitée dès son arrivée si une ressource est libre au même instant, sinon elle est rejetée. Concernant les deux derniers cas,

un certain temps d'attente est toléré avant de décider le traitement ou non de la tâche imprévue étant donnée que les ressources sont occupées. De ce fait, il existe, en plus des files d'attente des tâches programmables, des files d'attente des tâches aléatoires. D'où, deux cas de figure sont à distinguer selon le nombre de files d'attente :

a. Une file d'attente des tâches aléatoires commune à toutes les ressources et des files d'attente des tâches programmables propres aux ressources.
b. Une file d'attente des tâches aléatoires et une file d'attente des tâches programmables auprès de chaque ressource.

Remarque

L'ordonnancement statique étant le même, il ne sera pas repris à chaque fois, ce qui est modifiée, c'est la borne supérieure (la borne d'affectation). Par conséquent, à chaque modification de cette dernière une procédure est à appliquer, un nouvel ordonnancement est généré.

Pour chaque cas, un C_{max} est associé. Ce C_{max} est la borne supérieure de cet ordonnancement.

Sachant que la préemption est interdite pour le problème considéré, il est nécessaire d'augmenter la valeur de cette borne dès que l'exécution d'une tâche risque d'être interrompue en temps-réel (en ordonnançant les tâches aléatoires).

La valeur de BS (C_{max}) est augmentée en lui affectant la nouvelle valeur de C_{max} (celle obtenue sur une ressource en ayant ordonnancé une tâche aléatoire). De ce fait, à partir de cette itération, considérer à chaque fois le nouveau C_{max} en incluant les tâches aléatoires.

Cette démarche représente la procédure de la borne supérieure illustrée ci-après.

Procédure BS

Etape 1 : Ranger les tâches dans l'ordre croissant des dates de leur disponibilité,

Etape 2 : Affecter les tâches aux ressources dans l'ordre croissant des dates de disponibilité en respectant les délais,

Etape 3 : Considérer $BS = C_{\max}$

Etape 4 : **Si** sur chaque ressource, en intégrant les tâches aléatoires, la borne est respectée **alors** continuer à affecter les tâches imprévues aux ressources,

Sinon

Si BS n'est pas respectée **alors** augmenter la valeur de BS en considérant le nouveau $C_{\max}$.

Sinon aller à **Etape 4**.

II.7 Conclusion

Une présentation générale d'un problème d'ordonnancement sur des ressources en parallèle et en temps-réel a été faite.

Nous avons tenté de rapporter le maximum de résultats d'études effectués, dans les deux cas (statique et dynamique), par la communauté scientifique dédiant ses travaux à l'ordonnancement sur des ressources parallèles et au temps-réel.

L'approche de résolution du problème d'ordonnancement temps-réel sur deux ressources en parallèle se base sur les règles de priorité.

En ordonnançant les tâches aléatoires, quatre cas de figure, selon le nombre de files d'attente de ces tâches et l'instant de leur exécution, ont été introduits. Deux approches parmi les quatre sont traitées dans cette étude (une file d'attente unique des tâches aléatoires et plusieurs files d'attente des tâches aléatoires).

Le chapitre suivant concernera l'ordonnancement temps-réel sur des ressources identiques en parallèle dans le cas de la disposition du système d'une seule file d'attente des tâches aléatoires. Nous considérons le cas d'une organisation à deux ressources identiques parallèles.

Chapitre III :

Première Approche : Une File Unique D'attente Des Tâches Aléatoires

III.1 Introduction

Ce présent chapitre est consacré à l'un des cas de figure des problèmes d'ordonnancement temps-réel sur des ressources identiques en parallèle. Ce problème considère deux ressources identiques parallèles. Auprès de chaque ressource existe une file d'attente des tâches programmables et une file d'attente des tâches aléatoires est commune à toutes les ressources. Pour la résolution de ce problème, deux algorithmes d'ordonnancement sont proposés, un algorithme d'ordonnancement au plus tôt et un autre au plus tard.

III.2 Résolution du problème

L'ordonnancement dynamique consiste à inclure dans la séquence admissible, de l'exécution des tâches programmables, les nouvelles tâches intervenant aléatoirement.

Dans ce cas, les tâches aléatoires apparaissant sont affectées à la même file d'attente (File TA), par contre, les tâches programmables sont affectées aux différentes files d'attente (File TP) respectives des ressources.

Que la tâche programmable soit exécutée au plus tôt ou au plus tard, deux algorithmes d'ordonnancement sont à appliquer :

1) Algorithme d'ordonnancement au plus tard.
2) Algorithme d'ordonnancement au plus tôt.

Quelques définitions et rappels sont nécessaires au préalable :

i : variable désignant l'ordre de la tâche programmable i = 1,…,n ;

k : variable désignant l'ordre de l'itération après exécution d'une tâche aléatoire ;

P_i : tâche programmable d'ordre i ;

t_{now} : l'instant courant ;

ta_i et tb_i : les dates du début au plus tôt et au plus tard de la tâche programmable (les ordonnancements admissibles extrêmes),

BS : la borne supérieure d'affectation des tâches aux ressources.

Les caractéristiques des tâches aléatoires sont connues a priori. Les contraintes temporelles liées aux tâches aléatoires s'écrivent :

$$t_j + p_j \le d_j$$
$$t_j + p_j \le tb_i \qquad \text{(III.1)}$$

Ces contraintes assurent que la tâche aléatoire et la tâche programmable sont exécutées dans leurs intervalles temporels respectifs à l'instant considéré.

Dans le pire des cas, les contraintes sur les intervalles temporels à considérer sont les suivantes :

La première condition de délai pour qu'une tâche aléatoire j puisse être exécutée s'écrit :

$$r_j + p_j \leq d_j$$

Si toutes les tâches aléatoires ne vérifient pas cette condition, alors, il est certain qu'aucune d'elles ne pourra être exécutée.

La deuxième condition d'espace est donnée par le critère suivant :

$$\min_{j \in A}(p_j) \text{ ® } \max_{i=1,n-1}\left[tb_i - (ta_i + p_i)\right]$$

Si ce critère est faux, alors, il est certain qu'aucune tâche aléatoire ne pourra être traitée par la ressource.

Une explication de ce critère est donnée comme suit :

Dans le chapitre précédent (cf. II.6.2.2), nous avons défini la séquence admissible PA des tâches programmables, chaque tâche est caractérisée par une fenêtre temporelle $[ta_i \;\; tb_i]$[23].

Si la tâche *i* est exécutée au plus tôt et la tâche *i+1* au plus tard, nous disposons, ainsi, d'une marge libre $m_{i,i+1} = tb_{i+1} - (ta_i + p_i)$ pour insérer éventuellement une ou plusieurs tâches aléatoires. Cette marge relative aux tâches P_i et P_{i+1} est maximale.

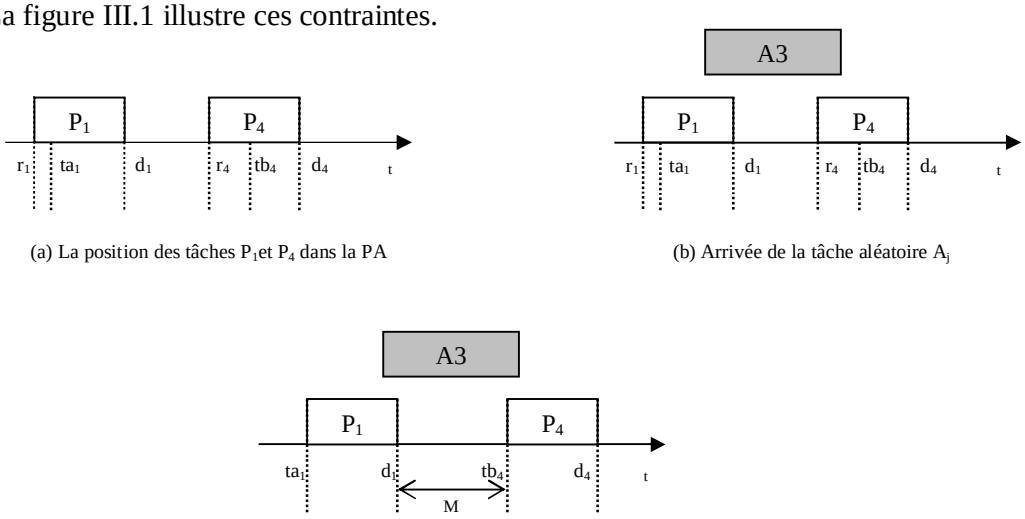
Si la plus faible durée opératoire, d'une tâche de l'ensemble A des tâches aléatoires, est supérieure à la plus grande marge (entre deux tâches programmables successives), soit $\min_{j \in A}(p_j) \text{ ¤ } \max_{i=1,n-1}\left[tb_i - (ta_i + p_i)\right]$, alors, nous sommes certains qu'aucune tâche aléatoires ne sera exécutée par la ressource.

Remarque

En temps-réel, à l'arrivée d'une tâche aléatoire la dernière tâche programmable, P_i, notée est exécutée ou en cours d'exécution par la ressource, si $p_j \text{ ¤ } \max_{i=1,n-1}\left[tb_i - (ta_i + p_i)\right]$, alors la tâches aléatoire *j* est automatiquement rejetée.

[23] ta_i date de début au plus tôt et tb_i date de début au plus tard pour $i = 1,\ldots, n$.

La figure III.1 illustre ces contraintes.



(a) La position des tâches P_1et P_4 dans la PA

(b) Arrivée de la tâche aléatoire A_j

(c) Ordonnancement de la tâche P_1 au plus tôt et la tâche P_4 au plus tard

Figure III.1 : La marge disponible permettant l'exécution de la tâche aléatoire A_j.

M est la marge disponible en ordonnançant les tâches programmables P_1 et P_4 respectivement au plus tôt et au plus tard. Cette marge ne permet pas d'ordonnancer la tâche aléatoire A_j intervenant dans le système dont la durée opératoire est la plus faible par rapport aux autres tâches aléatoires.

Remarque

Nous considérons le cas de deux ressources identiques en parallèle. Les deux algorithmes présentés, dans ce travail, sont généralisés au cas d'un nombre m de ressources identiques parallèles. Ceci ne modifie en rien l'étude de la complexité des algorithmes proposés, il suffit de considérer le nombre de ressources requis pour l'étude.

III.2.1 Algorithme d'ordonnancement temps-réel au plus tard

Cet algorithme donne la priorité à la tâche aléatoire dans l'utilisation des marges disponibles. La tâche aléatoire doit vérifier certaines conditions pour être ordonnancée. L'une de ces contrainte concerne son intervalle temporel (une tâche dont la date échue n'est pas respectée est rejetée $t_{now}+p_j \le d_j$). L'autre contrainte est liée à l'intervalle d'exécution des tâches programmables (l'ordonnancement d'une tâche aléatoire induisant un retard sur l'exécution d'une tâche programmable, ordonnancée au plus tard, est rejetée $t_{now} + p_j \le tb_i$).

L'algorithme suivant permet d'affecter les tâches aux ressources selon un ordonnancement au plus tard des tâches programmables :

Initialisation : i = 1, k = 0

Tant que t_{now} est inférieur à d_n **faire**

Si une tâche A_j intervient et toutes les ressources occupées **alors** $A_j \rightarrow$ File TA.

I. Pour chaque libération d'une ressource **faire** Rechercher dans File TA

Si une tâche A_j peut être exécutée satisfaisant les conditions $t_{now} + p_j \leq d_j \quad et \quad t_{now} + p_j \leq tb_i$ pour une ressource **alors** exécuter A_j , éliminer A_j de File TA, k = k + 1, aller à I

Sinon stocker A_j dans File TA, aller à I

Si la tâche A_j vérifie les conditions $t_{now} + p_j \leq d_j \quad et \quad t_{now} + p_j \leq tb_i$ pour plusieurs ressources **alors**

Si BS est respectée pour une ressource **alors** exécuter A_j, éliminer A_j de File TA, k = k + 1, aller à I

Sinon appliquer Procédure BS

Si la condition sur BS est vérifiée pour plusieurs ressources **alors** exécuter A_j sur une ressource selon une règle de priorité, éliminer A_j de File TA, k = k +1, aller à I

Sinon appliquer Procédure BS

Sinon stocker A_j dans File TA, aller à I

Si plusieurs tâches vérifient les conditions $t_{now} + p_j \leq d_j \quad et \quad t_{now} + p_j \leq tb_i$ **alors**

Si les conditions sont vérifiées pour une ressource **alors**

Si une tâche vérifie la condition sur BS **alors** exécuter A_j, k = k + 1, aller à I

Sinon appliquer Procédure BS

Sinon stocker A_j dans File TA, aller à I

Si plusieurs tâches vérifient la condition sur BS **alors** exécuter la tâche la plus prioritaire, affecter les autres tâches aléatoires à File TA, k = k + 1, aller à I

Sinon appliquer Procédure BS

Sinon stocker A_j dans File TA, aller à I

Si les conditions sont vérifiées pour plusieurs ressources **alors**

Si une tâche A_j vérifie la condition sur BS pour plusieurs ressources **alors** exécuter A_j sur une ressource, k = k + 1, aller à I

Sinon appliquer Procédure BS

Si plusieurs tâches vérifient la condition sur BS **alors** exécuter les tâches sur les ressources selon une règle de priorité, éliminer les tâches de File TA, k = k + 1, aller à I

Sinon appliquer Procédure BS

Sinon stocker les tâches dans File TA, aller à I

Sinon

Si $ta_i \geq t_{now}$ **alors** exécuter P_i , i = i + 1, $A_j \rightarrow$ File TA, aller à I

Sinon stocker les tâches dans File TA et File TP, aller à I

Fin (Pour)

II. Si une ressource est libre et A_j arrive **alors**

Si $t_{now} + p_j \leq d_j \quad et \quad t_{now} + p_j \leq tb_i$ **alors** exécuter la tâche A_j, k = k +1, aller à II

Sinon $A_j \rightarrow$ File TA

Si plusieurs ressources se libèrent et A_j arrive **alors**

Si $t_{now} + p_j \leq d_j \quad et \quad t_{now} + p_j \leq tb_i$ **alors**

Si BS est respectée pour une ressource **alors** exécuter A_j, k = k + 1, aller à II

Sinon appliquer Procédure BS

Sinon $A_j \rightarrow$ File TA

Si la condition sur BS est satisfaite pour plusieurs ressources **alors** exécuter A_j sur une ressource en utilisant une règle de priorité, k = k + 1, aller à II

Sinon appliquer Procédure BS

Sinon $A_j \rightarrow$ File TA

III. Pour chaque ressource disponible et $ta_i = t_{now}$ **faire**

exécuter chaque tâche P_i , i = i + 1, aller à III

Fin (Pour)

Fin (Tant que)

III.2.1.1 Complexité de l'algorithme proposé

L'algorithme d'ordonnancement au plus tard est constitué d'une boucle principale à trois parties indépendantes.

La première instruction conditionnelle de l'algorithme (**Si** une tâche A_j …) est exécutée, dans le pire des cas *l* fois, tel que *l* est le nombre total de tâches aléatoires apparues au cours de l'ordonnancement. La complexité de cette condition est *O (l)*.

La partie I

Cette partie ne contient qu'une seule boucle (**Pour** chaque …) à plusieurs instructions conditionnelles. Ces instructions seront exécutées, dans le pire des cas, une seule fois. Cette exécution est liée à la libération des ressources.

La boucle (**Pour** chaque…) est exécutée, dans le pire des cas, *m* fois (m étant le nombre de ressources). De là, la complexité de la partie I est *O (m)*.

La partie II

Cette partie est constituée de plusieurs instructions conditionnelles.

Les instructions (**Si** une ressource …) et (**Si** plusieurs ressources …) sont exécutées, dans le pire des cas, une fois.

L'instruction **Affecter A_i à FTA** est, aussi, exécutée dans le pire des cas une fois (une seule tâche qui arrive).

La complexité de cette partie est égale à *O (m)*.

La partie III

Elle est constituée d'une boucle **Pour**. Cette boucle dépend du nombre de ressources disponibles et du nombre de tâches prêtes à être exécutées (*n* dans le pire des cas).

La complexité de la boucle **Pour** est *O (n m)*.

Comme la boucle **Tant Que** est constituée des trois parties (I, II et III), la complexité de l'algorithme est *O (n m l)*.

III.2.1.2 Exemple

Considérant toujours l'exemple précédent (tableau II.1)

Le tableau suivant donne les caractéristiques de 10 tâches aléatoires à exécuter en plus des tâches programmables :

j	1	2	3	4	5	6	7	8	9	10
r_j	0	3	11	15	18	22	31	39	43	47
p_j	2	2	3	5	3	6	5	4	2	1
d_j	3	9	19	25	30	40	44	50	55	52

Tableau III.1 : Caractéristiques des tâches aléatoires.

Ø Affectation des tâches programmables aux ressources (voir figure II.2)

Ø Les séquences admissibles :

Sur R1 : $PA1 = \{P_1, P_3, P_5, P_7, P_9, P_{11}, P_{13}, P_{15}, P_{17}, P_{19}\}$

Sur R2 : $PA2 = \{P_2, P_4, P_6, P_8, P_{10}, P_{12}, P_{14}, P_{16}, P_{18}, P_{20}\}$

Ø Les ordonnancements admissibles extrêmes

Ressource 1

$ta_1 = 0$	$tb_1 = 4$
$ta_3 = 3$	$tb_3 = 10$
$ta_5 = 7$	$tb_5 = 14$
$ta_7 = 10$	$tb_7 = 17$
$ta_9 = 14$	$tb_9 = 22$
$ta_{11} = 20$	$tb_{11} = 32$
$ta_{13} = 25$	$tb_{13} = 35$
$ta_{15} = 30$	$tb_{15} = 37$
$ta_{17} = 38$	$tb_{17} = 41$
$ta_{19} = 44$	$tb_{19} = 52$

Ressource 2

$ta_2 = 1$	$tb_2 = 5$
$ta_4 = 5$	$tb_4 = 14$
$ta_6 = 11$	$tb_6 = 20$
$ta_8 = 16$	$tb_8 = 25$
$ta_{10} = 18$	$tb_{10} = 27$
$ta_{12} = 22$	$tb_{12} = 31$
$ta_{14} = 28$	$tb_{14} = 36$
$ta_{16} = 33$	$tb_{16} = 39$
$ta_{18} = 41$	$tb_{18} = 43$
$ta_{20} = 49$	$tb_{20} = 56$

Pour cet exemple BS = $C_{\max} = 51$.

En appliquant l'algorithme d'ordonnancement temps-réel au plus tard, nous avons :

t_{now} = 0

Les deux ressources sont disponibles ainsi que la tâche P_1. La tâche A_1 arrive.

Vérifier si les conditions :

$$t_{now} + p_j \le d_j \quad et \quad t_{now} + p_j \le tb_i$$

sont satisfaites.

Pour R1 : $0 + 2 < 3$ et $0 + 2 < 4$

Pour R2 : $0 + 2 < 3$ et $0 + 2 < 5$

Les conditions sont satisfaites pour les deux ressources.

Conclusion

Comme $t_{now} = t_{a1}$, alors exécuter P_1 sur la ressource 1. La ressource 2 étant libre à cet instant, par conséquent, ordonnancer A_1 sur la ressource 2.

t_{now} = 3

La ressource 2 est occupée à cet instant. La ressource 1 est libre et la tâche P_3 est disponible. La tâche A_2 intervient. La priorité est pour la tâche aléatoire.

Vérifier si les conditions :

$$t_{now} + p_j \le d_j \quad et \quad t_{now} + p_j \le tb_i$$

sont satisfaites.

$$3 + 2 < 9 \quad \text{et} \quad 3 + 2 < 10$$

Les conditions sont vérifiées.

Conclusion

La tâche A_2 est ordonnancée sur la ressource 1. La tâche P_3 est stockée dans File TP.

t_{now} = 5

A cet instant, les deux ressources se libèrent, la tâche P_4 est disponible et la tâche P_3 est en attente d'exécution. Aucune tâche aléatoire n'intervient

Conclusion

Exécuter la tâche P_3 sur R_1 et la tâche P_4 sur R_2.

t_{now} = 39

Les ressources sont occupées et la tâche A_8 intervient. La tâche A_7 est en attente d'exécution.

Conclusion

Stocker la tâche A_8 dans File TA. La tâche A_7 est rejetée car elle doit être exécutée au plus tard à cet instant ($t_{now} = 39$).

t_{now} = 46

La ressource 1 se libère. Les tâches A_8, A_9 et P_{19} sont en attente d'exécution.

Vérifier si les conditions

$$t_{now} + p_j \le d_j \quad et \quad t_{now} + p_j \le tb_i$$

sont satisfaites.

A_8 : $46 + 4 = 50$ et $46 + 4 < 52$ (tb_{19})

A_9 : $46 + 2 < 55$ et $46 + 2 < 52$ (tb_{19})

Les conditions sont vérifiées.

Conclusion

Du fait que les conditions sont satisfaites par les deux tâches pour la même ressource, leur affectation nécessite une règle de priorité. La règle de priorité choisie (parmi d'autres) est la règle FIFO (First In First Out). Par conséquent, exécuter la tâche A_8 sur la ressource 1 et affecter la tâche A_9 à File TA. Stocker la tâche P_{19} dans File TP de la ressource 1.

t_{now} = 50

Les deux ressources sont libres et les tâches A_{10}, P_{19} et P_{20} sont en attente d'exécution.

Vérifier si les conditions

$$t_{now} + p_j \le d_j \quad et \quad t_{now} + p_j \le tb_i$$

sont satisfaites.

A_{10} : $50 + 1 < 52$ et $50 + 1 < 52$ (tb_{19})

$50 + 1 < 52$ et $50 + 1 < 56$ (tb_{20})

Conclusion

En ordonnançant la tâche A_{10} sur le ressource 1, la borne supérieure sera augmentée de trois unités de temps ($BS = 54$), alors, qu'elle ne sera augmentée que de deux unités de temps ($BS = 53$) si elle est ordonnancée sur la ressource 2. Par conséquent, exécuter la tâche P_{19} sur

la ressource 1 et la tâche A_{10} sur la ressource 2. Stocker la ressource P_{20} dans File TP de la ressource 2. Elle est considérée à partir de instant comme suit : $BS = C_{\max} = 53$.

t_{now} = 51

La ressource 2 se libère et la tâche P_{20} est en attente d'exécution.

Conclusion

Exécuter la tâche P_{20} sur la ressource 2.

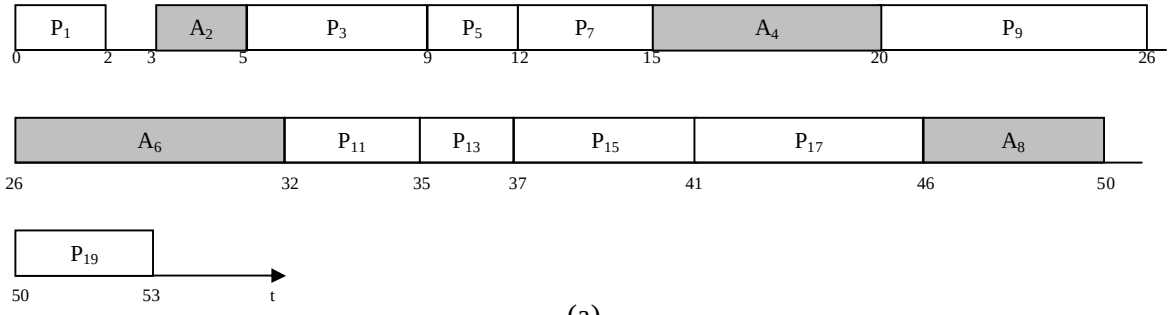
Résultat

Neuf tâches aléatoires ont été insérées dans les séquences admissibles des ressources et une seule a été rejetée. BS n'a été augmentée qu'une seule fois de deux unités de temps.

La séquence sur R1 est :{P_1, A_2, P_3, P_5, P_7, A_4, P_9, A_6, P_{11}, P_{13}, P_{15}, P_{17}, A_8, P_{19}}.

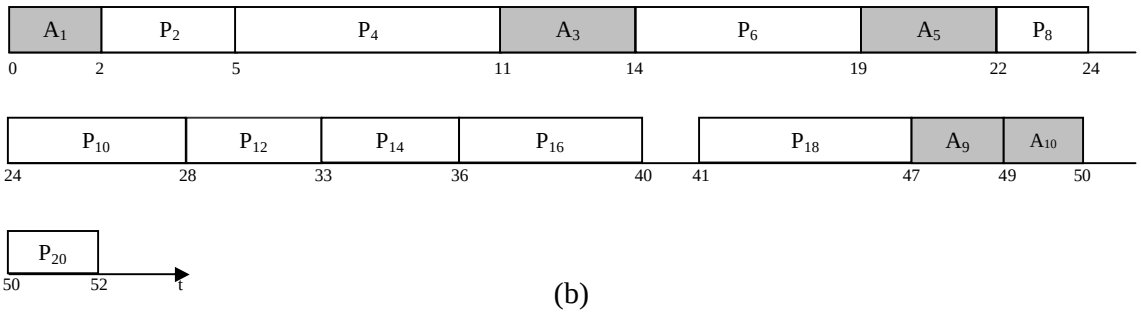
La séquence sur R2 est {A_1, P_2, P_4, A_3, P_6, A_5, P_8, P_{10}, P_{12}, P_{14}, P_{16}, P_{18}, A_9, A_{10}, P_{20}}.

Ressource 1



(a)

Ressource 2



(b)

Figure III.2 : Ordonnancement temps-réel au plus tard.

III.2.2 Algorithme d'ordonnancement temps-réel au plus tôt

Cet algorithme donne la priorité à la tâche programmable dans l'utilisation des marges disponibles. La tâche programmable est exécutée dès que l'instant courant est égal à la date de début au plus tôt de cette tâche $ta_i \geq t_{now}$.

L'algorithme suivant permet d'affecter les tâches aux ressources selon un ordonnancement au plus tôt :

Initialisation : i = 1, k = 0

Tant que t_{now} est inférieur à d_n **faire**

Si une tâche A_j intervient et toutes les ressources occupées **alors** $A_j \rightarrow$ File TA.

I. Pour chaque libération d'une ressource **faire**

Si $ta_i \geq t_{now}$ **alors** exécuter P_i, i = i + 1, $A_j \rightarrow$ File TA, aller à I

Sinon

Rechercher dans File TA

Si une tâche A_j peut être exécutée satisfaisant les conditions $t_{now} + p_j \leq d_j \quad et \quad t_{now} + p_j \leq tb_i$ pour une ressource **alors** exécuter A_j , éliminer A_j de File TA, k = k + 1, aller à I

Sinon stocker A_j dans File TA, aller à I

Si la tâche A_j vérifie les conditions $t_{now} + p_j \leq d_j \quad et \quad t_{now} + p_j \leq tb_i$ pour plusieurs ressources **alors**

Si BS est respectée pour une ressource **alors** exécuter A_j, éliminer A_j de File TA, k = k + 1, aller à I

Sinon appliquer Procédure BS

Si la condition sur BS est vérifiée pour plusieurs ressources **alors** exécuter Aj sur une ressource selon une règle de priorité, éliminer A_j de File TA, k = k +1, aller à I

Sinon appliquer Procédure BS

Sinon stocker A_j dans File TA, aller à I

Si plusieurs tâches vérifient les conditions $t_{now} + p_j \leq d_j \quad et \quad t_{now} + p_j \leq tb_i$ **alors**

Si les conditions sont vérifiées pour une ressource **alors**

Si une tâche vérifie la condition sur BS **alors** exécuter A_j, k = k + 1, aller à I

Sinon appliquer Procédure BS

Sinon stocker A_j dans File TA, aller à I

Si plusieurs tâches vérifient la condition sur BS **alors** exécuter la tâche la plus prioritaire, affecter les autres tâches aléatoires à File TA, k = k + 1, aller à I

Sinon appliquer Procédure BS

Sinon stocker A_j dans File TA, aller à I

Si les conditions sont vérifiées pour plusieurs ressources **alors**

Si une tâche A_j vérifie la condition sur BS pour plusieurs ressources **alors** exécuter A_j sur une ressource, k = k + 1, aller à I

Sinon appliquer Procédure BS

Si plusieurs tâches vérifient la condition sur BS **alors** exécuter les tâches sur les ressources selon une règle de priorité, éliminer les tâches de File TA, k = k + 1, aller à I

Sinon appliquer Procédure BS

Sinon stocker les tâches dans File TA, aller à I

Fin (Pour)

II. Si une ressource est libre et A_j arrive **alors**

Si $t_{now} + p_j \le d_j \quad et \quad t_{now} + p_j \le tb_i$ **alors** exécuter la tâche A_j, k = k +1, aller à II

Sinon $A_j \rightarrow$ File TA

Si plusieurs ressources se libèrent et A_j arrive **alors**

Si $t_{now} + p_j \le d_j \quad et \quad t_{now} + p_j \le tb_i$ **alors**

Si BS est respectée pour une ressource **alors** exécuter A_j, k = k + 1, aller à II

Sinon appliquer Procédure BS

Sinon $A_j \rightarrow$ File TA

Si la condition sur BS est satisfaite pour plusieurs ressources **alors** exécuter A_j sur une ressource en utilisant une règle de priorité, k = k + 1, aller à II

Sinon appliquer Procédure BS

Sinon $A_j \rightarrow$ File TA

III. Pour chaque ressource disponible et $ta_i = t_{now}$ **faire**

exécuter chaque tâche P_i , i = i + 1, aller à III

Fin (Pour)

Fin (Tant que)

III.2.2.1 Complexité de l'algorithme proposé

Les deux algorithmes au plus tôt et au plus tard sont équivalents. Ce qui les différencie est l'instant d'exécution de la tâche programmable (au plus tôt et au plus tard) au niveau de la Partie I. La complexité n'est pas affectée.

La complexité de l'algorithme d'ordonnancement temps-réel au plus tôt est O (*n m l*).

III.2.2.2 Exemple 1

Le même exemple (tableaux II.1 et III.1) est à considérer. A rappeler que la priorité dans l'utilisation de la marge disponible est à la tâche programmable. En appliquant l'algorithme d'ordonnancement au plus tôt, nous avons :

t_{now} = 0

Les deux ressources sont disponibles ainsi que la tâche P_1. La tâche A_1 arrive.

Comme $t_{now} = ta_1$, alors exécuter P_1 sur la ressource 1. La ressource 2 est libre (aucune tâche n'est programmée à cet instant), ainsi, ordonnancer A_1 sur la ressource 2 étant donné qu'elle vérifie les conditions $t_{now} + p_j \le d_j$ *et* $t_{now} + p_j \le tb_i$ (0 + 2 < 3 et 0 + 2 < 8).

t_{now} = 5

La ressource 2 se libère et la ressource 1 occupée. La tâche A_2 est en attente d'exécution. La tâche P_4 est disponible. $t_{now} = ta_4$.

Conclusion

t_{now} =13

La ressource 1 se libère. La tâche A_3 est en attente d'exécution.

Vérifier si les conditions

$$t_{now} + p_j \le d_j \quad et \quad t_{now} + p_j \le tb_i$$

sont satisfaites.

$$13 + 3 < 19 \quad \text{et} \quad 13 + 3 < 28 \ (tb_9)$$

Les conditions sont vérifiées.

Conclusion

La tâche A_3 est exécutée sur la ressource 1.

t_{now} =16

Les deux ressources se libèrent et les tâches P_8, P_9 et A_4 sont en attente d'exécution. La tâche P_{10} est disponible.

Vérifier si les conditions

$$t_{now} + p_j \leq d_j \quad et \quad t_{now} + p_j \leq tb_i$$

sont satisfaites.

$16 + 5 < 25$ et $16 + 5 < 30$ (tb_8)

$16 + 5 < 25$ et $16 + 5 < 28$ (tb_9)

Les conditions sont vérifiées.

Conclusion

Comme $t_{now} = 10 = ta_8$ a été dépassé, il est primordial d'exécuter la tâche P_8 à cet instant (t_{now} =16) sur la ressource 2 malgré les conditions vérifiées. De même pour la tâche P_9 qui devait être exécutée au plus tôt à $t_{now} = ta_9 = 14$.

La tâche A_4 est stockée dans File TA et la tâche P_{10} est affectée à File TP de R2.

t_{now} =22

Les deux ressources se libèrent et les tâches P_{11}, P_{12} et A_5 sont en attente d'exécution. la tâche A_6 intervient. $t_{now} = ta_{11} = 20$ et $t_{now} = ta_{12} = 22$.

Conclusion

La tâche P_{11} est exécutée sur la ressource 1 vu que sa date d'exécution au plus tôt a été dépassée. La tâche P_{12} est exécutée sur la ressource 2 (elle doit être exécutée au plus tôt à cet instant). Les tâches A_5 et A_6 sont affectées à File TA.

t_{now} =47

Les deux ressources se libèrent. La tâche A_{10} intervient à cet instant et les tâches A_8 et A_9 sont en attente d'exécution.

Il n'y plus de tâche programmable à exécuter sur la ressource 1.

La tâche A_8 est rejetée car sa date de début au plus tard est $t_{now} = 46$.

Vérifier si les conditions

$$t_{now} + p_j \leq d_j \quad et \quad t_{now} + p_j \leq tb_i$$

sont satisfaites par les tâches A_9 et A_{10} pour la ressource 2.

R2 : A_9 $47 + 2 < 55$ et $47 + 2 < 56$ (tb_{20})

A_{10} $47 + 2 < 55$ et $47 + 2 < 56$ (tb_{20})

Les conditions sont vérifiées.

Conclusion

Les conditions sur les intervalles temporels sont satisfaites par les deux tâches pour les deux ressources. La règle d'affectation de ces tâches aux ressources est la règle FIFO (First In First Out). La tâche A_9 est exécutée sur la ressource 1 et la tâche A_{10} est exécutée sur la ressource 2.

Remarque

En ordonnançant la tâche A_9 sur la ressource 1, aucun changement n'est remarqué. Cependant, l'exécution de la tâche A_{10} sur la ressource 2 engendre un temps mort en attendant que la tâche P_{20} soit disponible pour exécution. Par commodité, la tâche A_{10} est exécutée sur la ressource 1 et la tâche A_9 est exécutée sur la ressource 2.

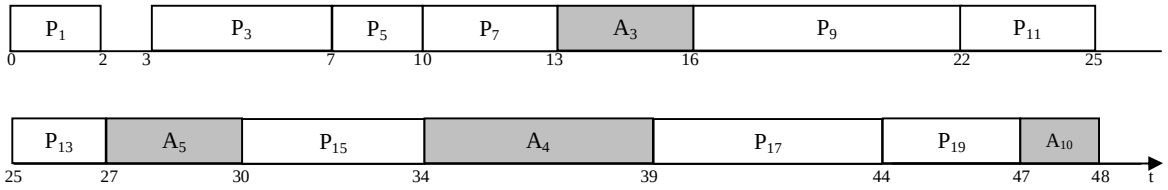
Résultat

Sept tâches aléatoires ont été insérées dans les séquences admissibles des ressources et trois ont été rejetées. BS a été respectée au cours de l'ordonnancement : $BS = C_{\max} = 51$.

La séquence sur R1 est : {P_1, P_3, P_5, P_7, A_3, P_9, P_{11}, P_{13}, A_5, P_{15}, A_7, P_{17}, P_{19}, A_{10}}.

La séquence sur R2 est : {A_1, P_2, P_4, P_6, P_8, P_{10}, P_{12}, A_6, P_{14}, P_{16}, P_{18}, A_9, P_{20}}.

Ressource 1



Ressource 2

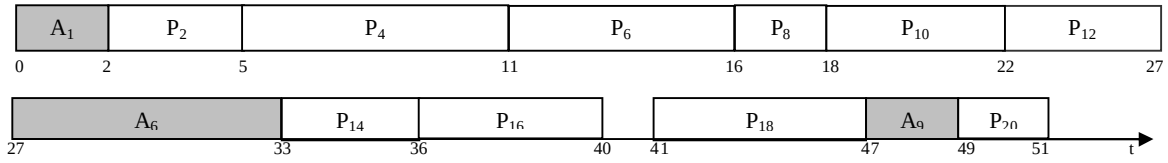


Figure III.3 : Ordonnancement temps-réel au plus tôt.

Conclusion

En appliquant les deux algorithmes d'ordonnancement temps-réel au plus tôt et au plus tard au même exemple, nous constatons que l'algorithme d'ordonnancement temps-réel au plus tard est plus performant que le second du point de vue maximisation du nombre de tâches aléatoires ordonnancées par contre, la durée totale d'ordonnancement est plus grande. Pour les deux algorithmes la borne supérieure a été augmentée.

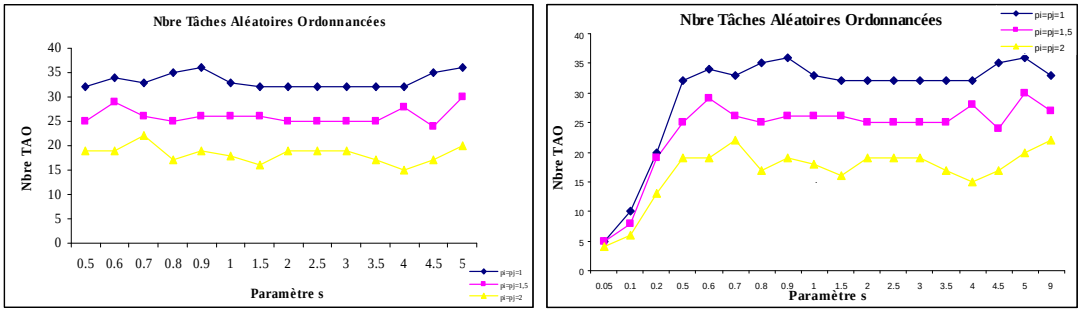
III.2.2.3 Exemple 2

Nous considérons 55 tâches programmables, dont les durées opératoires sont unitaires, et 40 tâches aléatoires apparaissant suivant le processus décrit dans II.3.2. Le paramètre s varie, en premier lieu, entre 0.5 et 5, et, en second lieu, entre 0.05 et 9[24].

Les durées opératoires des tâches sont unitaires et les dates échues $d_j = r_j + Ct$ et Ct = 1.

En appliquant les deux algorithmes d'ordonnancement temps-réel au plus tard et au plus tôt, nous aboutissons au résultat que l'algorithme d'ordonnancement temps-réel au plus tard donne de meilleurs résultats.

- En variant les durées opératoires des tâches: $p_i = p_j \in (1, 1.5, 2)$, Ct = p_j. Nous avons les résultats donnés par la figure III.4.



$s \in (0.5, 5)$ $s \in (0.05, 9)$

Figure III.4 : Nombre de tâches aléatoires ordonnancées.

[24] Voir Annexe III pour les caractéristiques des différentes tâches.

Pour cet exemple, nous avons constaté, figure III.3, le nombre des tâches aléatoires ordonnancées est indépendant de la valeur du paramètre s. Il dépend des marges libres obtenues dans les séquences admissibles sur chaque ressource en ayant ordonnancées toutes les tâches programmables.

Ø Nous considérons le cas où $p_i = p_j$ et le cas $p_i \neq p_j$. Le tableau III.2 illustre les résultats obtenus après application des deux algorithmes d'ordonnancement temps-réel au plus tard et au plus tôt[25].

Cas	Durées	BS	C_{max}	Nombre de tâches aléatoires ordonnancées
1 er cas	$p_i = p_j = 1$	100	100	20
	$p_i = p_j = 1.5$	100.5	100.5	16*
				18**
	$p_i = p_j = 2$	101	101	15*
				17**
2 nd cas	$p_i = 1, p_j \forall$	100	100.5	19*
				20**
	$p_i = 1.5, p_j \forall$	100.5	100.5	15*
				17**
	$p_i = 2, p_j \forall$	101	101	15*
				17**

Tableau III.2: Nombre de tâches aléatoires ordonnancées suivant les durées opératoires.

Nous remarquons que quelque soit la valeur de la durée opératoire des tâches aléatoires, constantes ($p_i = p_j$) ou variables ($p_i \neq p_j$), le nombre de tâches aléatoires ordonnancées est optimisé lors de l'ordonnancement temps-réel au plus tard.

[25] Voir Annexe IV pour les caractéristiques des différentes tâches.
* Algorithme d'ordonnancement temps-réel au plus tôt.
** Algorithme d'ordonnancement temps-réel au plus tard.

Résultats

Le nombre de tâches aléatoires ordonnancées dépend des marges disponibles en ordonnançant toutes les tâches programmables et non des durées opératoires des tâches aléatoires.

Quelques soient les durées opératoires des tâches aléatoires, les tâches imprévues à ordonnancer apparaissant après la dernière tâche programmable ordonnancée revient à un ordonnancement statique de ces tâches (une application d'une simple règle d'affectation aux ressources).

III.2.3 Etude de la performance de la méthode proposée

La performance est étudiée par rapport à la borne de Mac Naughton donnée par :

$$C^*_{\max} = \max\left(\frac{1}{m}\sum_{i=1}^{n} p_i, \max_{1\le i\le n} p_i\right)$$

En considérant l'exemple d'ordonnancement temps-réel précédent (tableaux II.1 et III.1), nous obtenons les valeurs de la borne de Mac Naughton et la règle LPT données par le tableau suivant après application des algorithmes d'ordonnancement temps-réel au plus tard et au plus tôt. Cette étude est effectuée en ramenant les différents ordonnancements temps-réel (au plus tard et au plus tôt) à des ordonnancements statiques[26].

BS	$\mathbf{C_{max}}$	$\mathbf{C^*_{max}}$	$C_{\max_{LPT}}$	**Nbre TAO**
51	53	51.5	52	9
	51	48.5	49	7

Tableau III.3: Présentation des valeurs de la borne de Mac Naughton et la règle LPT.

Nous constatons que, pour cet exemple, la méthode de Mac Naughton et la règle LPT sont plus performantes que les deux algorithmes proposés.

En considérant d'autres instances pour tester la performance des deux algorithmes par rapport à la méthode de Mac Naughton et la règle LPT.

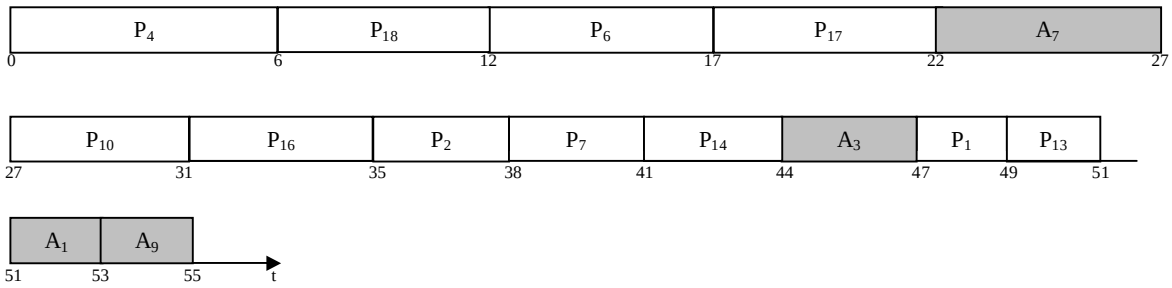
[26] La méthode de Mac Naughton et la règle LPT sont des méthodes optimales dans le cas d'un ordonnancement statique.

Cas	BS	C_{max}	C^*_{max}	$C_{max_{LPT}}$	Nbre TAO
1	47	47*	47*	47	7
		48**	46.5**		6
2	51	51	47	47	5
3	49	54*	51.5*	52*	8
		51**	48**	48**	7
4	51	52	50	50	7
5	51	51	50*	51*	9
			47.5**	48**	8
6	48	53*	50.5*	51*	7
		51**	48.5**	49**	6
7	50	50	47*	47*	6
			48**	48**	
8	51	53*	51*	51*	8
		51**	49.5**	50**	
9	51	53*	51.5*	52*	8
		52**	50**	51**	7
10	51	53*	50	50	8
		51**			7

Tableau III.4: Etude de la performance des algorithmes proposés.

Nous déduisons à partir de ce tableau que les méthodes optimales (la méthode de Mac Naughton et la règle LPT) sont plus performantes que les deux algorithmes en ramenant les ordonnancements dynamiques à des ordonnancements statiques. A préciser que lorsque la valeur de la durée totale d'ordonnancement C_{max} est la même pour les deux algorithmes d'ordonnancement temps-réel, ceci est dû au caractéristiques des différentes tâches aléatoires ordonnancées.

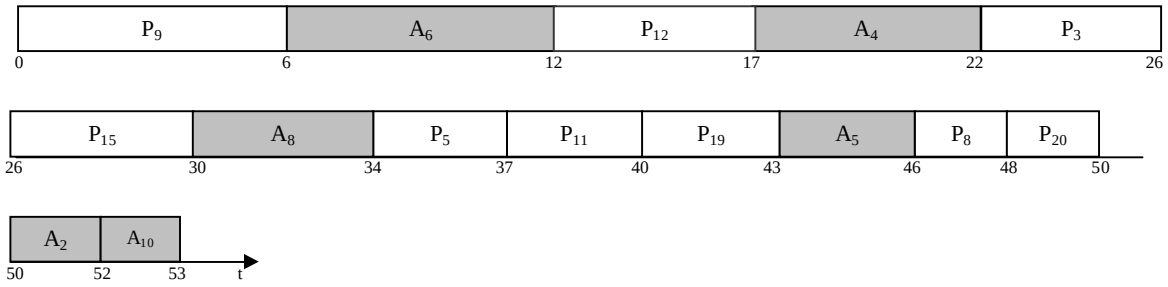
Ressource 1



(a)

* Application de l'algorithme d'ordonnancement au plus tard.
** Application de l'algorithme d'ordonnancement au plus tôt.

Ressource 2



(b)

Figure III.5 : Ordonnancement LPT.

III.3 Conclusion

Dans ce chapitre, deux algorithmes, consacrés à l'ordonnancement temps-réel des tâches sur deux ressources identiques parallèles, ont été présentés en ordonnançant la tâche programmable au plus tôt et au plus tard. A priori, d'après l'analyse des résultats, l'algorithme d'ordonnancement au plus tard, en l'appliquant à l'exemple considéré, est plus performant que l'algorithme d'ordonnancement temps-réel au plus tôt.

La fréquence d'apparition des tâches aléatoires, lors de l'ordonnancement temps-réel, n'affecte aucunement le nombre de tâches ordonnancées. Ce nombre dépend uniquement des marges libres disponibles en ordonnançant toutes les tâches programmables. Par conséquent, les caractéristiques des tâches aléatoires doivent correspondre à ces marges.

Les tâches aléatoires, intervenant après ordonnancement de la dernière tâche programmable ordonnancée, sont affectées aux différentes ressources suivant un ordonnancement statique (une simple règle d'affectation).

Le chapitre suivant traitera un autre cas de problèmes d'ordonnancement sur des ressources identiques en parallèle en temps-réel.

Chapitre IV:

Deuxième Approche : Plusieurs Files D'attente Des Tâches Aléatoires

IV.1 Introduction

Le présent chapitre traite un autre cas de figure des problèmes d'ordonnancement temps-réel sur des ressources identiques en parallèle. Il y a deux ressources identiques en parallèle. Pour chaque ressource, les tâches programmables en attente d'exécution sont stockées dans File TP, qui lui est propre, de même pour les tâches aléatoires qui sont stockées dans File TA dès leur apparition.

La démarche consiste à déterminer une procédure d'affectation des tâches aléatoires, dès leur apparition, aux files d'attente, de ces tâches, respectives des ressources.

Etant donné que l'ordonnancement statique est le même que dans le cas précédent (chapitre précédent), il ne sera considéré, à ce niveau, que l'ordonnancement dynamique.

Ce présent chapitre est réparti en plusieurs sections. La première est une introduction. La résolution du problème considéré est présentée dans la seconde section où deux cas sont à distinguer : le cas d'affectation des tâches aléatoires aux files d'attente leur correspondant en les positionnant directement dès leur apparition et le cas d'affectation des tâches aléatoires aux files un certain temps après leur arrivée. Le problème revient à trouver la meilleure position d'insertion de chaque tâche aléatoire dans le programme d'exécution des tâches programmables.

IV.2 Démarche de résolution du problème à deux files d'attente des tâches aléatoires

Résoudre ce problème revient à ramener le problème initial à deux problèmes à une ressource. Chaque ressource possède une File TP et une File TA.

L'idée est d'affecter la tâche A_j, dès son apparition, à la file d'attente d'une ressource, ainsi, fixer au préalable sa position d'exécution dans la séquence admissible obtenue lors de l'ordonnancement statique.

Cette disposition, une File TP et une File TA pour chaque ressource, peut être vue en tant qu'une ressource à une seule file d'attente englobant l'ensemble des tâches programmables et aléatoires. De ce fait, maintenir l'ordre d'exécution des tâches prévisionnelles de la séquence admissible, de là, décaler les tâches connues a priori dans leurs intervalles admissibles afin d'insérer la tâche qui intervient dans la position qui permet de respecter son délai, les intervalles temporels des tâches prévisionnelles et de façon à préserver au mieux l'avenir (l'ordonnancement des tâches qui apparaîtront plus tard).

IV.2.1 Description du problème

Une ressource doit exécuter un ensemble P de n tâches dites programmables en plus d'un certain nombre A de tâches arrivant aléatoirement.

La définition de certaines données est nécessaire :

t_i : la date de début réel d'exécution prévue pour la tâche i,

tb_i : la date de début au plus tard,

C_i (tf_i) : la date de fin d'exécution de la tâche i,

t_{now} : une variable globale indiquant le temps-réel,

d_f : une variable globale indiquant la date de fin de libération de la ressource, elle est égale à zéro à l'instant t égal à zéro.

Soit $PA = \{P_1, P_2, \ldots, P_n\}$ la séquence admissible représentant la solution du problème d'ordonnancement statique. De là, les dates de début au plus tôt (ta_i) et au plus tard (tb_i) des tâches programmables sont connues. Ces dates vérifient les relations suivantes :

$$\begin{aligned} ta_{i+1} &\geq ta_i + p_i \qquad i = 1, \ldots, n \\ tb_{i+1} &\geq tb_i + p_i \qquad i = 1, \ldots, n \end{aligned} \tag{IV.1}$$

Les contraintes liées aux tâches programmables s'écrivent :

$$\begin{aligned} t_i &\geq ta_i \qquad i = 1, \ldots, n \\ t_i &\leq tb_i \qquad i = 1, \ldots, n \\ ta_i &\neq tb_i \end{aligned} \tag{IV.2}$$

Ces contraintes définissent l'espace (intervalle admissible) dans lequel la tâche i doit être exécutée.

A l'apparition d'une tâche aléatoire A_j, ses caractéristiques sont connues a priori. Quand A_j est insérée, elle est dite tâche aléatoire programmée.

Les contraintes liées aux tâches aléatoires sont :

$$\begin{aligned} t_j + p_j &\leq d_j \\ t_j + p_j &\leq tb_i \end{aligned} \tag{IV.3}$$

sachant que A_j immédiatement prédécesseur à P_i.

Le critère à optimiser est la durée totale d'ordonnancement.

Deux cas sont à distinguer :

- Affecter les tâches aléatoires aux File TA des ressources en les positionnant directement.
- Affecter les tâches aléatoires aux File TA des ressources et trouver la meilleure position d'insertion de chaque tâche aléatoire dans le programme d'exécution des tâches programmables.

Remarque

Nous considérons le cas de deux ressources identiques en parallèle. Les deux algorithmes présentés, dans ce travail, sont généralisés au cas d'un nombre m de ressources identiques parallèles. Ceci ne modifie en rien l'étude de la complexité des algorithmes proposés, il suffit de considérer le nombre de ressources requis pour l'étude.

IV.2.2 Premier cas

Dans ce cas, l'algorithme à appliquer est l'algorithme d'ordonnancement temps-réel au plus tard. Il permet d'affecter les tâches aléatoires aux files d'attente des ressources en les insérant directement dans la position d'exécution. Par conséquent, à chaque apparition d'une tâche aléatoire, elle doit satisfaire des conditions pour pouvoir l'exécuter. Dès son affectation, elle a une position entre les tâches programmables.

IV.2.2.1 Algorithme d'ordonnancement temps-réel au plus tard

Cet algorithme donne la priorité à la tâche aléatoire, lors de l'ordonnancement dès son arrivée ou chaque fois qu'une ressource est libre, sans induire de retard sur l'exécution de la tâche programmable.

Initialisation : i = 1, k = 0

Tant que t_{now} est inférieur à d_n **faire**

Si une tâche A_j intervient et toutes les ressources occupées **alors** rejeter A_j

I. Pour chaque libération d'une ressource **faire**

Si une tâche A_j peut être exécutée satisfaisant les conditions $t_{now} + p_j \le d_j$ *et* $t_{now} + p_j \le tb_i$ pour une ressource **alors** exécuter A_j , k = k + 1, aller à I

Sinon rejeter A_j, aller à I

Si la tâche A_j vérifie les conditions $t_{now} + p_j \le d_j$ *et* $t_{now} + p_j \le tb_i$ pour plusieurs ressources **alors**

Si BS est respectée pour une ressource **alors** exécuter A_j,k = k + 1, aller à I

Sinon appliquer Procédure BS

Si la condition sur BS est vérifiée pour plusieurs ressources **alors** exécuter A_j sur une ressource selon une règle de priorité,k = k +1, aller à I

Sinon appliquer Procédure BS

Sinon rejeter A_j , aller à I

Si plusieurs tâches vérifient les conditions $t_{now} + p_j \le d_j \quad et \quad t_{now} + p_j \le tb_i$ **alors**

Si les conditions sont vérifiées pour une ressource **alors**

Si une tâche vérifie la condition sur BS **alors** exécuter A_j, k = k +1, aller à I

Sinon appliquer Procédure BS

Sinon rejeter A_j aller à I

Si plusieurs tâches vérifient la condition sur BS **alors** exécuter la tâche la plus prioritaire, rejeter les autres tâches aléatoires, k = k + 1, aller à I

Sinon appliquer Procédure BS

Sinon rejeter A_j , aller à I

Si les conditions sont vérifiées pour plusieurs ressources **alors**

Si une tâche A_j vérifie la condition sur BS pour plusieurs ressources **alors** exécuter A_j sur une ressource, k = k + 1, aller à I

Sinon appliquer Procédure BS

Si plusieurs tâches vérifient la condition sur BS **alors** exécuter les tâches sur les ressources selon une règle de priorité, rejeter les autres tâches, k = k + 1, aller à I

Sinon appliquer Procédure BS

Sinon rejeter les tâches, aller à I

Sinon

Si $ta_i \ge t_{now}$ **alors** exécuter P_i , i = i + 1, rejeter A_j, aller à I

Sinon rejeter les tâches aléatoires et stocker les tâches programmables dans leurs File TP respectives, aller à I

Fin (Pour)

II. Si une ressource est libre et A_j arrive **alors**

Si $t_{now} + p_j \le d_j$ *et* $t_{now} + p_j \le tb_i$ **alors** exécuter la tâche A_j , k = k +1, aller à II

Sinon rejeter A_j

Sinon rejeter A_j

Si plusieurs ressources se libèrent et A_j arrive **alors**

Si $t_{now} + p_j \le d_j$ *et* $t_{now} + p_j \le tb_i$ **alors**

Si BS est respectée pour une ressource **alors** exécuter A_j, k = k + 1, aller à II

Sinon appliquer Procédure BS

Sinon rejeter A_j

Si la condition sur BS est satisfaite pour plusieurs ressources **alors** exécuter A_j sur une ressource en utilisant une règle de priorité, k = k + 1, aller à II

Sinon appliquer Procédure BS

Sinon rejeter A_j

Sinon rejeter A_j

III. Pour chaque ressource disponible et $ta_i = t_{now}$ **faire**

exécuter chaque tâche P_i , i = i + 1, aller à III

Fin (Pour)

Fin (Tant que)

IV.2.2.2 Complexité de l'algorithme proposé

L'algorithme d'ordonnancement au plus tard est constitué d'une boucle principale à trois parties indépendantes.

La partie I

Cette partie ne contient qu'une seule boucle (**Pour** chaque …) à plusieurs instructions conditionnelles. Ces instructions seront exécutées, dans le pire des cas, une seule fois. Cette exécution est liée à l'apparition des tâches aléatoires.

La boucle (**Pour** chaque…) est exécutée, dans le pire des cas, *l* fois tel que *l* est le nombre total de tâches aléatoires apparues au cours de l'ordonnancement. La complexité de cette condition est *O (l)*.

La partie II

Cette partie est constituée de plusieurs instructions conditionnelles.

Les instructions (**Si** une ressource …) et (**Si** plusieurs ressources …) sont exécutées, dans le pire des cas, une fois.

L'instruction **Affecter A_i à FTA** est, aussi, exécutée dans le pire des cas une fois (une seule tâche qui arrive).

La complexité de cette partie est égale à *O (m)*.

La partie III

Elle est constituée d'une boucle **Pour**. Cette boucle dépend du nombre de ressources disponibles et du nombre de tâches prêtes à être exécutées (*n* dans le pire des cas).

La complexité de la boucle **Pour** est *O (n m)*.

Comme la boucle **Tant Que** est constituée des trois parties (I, II et III), la complexité de l'algorithme est *O (n m l)*.

IV.2.2.3 Exemple

Considérant le même exemple présenté précédemment dont les caractéristiques des tâches programmables et aléatoires sont présentées par les tableaux suivant :

i	1	2	3	4	5	6	7	8	9	10	11	12
r_i	0	1	3	5	6	7	9	10	14	16	19	20
p_i	2	3	4	6	3	5	3	2	6	4	3	5
d_i	6	8	32	20	19	25	20	30	28	35	37	38

i	13	14	15	16	17	18	19	20
r_i	25	28	30	33	38	41	44	49
p_i	2	3	4	4	5	6	3	2
d_i	43	50	45	43	46	49	55	58

Tableau IV.1 : Caractéristiques des tâches programmables.

j	1	2	3	4	5	6	7	8	9	10
r_j	0	3	11	15	18	22	31	39	43	47
p_j	2	2	3	5	3	6	5	4	2	1
d_j	3	9	19	25	30	40	44	50	55	52

Tableau III.1 : Caractéristiques des tâches aléatoires.

Dans cette étape, l'algorithme d'ordonnancement temps-réel au plus tard est à appliquer.

$t_{now} = 0$

Les deux ressources sont disponibles ainsi que P_1. La tâche A_1 arrive.

Vérifier les conditions

$$t_{now} + p_j \leq d_j \quad et \quad t_{now} + p_j \leq tb_i$$

Pour R1 : $0 + 2 < 3$ et $0 + 2 < 6$

Pour R2 : $0 + 2 < 3$ et $0 + 2 < 8$

Les conditions sur les intervalles temporels sont vérifiées pour les deux ressources.

Il faut tester la condition sur BS.

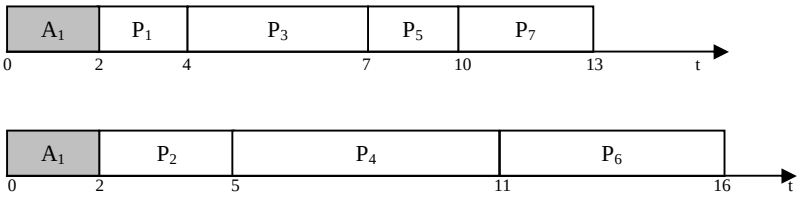


Figure IV.1 : Ordonnancement de la tâche A_1.

Les conditions sur BS sont satisfaites pour les deux ressources.

Affecter la tâche A_1 à une ressource suivant une règle de priorité.

Conclusion

Etant donné que la tâche P_1 est disponible à cet instant et aucune tâche n'est en attente d'exécution sur la ressource 2, la tâche A_1 sera exécutée, à cet instant, sur la ressource 2.

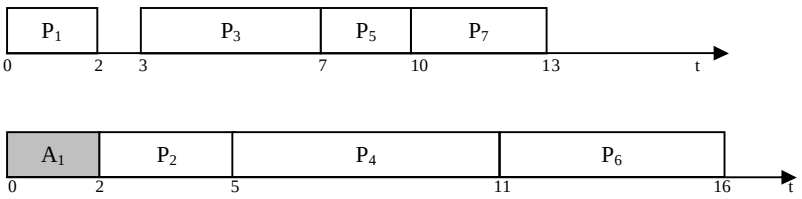


Figure IV.2 : Ordonnancement de la tâche A_1.

$t_{now} = 2$

La tâche P_2 est disponible et les ressources sont libres. Aucune tâche aléatoire n'intervient à cet instant.

Conclusion

La tâche P_2 est exécutée sur la ressource 2.

t_{now} = 3

La ressource 2 est occupée à cet instant. La ressource 1 est libre et la tâche P_3 est disponible. La tâche A_2 intervient.

Vérifier si les conditions :

$$t_{now} + p_j \leq d_j \; et \; t_{now} + p_j \leq tb_i$$

sont satisfaites.

$$3 + 2 < 9 \quad \text{et} \quad 3 + 2 < 10$$

Les conditions sont vérifiées.

Conclusion

La tâche A_2 est ordonnancée sur la ressource 1. La tâche P_3 est stockée dans File TP.

t_{now} = 5

A cet instant, les deux ressources se libèrent, la tâche P_4 est disponible et la tâche P_3 est en attente d'exécution. Aucune tâche aléatoire n'intervient

Conclusion

Exécuter la tâche P_3 sur R_1 et la tâche P_4 sur R_2.

t_{now} = 9

Les tâches P_3 et P_4 sont disponibles et les ressources sont libres. Aucune tâche aléatoire n'intervient à cet instant.

Conclusion

Les tâches P_3 et P_4 sont exécutées respectivement sur la ressource 1 et la ressource 2.

t_{now} = 39

Les ressources sont occupées et la tâche A_8 intervient. La tâche A_7 est en attente d'exécution.

Conclusion

Stocker la tâche A_8 dans File TA. La tâche A_7 est rejetée car elle doit être exécutée au plus tard à cet instant (t_{now} = 39).

t_{now} = 46

La ressource 1 se libère. Les tâches A_8, A_9 et P_{19} sont en attente d'exécution.

Vérifier si les conditions

$$t_{now} + p_j \le d_j \quad et \quad t_{now} + p_j \le tb_i$$

sont satisfaites.

A_8 : 46 + 4 = 50 et 46 + 4 < 52 (tb_{19})

A_9 : 46 + 2 < 55 et 46 + 2 < 52 (tb_{19})

Les conditions sont vérifiées.

Conclusion

Du fait que les conditions sont satisfaites par les deux tâches pour la même ressource, leur affectation nécessite une règle de priorité. La règle de priorité choisie (parmi d'autres) est la règle FIFO (First In First Out). Par conséquent, exécuter la tâche A_8 sur la ressource 1 et affecter la tâche A_9 à File TA. Stocker la tâche P_{19} dans File TP de la ressource 1.

t_{now} = 50

Les deux ressources sont libres et les tâches A_{10}, P_{19} et P_{20} sont en attente d'exécution.

Vérifier si les conditions

$$t_{now} + p_j \le d_j \quad et \quad t_{now} + p_j \le tb_i$$

sont satisfaites.

A_{10} : 50 + 1 < 52 et 50 + 1 < 52 (tb_{19})

50 + 1 < 52 et 50 + 1 < 56 (tb_{20})

Conclusion

En ordonnançant la tâche A_{10} sur le ressource 1, la borne supérieure sera augmentée de trois unités de temps ($BS = 54$), alors, qu'elle ne sera augmentée que de deux unités de temps ($BS = 53$) si elle est ordonnancée sur la ressource 2. Par conséquent, exécuter la tâche P_{19} sur la ressource 1 et la tâche A_{10} sur la ressource 2. Stocker la ressource P_{20} dans File TP de la ressource 2. Elle est considérée à partir de instant comme suit : $BS = C_{\max} = 53$.

t_{now} = 51

La ressource 2 se libère et la tâche P_{20} est en attente d'exécution.

Conclusion

Exécuter la tâche P_{20} sur la ressource 2.

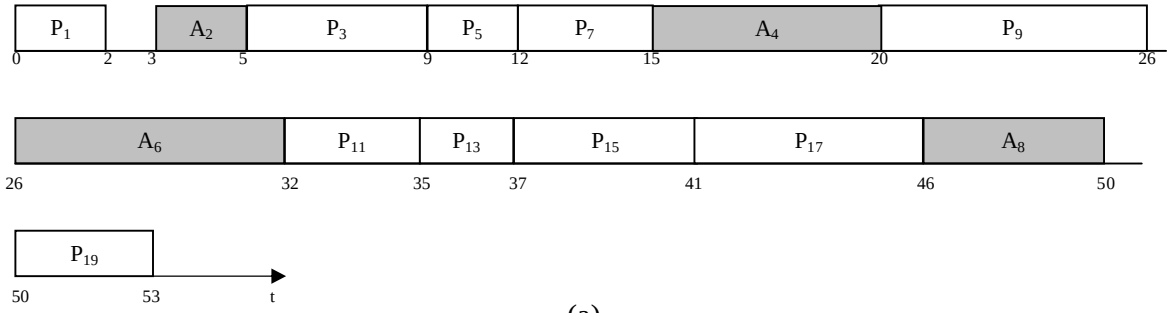
Résultat

Neuf tâches aléatoires ont été insérées dans les séquences admissibles des ressources et une seule a été rejetée. BS n'a été augmentée qu'une seule fois de deux unités de temps.

La séquence sur R1 est : {P_1, A_2, P_3, P_5, P_7, A_4, P_9, A_6, P_{11}, P_{13}, P_{15}, P_{17}, A_8, P_{19}}.

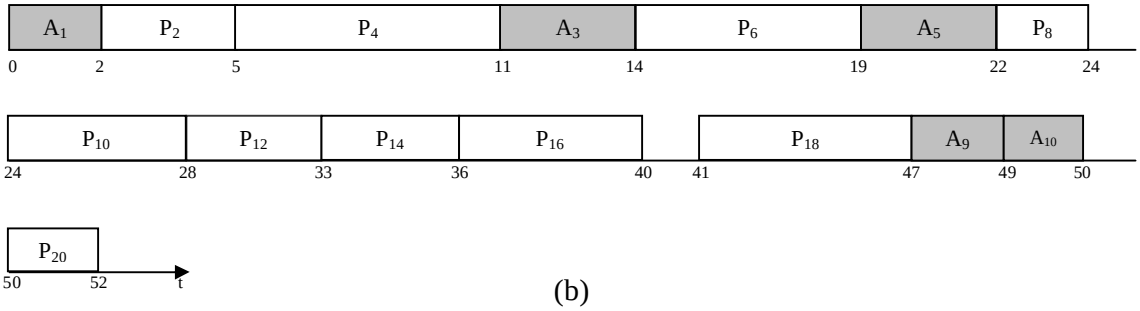
La séquence sur R2 est : {A_1, P_2, P_4, A_3, P_6, A_5, P_8, P_{10}, P_{12}, P_{14}, P_{16}, P_{18}, A_9, A_{10}, P_{20}}.

Ressource 1



(a)

Ressource 2



(b)

Figure IV.3 : Ordonnancement temps-réel au plus tard.

IV.2.3 Deuxième cas

Pour définir l'approche de résolution, la démarche est constituée principalement de deux étapes. La première étape détermine le premier espace libre entre deux tâches programmables P_k et P_{k+1}, pour insérer la tâche aléatoire intervenant. De là, une première position provisoire pour A_j est obtenue. La deuxième étape définit la position d'insertion de la tâche qui permet de maximiser sa chance d'acceptation et qui préserve au mieux l'avenir. L'idée, dans ce cas, consiste à avancer éventuellement certaines tâches programmées vers la droite pour insérer A_j plus tôt que sa position provisoire.

IV.2.3.1 Définition de la position provisoire d'une tâche aléatoire

A l'instant initial, la date de début d'exécution prévue pour chaque tâche programmable correspond à sa date de début au plus tôt

$$(t_i = ta_i \qquad i = 1 \text{ i } n)$$

De ce fait, aucune tâche aléatoire n'est prévue pour exécution.

Soit S, l'ordonnancement semi actif des tâches programmables à leurs dates de début au plus tôt, et, S' l'ordonnancement de ces tâches à leurs dates de début au plus tard (les tâches sont décalées vers la droite dans S jusqu'à leurs limites supérieures).

Deux cas sont à considérer pour caractériser la position provisoire d'une tâche aléatoire :

1. Aucune tâche aléatoire n'est prévue pour exécution.
2. Une ou plusieurs tâches aléatoires sont programmées.

IV.2.3.1.a Aucune tâche aléatoire n'est prévue pour exécution

A l'instant d'arrivée de la tâche à insérer aucune tâche aléatoire n'est prévue dans l'ordonnancement des tâches programmables et la première tâche prévisionnelle ordonnancée est P_l (la prochaine tâche à exécuter).

Le problème consiste, alors, à placer A_j dans S de manière à exécuter les tâches programmables dans leurs intervalles admissibles.

Trois positions possibles pour la tâche A_j :

- placer A_j avant P_l,
- placer A_j entre P_i *et* P_{i+1} $\quad i \in \{l, \text{¡}, n\}$,
- il n'existe pas de place pour A_j.

Pour chaque tâche $P_i, i = l \text{ ¡ } n,$ est définie une marge libre s_i donnée par :

$$s_i = \begin{cases} tb_i - \max(t_{now}, d_f) & i = l \\ tb_i - (t_{i-1} + p_{i-1}) & i \neq l \end{cases} \qquad \text{(IV.4)}$$

Pour pouvoir insérer une tâche A_j dans l'ordonnancement, la condition nécessaire qui est appelée condition d'espace doit être vérifiée :

$$\exists s_i / s_i \geq p_j \quad i = l, l+1, \text{¡} \qquad \text{(IV.5)}$$

En plus de la contrainte sur le délais de A_j :

$$t_j + p_j \leq d_j \qquad \text{(IV.6)}$$

IV.2.3.1.b Une (plusieurs) tâche (s) aléatoire (s) est (sont) prévue (s) pour exécution

A l'instant t_{now}, une ou plusieurs tâches aléatoires programmées existent pour exécution.

L'ordonnancement des tâches programmables P_k et P_{k+1}, respectivement au plus tôt et au plus tard, a une conséquence directe sur l'insertion de la tâche A_j.

Pour insérer des tâches dans des espaces libres, il est préférable d'avoir des marges larges.

IV.2.3.1.b.1 Définition des marges

Soient:

$A(k) = \{A_j \in S / A_j$ ordonnancée avant $P_k\}$

$A(i) = \{A_j \in S / A_j$ calée entre P_i et $P_{i+1}\}$, i = k-1,...,n

$A(n+1) = \{A_j \in S / A_j$ ordonnancée après $P_n\}$

Les marges libres s'expriment par :

$$v_i = \begin{cases} tt_i - \max(t_{now}, d_f) - \sum_{j \in A(k)} p_j & i = k \\ tt_i - (t_{i-1} + p_{i-1}) - \sum_{j \in A(i)} p_j & i = k+1, \dots, n \end{cases} \quad \text{(IV.7)}$$

avec

$$tt_i = \begin{cases} \min\left(tb_n, t_n + \min_{j \in A(n+1)}\left(d_j - p_j\right)\right) & i = n \\ \min\left(tb_i, t_i + \min_{j \in A(i+1)}\left(d_j - p_j\right), tt_{i+1} - p_i - \sum_{j \in A(i+1)} p_j\right) & i = n-1, \dots, k \end{cases} \quad \text{(IV.8)}$$

et

$$t_i = \begin{cases} \max\left(ta_i, \max(d_f, t_{now}) + \sum_{j \in A(k)} p_j\right) & i = k \\ \max\left(ta_i, t_{i-1} + p_{i-1} + \sum_{j \in A(i)} p_j\right) & i = k+1 \dots n \end{cases} \quad \text{(IV.9)}$$

La figure IV.3 présente les marges dans le cas d'existence de tâches aléatoires programmées.

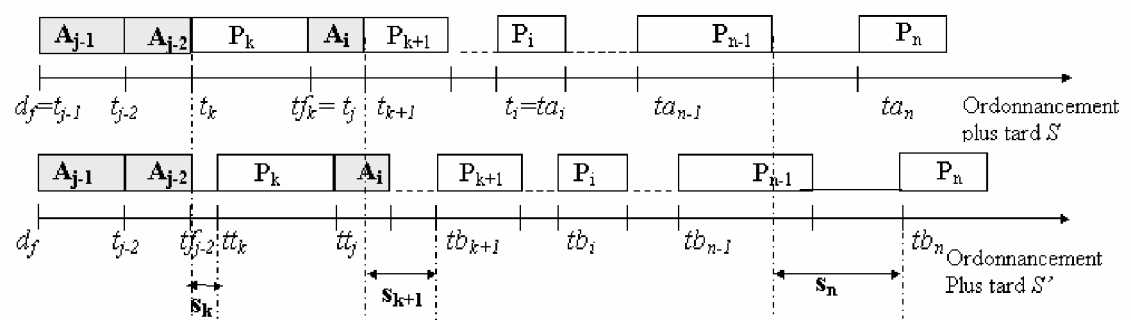


Figure IV.3: Les marges libres [Our 01].

IV.2.3.1.b.2 Algorithme de calcul de la marge disponible

A l'instant courant, vérifier les contraintes sur l'intervalle temporel de la tâche aléatoire $t_{now} + p_j \le d_j$ et sur la marge libre permettant d'insérer cette tâche $s_i \ge p_j$ (si aucune tâche aléatoire n'est programmée). Lorsqu'une ou plusieurs tâches aléatoires sont programmées pour exécution, la contrainte sur la marge à vérifier est $v_i \ge p_j$.

A chaque apparition d'une tâche aléatoire, l'algorithme suivant est à appliquer.

Initialisation : i = 1, k = 0

Tant que t_{now} inférieur à d_n **faire**

Pour chaque apparition d'une tâche aléatoire **faire**

Si aucune tâche aléatoire n'est programmée **faire**

Si la tâche A_j vérifie les conditions $t_{now} + p_j \le d_j$ *et* $s_i \ge p_j$ pour une ressource **alors** la tâche A_j est affectée à File TA, k = k + 1

Sinon rejeter A_j

Si une tâche A_j vérifie les conditions $t_{now} + p_j \le d_j$ *et* $s_i \ge p_j$ pour plusieurs ressources **alors**

Si la condition sur BS est satisfaite pour une ressource **alors** la tâche A_j est affectée à File TA, k = k + 1

Sinon appliquer Procédure BS

Si la condition sur BS est satisfaite pour plusieurs ressources **alors** la tâche A_j est affectée à une File TA suivant une règle de priorité, k = k + 1

Sinon appliquer Procédure BS

Sinon la tâche A_j est rejetée

Sinon

Si la tâche A_j vérifie les conditions $t_{now} + p_j \le d_j \quad et \quad v_i \ge p_j$ pour une ressource **alors** la tâche A_j est affectée à File TA, k = k + 1

Sinon $A_j \rightarrow$ File TA

Si une tâche A_j vérifie les conditions $t_{now} + p_j \le d_j \quad et \quad v_i \ge p_j$ pour plusieurs ressources **alors**

Si la condition sur BS est satisfaite pour une ressource **alors** la tâche A_j est affectée à File TA, k = k + 1

Sinon appliquer Procédure BS

Si la condition sur BS est satisfaite pour plusieurs ressources **alors** la tâche A_j est affectée à une File TA suivant une règle de priorité,

k = k + 1

Sinon appliquer Procédure BS

Sinon la tâche A_j est rejetée

Fin (Pour)

Fin (Tant que)

IV.2.3.1.b.3 Complexité de l'algorithme proposé

L'algorithme est constituée d'une boucle principale (**Tant Que**) imbriquée avec une autre (**Pour** chaque …).

La boucle (**Pour** chaque …) est exécutée, dans le pire des cas, l fois tel que *l* est le nombre total de tâches aléatoires apparues au cours de l'ordonnancement. La complexité de cette boucle est *O (l)*.

L'affectation de la tâche A_j à la file d'attente d'une ressource est, aussi, exécutée dans le pire des cas une fois (une seule tâche aléatoire disponible devant être ordonnancée sur une ressource). La complexité de cette partie est égale à *O (m)*.

La boucle (**Pour** chaque …) est de complexité *O (m l)*.

La boucle **Tant Que** est vérifiée, dans le pire des cas, n fois (n étant le nombre de tâches programmables). Sa complexité est *O (n)*. Comme cette boucle est imbriquée avec la boucle (**Pour** chaque …), la complexité de l'algorithme est *O (n m l)*.

IV.2.3.1.b.4 Exemple

En considérant le même exemple des tableaux II.1 et III.1.

t_{now} = 0

Les deux ressources sont disponibles ainsi que P_1. La tâche A_1 arrive. Aucune tâche aléatoire n'est programmée.

Vérifier si les conditions

$$t_{now} + p_j \leq d_j \quad et \quad s_i \geq p_j$$

sont satisfaites.

Pour R1 : $0 + 2 < 3$ et $5 > 2$

Pour R2 : $0 + 2 < 3$ et $7 > 2$

Les conditions sur les intervalles temporels sont vérifiées pour les deux ressources.

Il faut tester la condition sur BS.

En ordonnançant A_1 sur les deux ressources, la borne est respectée. Pour la ressource 1, la tâche P_1 est décalée, en ordonnançant A_1. Par contre, aucune tâche n'est décalée concernant la ressource 2.

Conclusion

Il est plus intéressant d'exécuter la tâche A_1 sur la ressource 2 puisque aucune tâche prévisionnelle n'est programmée, alors que, sur la ressource 1 la tâche P_1 est en attente d'exécution.

La tâche A_1 est affectée à File TA de la ressource 2.

t_{now} = 3

A cet instant la tâche A_2 intervient. La ressource 1 est libre et la ressource 2 est occupée. Aucune tâche aléatoire n'est ordonnancée sur cette ressource

Vérifier les conditions sur les intervalles temporels

$$t_{now} + p_j \leq d_j \quad et \quad s_i \geq p_j \quad ?$$

$$3 + 2 < 9 \quad \text{et} \quad 7 > 2$$

Les conditions sont satisfaites pour la ressource 1.

Conclusion

La tâche A_2 sera exécutée sur la ressource 1. Elle est affectée à File TA de cette ressource.

t_{now} = 11

A cet instant, la tâche A_3 arrive et la ressource 2 se libère.

Vérifier pour cette ressource

$$t_{now} + p_j \le d_j \quad et \quad v_i \ge p_j \quad ?$$

$$11 + 3 \quad < 19 \quad \text{et} \quad 1 < 3$$

A cet instant, pour la ressource 2, il n'y a pas de marge libre pour insérer la tâche A_3. A cet instant la tâche P_4 est exécutée sur la ressource 2.

Il faut trouver une position, à un autre instant, pour la tâche A_3 sur la ressource 1 ou la ressource 2.

Considérer $t_{now} = 16$ pour la ressource 2.

Vérifier

$$t_{now} + p_j \le d_j \quad et \quad v_i \ge p_j \quad ?$$

$$16 + 3 \quad = 19 \quad \text{et} \quad v_8 < p_3$$

Il n'existe pas de position pour A_3, à cet instant sur R2.

Considérer $t_{now} = 12$ pour la ressource 1

$$12 + 3 \quad < 19 \quad \text{et} \quad v_7 < p_3$$

Considérer $t_{now} = 15$

$$15 + 3 \quad < 19 \quad \text{et} \quad v_9 < p_3$$

Conclusion

La tâche A_3 est rejetée (d_j=19). Exécuter à $t_{now} = 11$ la tâche P_6 sur R2et à $t_{now} = 12$ la tâche P_7 sur R2.

$t_{now} = 18$

La tâche A_5 apparaît à cet instant.

Vérifier

$$t_{now} + p_j \le d_j \quad et \quad v_i \ge p_j \quad ?$$

$$18 + 3 \quad < 30 \quad \text{et} \quad v_{10} < p_5$$

$$22 + 3 \quad < 30 \quad \text{et} \quad v_{10} = p_5$$

Les conditions sont vérifiées pour la ressource 2 à l'instant $t_{now} = 22$.

Conclusion

La tâche A_5 sera exécutée sur la ressource 2 à $t_{now} = 22$. Elle est affectée à File TA de cette ressource.

Exécuter la tâche P10 à $t_{now} = 18$.

$t_{now} = 39$

La tâche A_8 est disponible à cet instant. La ressource 1 se libère et la ressource2 est occupée.

Vérifier

$$t_{now} + p_j \leq d_j \quad et \quad v_i \geq p_j \quad ?$$

$$39 + 4 < 50 \quad \text{et} \quad v_{19} > p_8$$

Les conditions sont satisfaites.

Conclusion

La tâche A_8 sera exécutée sur la ressource 1 à cet instant. Elle est affectée à File TA de cette ressource.

Les séquences d'ordonnancement sur les deux ressources sont :

Sur R1 : {P_1, A_2, P_3, P_5, A_4, P_7, P_9, P_{11}, P_{13}, P_{15}, P_{17}, A_8, A_9, P_{19}}.

Sur R2 : {A_1, P_2, P_4, P_6, P_8, P_{10}, A_5, P_{12}, A_6, P_{14}, P_{16}, P_{18}, A_{10}, P_{20}}.

Résultat

Huit tâches aléatoires ont été insérées dans le programme d'ordonnancement des tâches programmables.

La borne a été augmentée à la fin de l'ordonnancement de toutes les tâches (programmables et aléatoires).

Par application des deux algorithmes d'ordonnancement temps-réel présentés respectivement dans le premier et le deuxième cas, nous déduisons que, pour cet exemple, le premier algorithme donne de meilleurs résultats.

IV.2.3.2 Position d'insertion de la tâche aléatoire qui maximise sa chance d'acceptation

A ce niveau, considérer une ressource et étudier la meilleure position d'insertion d'une tâche aléatoire affectée à cette ressource par rapport aux autres tâches aléatoires.

Partant du fait que la position provisoire prévoit l'insertion de la tâche aléatoire A_j entre P_i et P_{i+1}, deux cas sont à considérer :

- Ø Des tâches aléatoires programmées entre P_i et P_{i+1}.
- Ø Des tâches aléatoires programmées avant P_i.

IV.2.3.2.1 Premier cas

Soit A_l la dernière tâche aléatoire programmée entre P_i et P_{i+1}, A_j est positionnée provisoirement après A_i (fig. IV.6).

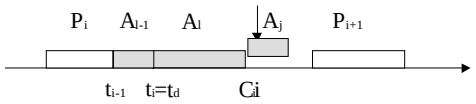


Figure IV.6: Position provisoire de A_j.

L'ordonnancement des tâches aléatoires 'rangées entre deux tâches programmables' dans l'ordre croissant de leurs dates échues (EDD) maximise la chance d'acceptation de la tâche à intercaler dans le temps.

Preuve

Considérons le cas où d_j ® d_l (le raisonnement est le même dans le cas contraire)

Il est possible de placer j après l (j et l rangées selon l'ordre EDD) ou bien avant l (fig.IV.6). Nous analysons ces deux situations pour montrer que le rangement des tâches selon EDD maximise la chance d'acceptation de A_j.

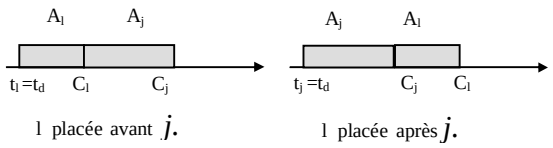


Figure IV.7 : Ordre des tâches A_j et A_l.

1. Placer j après l

Condition sur le délai de j :

$$t_d + p_l + p_j \le d_j \qquad (1.1)$$

2. Placer j avant l

Condition sur le délai de j est :

$$t_d + p_j \le d_j \qquad (2.1)$$

Condition sur le délai de l est :

$$t_d + p_j + p_l \le d_l \qquad (2.2)$$

En comparant les deux situations, placer *j* après *l* et placer *j* avant *l* sachant que $d_j \leq d_l$, nous concluons que, si *j* ne peut être ordonnancée avant *l*, elle ne peut être ordonnancée après l, ceci s'explique par :

- Si la condition (1.1) est fausse, la condition (2.2) peut être vraie car $d_l \geq d_j$
- Si la condition (2.2) est fausse, la condition (1.1) est aussi fausse car d_j ® d_l

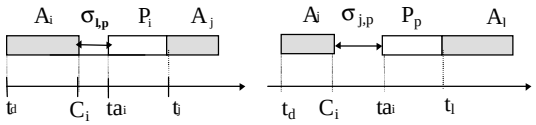
Supposons que (2.2) soit fausse, et que $d_l < d_{l-1}$. D'après le raisonnement qui vient d'être présenté la séquence (A_{l-1}, A_l, A_j) peut permettre l'exécution de A_{l-1} dans le délai. Si la séquence (A_j, A_l, A_{l-1}) ne permet pas l'exécution des tâches A_l et A_{l-1} dans leurs délais, alors il est certain que la tâche A_j ne puisse être insérée.

Résultat

Une condition nécessaire mais non suffisante pour insérer A_j est que son rangement avec les autres tâches aléatoires selon l'ordre EDD permet aux tâches qui lui succéderont (si elles existent) d'être exécutées dans leurs délais. Dans le cas contraire A_j est rejetée.

IV.2.3.2.2 Deuxième cas

Soit A_j la tâche aléatoire programmée avant P_i, A_j est positionnée provisoirement après P_i. Nous analysons alors deux situations (fig.IV.7) afin de définir la position d'insertion éventuelle de la tâche A_j qui maximise sa chance d'acceptation.



Situation 1:l placée avant j Situation 2: l placée après j

Figure IV.8 : Ordre des tâches A_j et A_l.

Soit $\sigma_{u,i}$ une variable réelle définissant l'écart entre une tâche aléatoire programmée *u* et la tâche programmable i immédiatement successeur à *u*, elle s'écrit :

$$\sigma_{u,i} = \begin{cases} ta_i - C_u & si \quad ta_i - C_u \text{ ¤} 0 \\ 0 & si \quad ta_i - C_u \leq 0 \; et \; tb_i - C_u \geq 0 \\ tb_i - C_u & si \quad tb_i - C_u \text{ ®} 0 \end{cases} \qquad \text{(IV.10)}$$

Nous considérons le cas où d_j ® d_i (le raisonnement est le même dans le cas contraire)

- Dans la situation 1, $\sigma_{l,i} \geq 0$ car A_l est ordonnancée avant P_i, et $\sigma_{j,i+1} \geq 0$ du fait que A_j est positionnée provisoirement après P_i.
- Dans la situation 2, $\sigma_{j,i}$ *et* $\sigma_{l,i+1}$ peuvent être positifs ou négatifs ou nuls. Nous considérons dans cette étude que $\sigma_{j,i}$ *et* $\sigma_{l,i+1} \geq 0$.

Comparons alors les deux situations, placer j après l, et placer j avant l sachant que $d_j \leq d_l$.

1. **Placer j après l**

Condition sur le délai de j est :

$$C_j = t_d + p_j + \sigma_{l,i} + p_i + p_j \leq d_j \qquad (1.1)$$

2. **Placer j avant l**

Condition sur le délai de j est :

$$C_j = t_d + p_j \leq d_j \qquad (2.1)$$

Condition sur le délai de i est :

$$t_d + p_j + \sigma_{j,i} + p_i + p_l \leq d_l \qquad (2.2)$$

Poser : $A = t_d + p_j + p_l + p_i$, d'où : $A \leq d_j - \sigma_{l,i}$ (1.1) et $A \leq d_l - \sigma_{j,i}$ (2.2)

a. $\sigma_{l,i}$ ¤ $0, \sigma_{j,i} = 0$

Si la condition (1.1) est fausse alors la condition (2.2) peut être vraie car $d_l \geq d_j - \sigma_{l,i}$

Si la condition (2.2) est fausse alors la condition (1.1) est aussi fausse car $d_j - \sigma_{l,i}$ ® d_l

Résultat

Si le rangement des tâches A_l et A_j dans l'ordre EDD permet d'ordonnancer A_l en respectant son délai

alors la tâche A_j est rangée provisoirement dans la position définie par cet ordre

Sinon A_j est rejetée.

b. $\sigma_{l,i} = 0, \sigma_{j,i} \geq 0$.

Etudier sous quelles conditions (2.2) est fausse alors que (1.1) est vraie.

$$\left.\begin{array}{l}(2.2)\ \textit{fausse}: A \text{¤} d_l - \sigma_{j,i} \\ (1.1)\ \textit{vraie}: A \le d_j\end{array}\right\} \Rightarrow d_l - \sigma_{j,i} \text{®} d_j$$

$$\Rightarrow \quad d_l - d_j \text{®} \sigma_{j,i}$$

L'inéquation $d_l - d_j \text{®} \sigma_{j,i}$ peut être vraie.

Résultat

Si $d_l - d_j \ge \sigma_{j,i}$ **alors**

Si le rangement des tâches A_l et A_j dans l'ordre EDD permet d'ordonnancer A_l dans le délai **alors** la tâche A_j est rangée provisoirement dans la position définie par cet ordre

Sinon A_j est rejetée.

Si non

Si (1.1) est vraie **alors** la tâche A_j est rangée après A_l

Sinon A_j est rejetée.

c. $\sigma_{l,i} \text{¤} 0, \sigma_{j,i} \text{¤} 0$

Pour les deux situations 1 et 2:

$$t_d + p_j + \sigma_{j,i} = ta_i \quad \text{et} \quad t_d + p_i + \sigma_{l,i} = ta_i$$

D'où : $ta_i + p_i + p_j \le d_j$ (1.1) et

$ta_i + p_i + p_l \le d_l$ (2.2)

Soit, $B \le d_j - p_j$ (1.1) ; $B \le d_l - p_l$ (2.2)

avec $B = ta_i + p_k$

Dans le cas où $d_l - p_l \ge d_j - p_j$, si la condition (2.2) est vraie, la condition (1.1) peut être fausse, dans le cas contraire, si la condition (2.2) est fausse, la condition (1.1) peut être vraie.

Résultat

Si $d_l - p_l \ge d_j - p_j$ **alors**

Si le rangement des tâches A_l et A_j dans l'ordre EDD permet d'ordonnancer A_l dans le délai **alors** la tâche A_j est rangée provisoirement dans la position définie par cet ordre

Sinon A_j est rejetée.

Sinon

Si condition 1.1 est vraie **alors** A_j est rangée après A_l

Sinon la tâche A_j est rejetée.

IV.2.3.3 Position de la tâche aléatoire qui préserve au mieux l'avenir

Ranger les tâches aléatoires dans l'ordre EDD, cet ordre permet le décalage le plus élevé de toutes les tâches aléatoires vers la droite.

Quand une tâche aléatoire A_m arrive avec une date échue imposant à cette tâche d'être placée avant d'autres tâches aléatoires programmées afin de respecter son délai, ceci a pour conséquence de décaler certaines ou toutes les tâches qui lui succéderont vers la droite.

Chaque tâche A_j dispose d'une marge libre $\delta_j = d_j - (t_j + p_j)$ lui permettant de mouvoir vers la droite. Le décalage maximal toléré est égal à $\min \delta_j$, j successeur de m.

Soient deux tâches A_l et A_j rangée respectivement dans l'ordre après A_m.

Les marges sont, alors, données par :

$$\delta_l = d_l - (t_l - p_l),$$

$$\delta_j = d_j - (t_j - p_j)$$

avec t_l ® t_j.

Le but étant de ne pas avoir un écart important entre δ_l et δ_j, d'après les équations précédentes, il faut avoir d_j ¤ d_l pour que la différence entre les marges libres ne soit pas importante.

Remarque

Quand une tâche A_j à insérer ne peut plus être avancée vers la gauche tout en respectant l'ordre croissant des dates échues avec la tâche aléatoire qui la précède alors, si la condition sur le délai est vérifiée pour A_j, cette tâche est insérée à cette position. Dans le cas contraire, elle est rejetée.

Exemple

En considérant une seule ressource. Supposer un ordonnancement d'une tâche programmable P_{11} et d'une tâche aléatoire A_4 dont les caractéristiques sont :

$ta_{11} = 16$, $tb_{11} = 18$, $d_{11} = 22$, $p_{11} = 4$.

$r_4 = 8$, $p_4 = 3$, $d_4 = 25$.

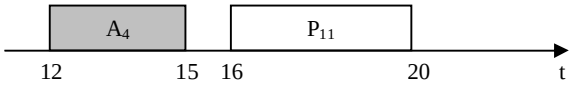


Figure IV.9: Ordonnancement des tâches P_{11} et A_4.

La date de libération de la ressource est $t_d = 12$. Avant cet instant la tâche A_6 apparaît, avec $r_6 = 8$, $p_6 = 3$, $d_6 = 25$.

Le calcul de sa position provisoire prévoit son insertion après la tâche P_{11}. A cette position, son délai n'est pas respecté.

Il faut étudier les conséquences de permutation entre les deux tâches.

Le cas d_6 ® d_4 est à considérer.

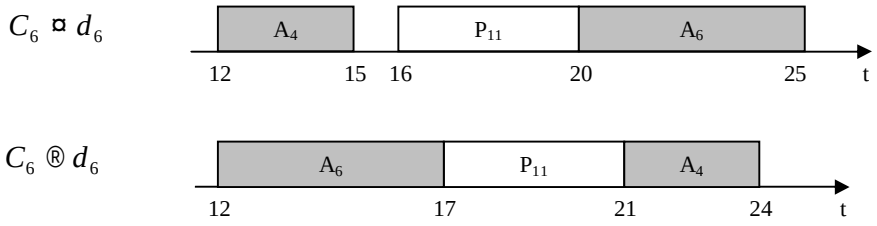


Figure IV.10: ordonnancement de la tâche A_6.

D'après la formule (20), le calcul des marges donne :

$$\sigma_{4,11} = ta_{11} - C_4 = ta_{11} - (t_d + p_4) = 16 - (12 + 3) = 1 \quad (marge\ positive)$$

$$\sigma_{6,11} = 0 \quad car \begin{cases} ta_{11} - C_6 = ta_{11} - (t_d + p_6) = 16 - (12 + 5) = -1\ (marge\ négative) \\ tb_{11} - C_6 = 18 - 17 = 1 \qquad (marge\ positive) \end{cases}$$

Comme $\sigma_{4,11}$ ¤ 0 et $\sigma_{6,11} = 0$, le cas (b) est considéré.

Il est nécessaire de vérifier $d_4 - d_6 \geq \sigma_{6,11}$. Cette relation est satisfaite. Le rangement des tâches aléatoires doit se faire dans l'ordre croissant des dates échues si les délais des tâches aléatoires sont respectés.

IV.3 Conclusion

Deux cas sont à considérer, lors de la disposition du système de deux files d'attente des tâches aléatoires, suivant la position d'insertion de ces tâches dans le programme d'exécution des tâches prévisionnelles. Il est possible d'affecter les tâches directement, suivant un algorithme proposé, dans une position ou de calculer, auparavant, la marge permettant cette insertion.

L'analyse des résultats montre que l'ordonnancement de la tâche aléatoire suivant l'algorithme d'ordonnancement temps-réel au plus tard est plus performant que le calcul de la marge libre permettant l'insertion de la tâche imprévue.

En considérant les tâches aléatoires, la position provisoire d'une tâche peut être modifiée lors de l'apparition d'une autre tâche, indépendamment des caractéristiques de cette dernière. D'où, la possibilité de changement de position d'insertion d'une tâche aléatoire dès l'apparition d'une autre.

En positionnant les tâches inattendues, le respect des délais des tâches apparaissant ultérieurement doit être pris en compte et cela en ordonnançant les tâches dans l'ordre EDD à chaque apparition d'une tâche aléatoire.

Conclusion

Générale

L'étude menée dans ce mémoire concerne une approche de résolution du problème d'ordonnancement temps-réel sur deux ressources identiques en parallèle. L'objectif est de minimiser la durée totale d'ordonnancement tout en maximisant le nombre de tâches aléatoires ordonnancées. Ceci est effectué en deux étapes :

- Résoudre le problème d'ordonnancement statique en ordonnançant toutes les tâches programmables et fixer la borne supérieure de cet ordonnancement.
- Résoudre le problème d'ordonnancement dynamique en essayant de trouver la meilleure approche d'insertion des tâches aléatoires dès leur apparition ou un temps d'attente après leur disponibilité.

Deux approches de résolution sont proposées, respectivement, dans le troisième et le quatrième chapitre, suivant le nombre de files d'attente réservées aux tâches aléatoires dans le système.

La première approche consiste à intégrer les tâches aléatoires dans le programme d'exécution des tâches programmables. Afin d'y arriver, deux algorithmes d'ordonnancement sont proposés, suivant que la tâche aléatoire est exécutée avant ou après la tâche programmable disponible à cet instant là. Les algorithmes sont appelés respectivement ordonnancement temps-réel au plus tard et au plus tôt. L'ordonnancement temps-réel au plus tard semble être plus performant que celui au plus tôt, du fait que le nombre de tâches aléatoires ordonnancées est maximisé lors de son application.

La seconde approche consiste à trouver la meilleure position d'insertion de la tâche apparaissant aléatoirement. Elle considère la disposition de chaque ressource de sa propre file d'attente des tâches aléatoires. Pour cette configuration, il est possible d'ordonnancer la tâche aléatoire dès son arrivée ou en essayant de lui trouver une marge libre pour l'insérer.

L'algorithme d'ordonnancement temps-réel au plus tard présente une meilleure performance que, respectivement, l'algorithme d'ordonnancement temps-réel au plus tôt, dans le cas de la disponibilité d'une file d'attente des tâches aléatoires, et, le calcul d'une marge libre, dans le cas de l'existence d'une file d'attente de ces tâches pour chacune des ressources.

En considérant les tâches aléatoires, la position provisoire d'une tâche peut être modifiée lors de l'apparition d'une autre tâche, dépendamment des caractéristiques de cette dernière. D'où, la possibilité de changement de position d'insertion d'une tâche aléatoire dès l'apparition d'une autre.

En positionnant les tâches inattendues, le respect des délais des tâches apparaissant ultérieurement doit être pris en compte et cela en ordonnançant les tâches dans l'ordre EDD à chaque apparition d'une tâche aléatoire.

Nous avons constaté, dans la première approche, que la fréquence d'apparition des tâches aléatoires, lors de l'ordonnancement temps-réel, n'affecte aucunement le nombre de tâches ordonnancées. Ce nombre dépend uniquement des marges libres disponibles en ordonnançant toutes les tâches programmables. Par conséquent, les caractéristiques des tâches aléatoires doivent correspondre à ces marges.

Les tâches aléatoires, intervenant après ordonnancement de la dernière tâche programmable ordonnancée, sont affectées aux différentes ressources suivant un ordonnancement statique (une simple règle d'affectation).

L'analyse des résultats obtenus en appliquant les quatre algorithmes, au même exemple, a montré, que l'exécution de la tâche programmable au plus tard permet aux tâches aléatoires de disposer de marges libres larges permettant leur exécution.

Cette étude est une initiation dans le domaine de la productique. Elle m'a permis d'acquérir des connaissances dans le domaine de l'ordonnancement en général et, plus particulièrement, l'ordonnancement temps-réel dans une organisation d'ateliers à ressources parallèles.

Ce mémoire est une tentative de contribution à l'ordonnancement temps-réel sur des ressources identiques parallèles. Il ouvre, cependant, des perspectives pour des travaux futurs dans ce domaine :

- Ø Nous avons considéré que la sélection de la tâche aléatoire à ordonnancer est basée sur des règles de priorité statiques. Il serait intéressant d'analyser l'impact d'un choix dynamique de ces règles. Dans ce cas, chaque sélection de la tâche aléatoire est basée sur une règle de priorité différente d'un ordonnancement à un autre.
- Ø Nous avons considéré que le respect des intervalles temporels des tâches aléatoires est primordial. L'acceptation d'un retard sur les délais de ces tâches modifierait complètement l'ordonnancement. Il est intéressant de considérer, comme objectif de production à optimiser, la somme des retards ou le retard maximal.
- Ø Etudier le cas où les durées opératoires (les tâches programmables et les tâches aléatoires) sont variables est intéressant étant donné que l'exemple étudié, dans ce travail, considère dans un premier temps les durées opératoires égales pour les deux types de tâches et les durées opératoires des tâches aléatoires variables.
- Ø Comme une autre perspective, il serait intéressant, voire même nécessaire, de trouver une approche d'évaluation de la performance d'une heuristique développée pour un ordonnancement temps-réel. Une approche empirique ou analytique est envisageable.

Annexes

Annexe I

1. La production

Dans la littérature, nous trouvons différentes définitions de la production dont les suivantes parmi d'autres :

La production est une transformation de ressources appartenant à un système productif et conduisant à la création de biens ou de services. Les ressources mobilisées à cette fin peuvent être de quatre types [Gia 88], [Lam 91] et [Art 97] :

- Des équipements (bâtiments, machines,...),
- Des hommes (opérateurs intervenant directement dans le processus de transformation ou contribuant d'une manière ou d'une autre à son bon déroulement),
- Des matières (matières premières, composants,...),
- Des informations techniques ou procédurales (gammes[1], nomenclatures[2], consignes, procédures,…) ou relatives à l'état d'utilisation du système productif (ce qui permet de programmer la production et de réagir aux perturbations observées,…).

La production, dans [Gra 02], se définit comme la transformation de ressources ayant pour objectif la création de biens ou de services. Concrètement, l'entreprise modifie les caractères physiques, spatiaux ou temporels des ressources dont elle dispose en les transformant (les matières premières), en les transportant ou en les stockant.

La production est une activité opérationnelle utilisant des ressources acquises (matériels et humains) en vue de créer des biens matériels ou d'assurer des services d'utilité moyennant une ou plusieurs transformations. Ces transformations sont caractérisées par un ensemble d'opérations, qui peuvent consister, soit dans une modification de la forme des pièces, soit dans la combinaison de plusieurs pièces. L'ensemble de ces transformations constituant la production seront soumises à des critères impératifs de coût, délai, quantité et qualité [Ben 87].

2. Caractéristiques de la production

2.1 Nature de la production

La nature de la production permet de distinguer entre les entreprises fournissant des services (bureaux d'études, les entreprises de transport,…), les entreprises de montage (les industries de l'électronique et d'assemblage, les chaînes de montage automobiles,…)

[1] Gamme : un document décrivant en détail la séquence d'opérations de fabrication, d'assemblage, d'inspection ou de transport nécessaires à la fabrication d'un composant ou d'un produit fini.
[2] Nomenclature : liste structurée des composants et des matières nécessaires à la fabrication d'un composant ou d'un produit.

et celles fabricant des produits après transformation de la matière première (les industries mécaniques, les industries chimiques,…).

2.2 Modes de production

Il existe essentiellement deux principaux modes de production :

2.2.1 Production continue

Elle concerne tous les produits pour lesquels le processus de transformation de la matière première ne s'interrompe pas entre deux installations technologiques consécutives (machines,...). Ce mode de production est caractérisé par l'inexistence de stockage entre les postes de travail sauf parfois pour la régulation du flux, dans ce cas là, on parle souvent d'industries de process[3]. Ces entreprises appartiennent principalement à la pétrochimie, la chimie lourde, la sidérurgie mais on en trouve également dans le secteur des industries alimentaires.

2.2.2 Production discrète

Il concerne les produits réalisés suivant un processus de fabrication pouvant être fractionné pour permettre de reprendre des produits semi-finis. Dans ce cas, la présence d'espaces de stockage entre ateliers (machines) est requise. On parle d'industries manufacturières et on cite à titre d'exemple les entreprises d'assemblages.

Il est courant de trouver des entreprises reliant ces deux modes, d'où l'existence d'un troisième mode dit production mixte ou hybride.

2.3 Les types de production

Les types de production sont au nombre de trois : [Hen 99] et [Woo 65]

2.3.1 Production par lot

Le lot de fabrication peut être composé de quelques unités à plusieurs centaines d'unités, cela pose le problème de la fréquence des réglages de machines et des changements d'outils. Généralement, nous essayons de trouver une taille de lot acceptable, dans ce cas, nous parlons souvent de quantité économique de production. Ce type de production est souvent caractérisé par des temps de réglages machines importants entre les différents produits.

[3] Process : terme désignant des entreprises produisant des produits par opérations de fusion, de séparation ou de transformations chimiques.

2.3.2 Production de masse

Elle concerne les produits standards qui sont fabriqués en très grande quantité et leur écoulement sur le marché est relativement facile par rapport au type de production unitaire qui n'est fabriqué que sur commande.

2.3.3 Production unitaire

Ce type de production concerne les produits singuliers et spécifiques (construction navale, ponts, bâtiments,…). Ce genre de produits est réalisé sur commande et nécessite un devis de production.

2.3 Les sous fonction du pilotage de la production

Elles sont citées ci-après :

La planification

Elle détermine le plan directeur de production, compromis entre les objectifs commerciaux, financiers et de la production de l'entreprise. L'horizon de planification est souvent le moyen ou le long terme. Cette fonction est caractérisée surtout par un horizon temporel étendu et par un degré de finesse des informations traitées très générique.

La programmation

Elle établit un programme prévisionnel de production basé sur le plan directeur de production et sur l'état des stocks. L'horizon de la programmation est souvent le très court terme (la semaine).

L'ordonnancement

Il détermine d'abord les priorités de passage des travaux sur les ateliers (séquence) et les différentes affectations temporelles des ressources. En fait, cette sous fonction est constituée d'une partie de la séquence et d'une autre d'affectation. L'horizon peut être le court ou le moyen terme.

La conduite

Elle regroupe les activités décisionnelles qui sont traduites en ordres et transmises au niveau commande. Ce dernier niveau gère la réalisation du processus physique de fabrication et retourne au niveau conduite un ensemble de données de suivi. Cette approche reprend la notion de boucle de rétroaction.

La conduite en temps-réel d'un système de production est une tâche complexe qui requiert des connaissances dans les domaines de l'informatique, de l'automatique, de la production, de la communication homme – machine, etc. Sa difficulté est due [Dra 98] :

- à la complexité du problème d'ordonnancement ;
- à la nécessité de résoudre le problème d'ordonnancement dans un contexte perturbé ;
- au problème intrinsèque du suivi, souvent banalisé dans la plupart des systèmes ;
- à l'intégration du système de conduite dans le système global de gestion de production.

La commande

Elle se charge de la traduction de l'ordre de fabrication en une séquence d'instructions exécutables par une ressource.

3. Les classes d'ordonnancement

Les classes d'ordonnancement sont :

- **Ordonnancement de projet (Project Scheduling)**

Ses champs d'application sont :

1. Gestion de grands projets pour lesquels un grand nombre de tâches est à réaliser (tâches entre lesquelles existent des contraintes d'antériorité et des possibilités de parallélisme) ;
2. Le problème de type « série unitaire » se définissant par une mobilisation de toutes les ressources pour la réalisation d'un projet de production exécuté sur une assez longue période (travaux publics, construction navale).

Le but est de connaître le déroulement du projet, prévoir l'enchaînement des opérations, estimer la date prévue de chaque opération et estimer le coût du projet. Les outils techniques de planification associés sont :

- PERT (Program Evaluation and Review Technic);
- CPM (Critical Path Method);
- Méthodes des POTENTIELS duale du PERT.

L'ordonnancement de projet ne constituant pas le cadre de notre travail, nous ne donnons pas de détail sur ces outils techniques que nous pouvons trouver dans [Bot 99] et [Hen 03].

- **Ordonnancement d'ateliers (Job Scheduling) (Ordonnacement des Fabrications)**

Dans le cas d'entreprises travaillant sur des petites et moyennes séries, il s'agit de gérer dans les ateliers le passage des travaux sur les machines. L'ordonnancement d'atelier est d'ordre opérationnel et on dit souvent que c'est l'ordonnancement des fabrications [Hen 03].

Dans la littérature une grande diversité de méthodes pour ordonnancer les tâches d'un atelier, elles sont généralement basées selon deux dimensions. La première dimension est basée sur la disponibilité des tâches qui peut être statique ou dynamique. La seconde est basée sur le dynamisme de la méthode d'ordonnancement qui peut être prévisionnelle : un ordonnancement est statique (prévisionnel) si l'ensemble des informations nécessaires à sa résolution est fixé a priori et n'est pas remis en cause durant la résolution (ensemble des tâches, des ressources et leurs caractéristiques), ou temps-réel (real time) nous parlons d'ordonnancement dynamique ou « temps-réel, réactif » quand les décisions d'ordonnancement des opérations sur une machine sont prises une par une chaque fois qu'une machine se libère ou qu'une opération devient disponible. Ces problèmes se caractérisent par des arrivées successives des tâches sans références précises à une période donnée.

	Statique	**Dynamique**
Univers	Certain	Aléatoire
Outils	Méthodes d'optimisation Recherche opérationnelle	Théorie des files d'attente Simulation

Tableau 1 : Ordonnancement statique et dynamique.

Par ailleurs la résolution des problèmes d'ordonnancement doit considérer le modèle déterministe ou le modèle stochastique [Gaz 03].

- ***Le modèle déterministe***

Dans un ordonnancement en contexte déterministe, les caractéristiques de l'environnement (durées des tâches, disponibilité des machines) sont connues et changent avec le temps.

- ***Le modèle stochastique***

L'environnement réel de production est le plus souvent sujet à plusieurs sources d'incertitude. Ces sources, qui caractérisent un modèle stochastique, peuvent être : les pannes de machines, les temps de traitement aléatoires, etc.

Dans notre étude nous traitons des problèmes d'ordonnancement d'ateliers en deux parties statique/ déterministe et dynamique/stochastiques.

4. Les éléments fondamentaux de l'ordonnancement

Les éléments de l'ordonnancement d'un problème d'atelier sont :

4.1 Les tâches

Une tâche (job) correspond à la fabrication d'un objet (ou la fourniture d'une prestation, d'un service) [Bot 99].

Une tâche est une entité élémentaire de travail localisée dans le temps par une date de début t_i ou de fin c_i, dont la réalisation est caractérisée par une durée de traitement p_i (si i n'est pas interrompue alors $c_i = t_i + p_i$) et par l'intensité a_i^k avec laquelle elle consomme certain moyens k, ou ressources. Pour simplifier, on supposera que pour chaque ressource requise, cette intensité est constante durant l'exécution de la tâche. Selon les problèmes, les tâches peuvent être exécutées par morceaux ou sans interruption. On parle alors, respectivement, de problèmes préemptifs et non préemptifs. Suivant les conditions de fabrication, chacune des phases, dans l'exécution d'une tâche, peut fortement conditionner le temps de séjour d'un produit dans un atelier [Esq 99].

4.2 Les opérations

Lorsque la réalisation d'une tâche requiert successivement plusieurs ressources, on dira que chaque ressource réalise une opération de la tâche considérée.

La gamme d'une tâche décrit la séquence des opérations de cette tâche [Gaz 03].

Les gammes

Les gammes de fabrication décrivent la succession des opérations à réaliser pour passer d'une matière preière au composant ou du produit semi-fini au produit fini, c'est le processus d'élaboration. On trouve dans la littérature non spécialisée, l'appellation de routage de la tâche et dans la littérature anglophone le terme : routing. Il existe plusieurs sortes de gammes, dont on distingue [Hen 99] :

- **Gamme libre *:*** L'exécution des opérations est indépendante de l'ordre, il n'existe aucune contrainte de succession. Les gammes libres caractérisent les problèmes d'ordonnancement du type Open Shop.
- ***Gamme linéaire :*** L'ordre d'exécution des opérations est entièrement imposé et prédéterminé. Ce genre de gammes est présent dans le cas du Job Shop (chaque tâche a sa propre route à suivre autrement dit toutes les tâches ont des gammes

prédéterminées mais non identiques) et celui du Flow Shop où toutes les tâches ont le même chemin à suivre (la même route ou la même gamme).

- ***Gamme mixte ou semi-linéaire :*** L'ordre d'exécution des opérations est partiellement déterminé, on trouve des tâches qui ont des gammes prédéterminées (on sait a priori la route à suivre par ces tâches), et d'autres non (on ne sait pas, a priori, la route à suivre par ces tâches)

Gamme technique : Elle est due essentiellement au choix prédéterminé de certaines tâches à passer sur des machines spécifiques.

4.3 Les ressources

Une ressource j est un moyen technique ou humain destiné à être requis pour la réalisation d'une tâche et disponible en quantité limitée, sa capacité a_j est supposée constante. On distingue plusieurs types de ressources [Esq 99] et [Lop 01] :

- Ø **Ressource disjonctive :** La ressource disjonctive (non partageable) principalement dans le cas de ressource renouvelable, est une ressource qui ne peut exécuter qu'une tâche à la fois (machine outil, robot manipulateur, etc.).
- Ø **Ressource consommable :** Une ressource est consommable si elle devient non disponible après avoir été utilisée par une ou plusieurs tâches (matière première, budget, etc.), la consommation globale au cours du temps est limitée.
- Ø **Ressource cumulative :** La ressource cumulative (partageable) est une ressource qui peut être utilisée par plusieurs tâches simultanément (équipe d'ouvriers, poste de travail).
- Ø **Ressource doublement contrainte :** Une ressource doublement contrainte lorsque son utilisation instantanée et sa consommation globale sont toutes les deux limitées (sources d'énergie, financement, etc.).
- Ø **Ressource renouvelable :** Une ressource est renouvelable si, après avoir été utilisée par une ou plusieurs tâches, elle est à nouveau disponible en même quantité (les hommes, les machines, l'espace, l'équipement en général, etc.). La quantité de ressource utilisable à chaque instant est limitée.

La nature des ressources prises en considération permet de dresser une typologie des problèmes d'ordonnancement. La figure suivante illustre cette dernière.

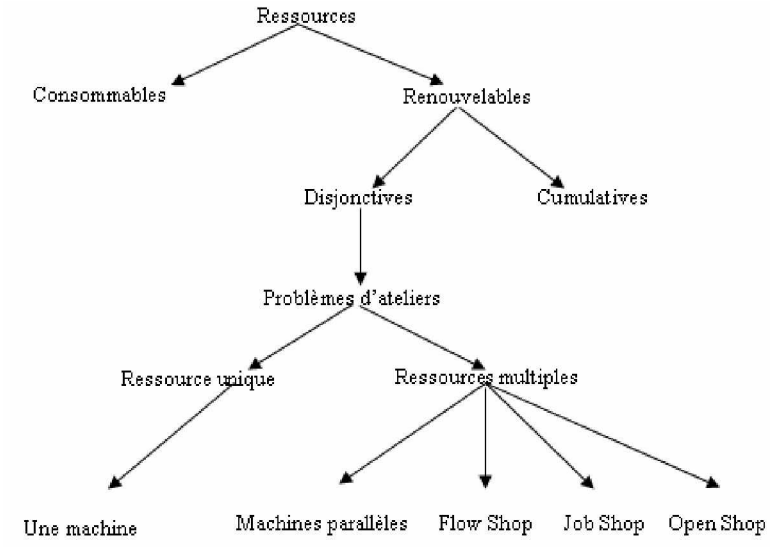


Figure 1 : Typologie par les ressources des problèmes d'ordonnancement [Esq 99].

Remarque : Les problèmes non préemptifs à ressources renouvelables disjonctives couvrent une classe importante d'applications : les problèmes d'ateliers.

4.4 Les contraintes

Une contrainte exprime des restrictions sur les valeurs que peuvent prendre conjointement une ou plusieurs variables de décision. On distingue :

- **Contraintes cumulatives.** Elles sont liées à l'utilisation simultanée d'une même ressource par plusieurs tâches et indiquent que leur capacité est limitée.
- **Contraintes disjonctives** spécifient que le traitement d'un ensemble de tâches sur la ressource disjonctive ne peut se faire simultanément au même instant.
- **Contraintes physiques** spécifient les caractéristiques limitant le fonctionnement de l'atelier, dues à l'organisation physique des moyens de production.
- **Contraintes temporelles.** Elles regroupent les contraintes de précédences et de localisation temporelle.
  - Les contraintes de précédences entre les tâches prennent en compte la succession des opérations de la gamme opératoire. Ces contraintes, indiquant que la date de début d'une tâche j doit commencer après la date de fin de réalisation de la tâche i ;
  - Les contraintes de localisation temporelle définissent l'intervalle de temps sur lequel la tâche doit être traitée. Cet intervalle est limité par les valeurs des dates de début au plutôt et de fin au plus tard.

4.5 Les propriétés des ordonnancements

- **Ordonnancement actif.** Dans un ordonnancement actif aucun glissement à gauche-local ou global[4]- n'est possible. Aucune tâche ne peut être commencée plus tôt sans reporter le début d'une autre.
- **Ordonnancement semi actif.** Dans un ordonnancement semi actif, aucun glissement à gauche n'est possible. On ne peut avancer une tâche sans modifier la séquence sur la ressource.
- **Ordonnancement admissible.** Un ordonnancement est dit admissible, s'il respecte les contraintes du problème (dates limites, précédence, limitation des ressources…).
- **Ordonnancement au plus tôt / au plus tard.** Le calcul de l'ordonnancement au plus tôt consiste à attribuer à chaque tâche j une date de début au plus tôt t_j compte tenu des dates de début au plus tôt t_i des tâches i qui précèdent j. Si l'on admet comme objectif, déterminer le projet au plus tôt, on peut déterminer symétriquement pour chaque tâche une date de début au plus tard t_i en calculant cette fois-ci la longueur des plus longs chemins entre tout sommet i et le dernier sommet (sommet Fin).
- **Ordonnancement sans retard.** Dans cet ordonnancement, on ne doit pas retarder l'exécution d'une tâche si celle-ci est en attente et si la ressource est disponible.

La figure ci-après représente les différents ordonnancements.

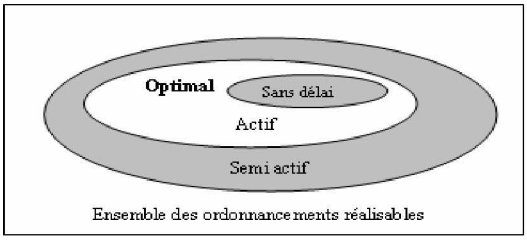


Figure 2 : Diagramme de Venn [Our 01].

5. Les organisations d'ateliers

Les organisations d'ateliers sont répertoriées en trois catégories :

[4] Nous parlons de glissement à gauche local lorsqu'on avance le début d'une tâche sans remettre en cause l'ordre relatif entre les tâches. Nous parlons de glissement à gauche global lorsqu'on avance le début d'une tâche en modifiant l'ordre relatif entre au moins deux tâches.

5.1 Les organisations à ressource unique

Les organisations à ressource unique sont un cas que l'on peut rencontrer rarement dans les entreprises industrielles, par contre ces organisations sont plus fréquentes dans les systèmes informatiques où on dénote souvent la présence d'une seule imprimante, d'un seul microprocesseur et plusieurs usagés (programmeurs compilant leurs codes, des impressions de documents,…). Parfois, la présence d'une machine goulot dans une chaîne de fabrication peut pousser certain chercheurs à aborder le problème comme une seule machine à étudier afin d'utiliser des méthodes et heuristiques appropriées aux systèmes à une machine.

5.2 Les organisations à ressources multiples

Dans ce type d'organisations, on distingue plusieurs cas selon l'ordre d'exécution des opérations des tâches :

- Le cas des organisations à ressources parallèles est caractérisé par la présence de plusieurs machines pour l'exécution d'un travail ne nécessitant qu'une seule (figure 3).

Pour les problèmes d'ordonnancement, nous confondons souvent les ressources avec les machines. Généralement, les machines sont groupées par atelier. Les exemplaires de machines d'un atelier diffèrent souvent uniquement par la rapidité d'exécution des opérations qu'elles réalisent. Nous distinguons alors plusieurs types d'exemplaires de machines :

- Les exemplaires de machines identiques : la durée de traitement est la même sur toutes les ressources.
- Les exemplaires de machines uniformes : la durée de traitement varie uniformément en fonction de la performance intrinsèque de la ressource.
- Les exemplaires de machines différentes : la durée de traitement est totalement variable entre les différentes ressources.

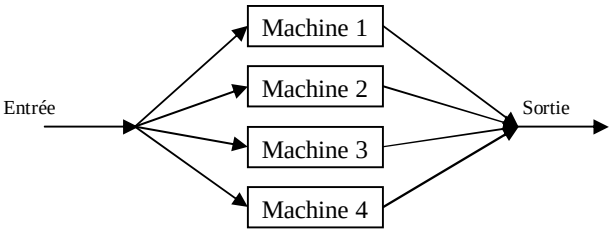


Figure 3 : Exemple d'organisation parallèle avec 4 machines.

F Dans le cas des organisations du type Flow Shop Simple [Hen 96], les travaux ont une gamme identique (un même cheminement), le travail (la tâche) est ainsi constitué de plusieurs opérations visitant différentes machines et enchaînées de manière linéaire suivant une chaîne [Esq 99], par conséquent, l'ordre de passage sur les postes de travail est le même (figure 4). Sinon nous parlons d'une organisation du type Flow Shop hybride. La figure 5 illustre cette dernière.

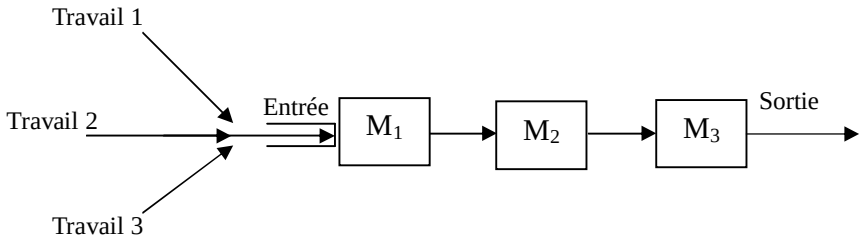


Figure 4 : Flow Shop à 3 machines dont la gamme est $M_1 \rightarrow M_2 \rightarrow M_3$.

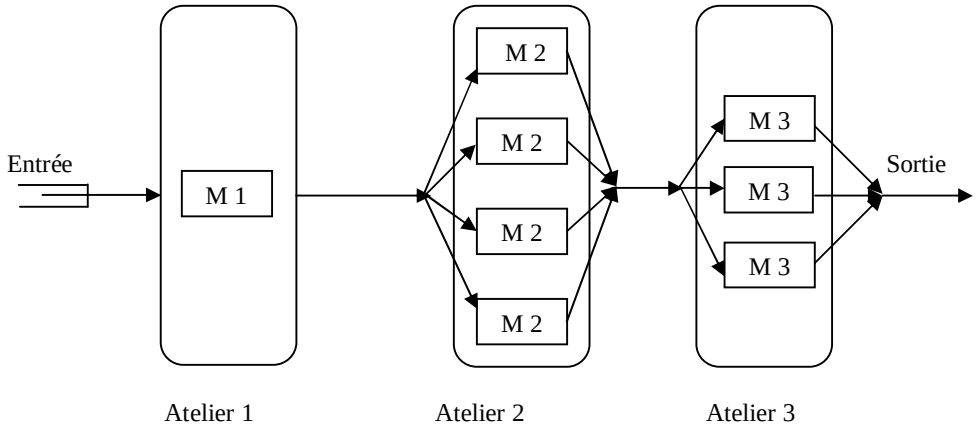


Figure 5 : Flow Shop hybride à 3 Ateliers dont la gamme est $M_1 \rightarrow M_2 \rightarrow M_3$.

F Dans le cas des organisations du type Job Shop simple, le cheminement d'exécution des travaux (des tâches) se fait suivant un cheminement propre au travail. A chaque travail est associée une gamme propre (chaque travail passe par toutes les machines dans un ordre fixé qui lui est propre [Esq 99], (figure 6)). Dans le cas contraire, nous considérons une organisation du type Job Shop Hybride (figure 7).

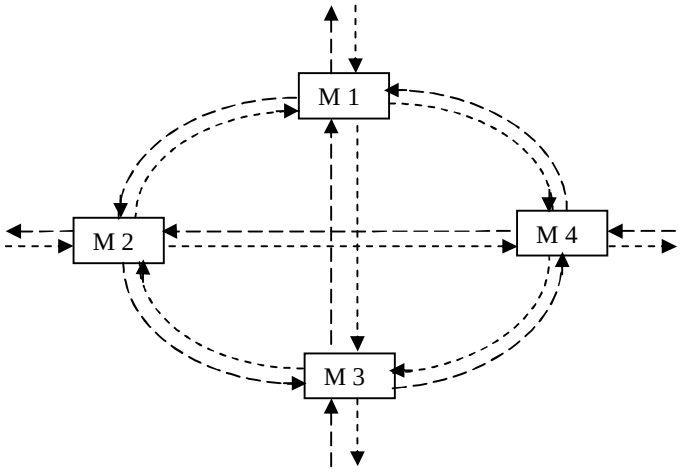


Figure 6: Exemple d'organisation Job Shop Simple à 4 machines.

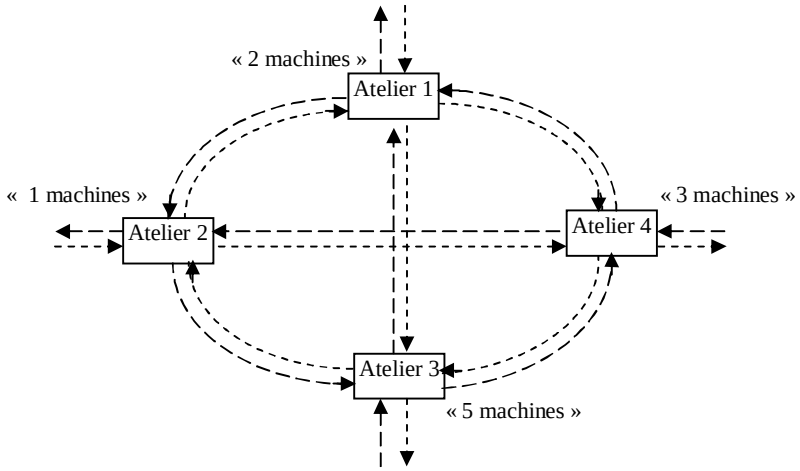


Figure 7: Exemple d'organisation Job Shop hybride à 4 ateliers.

5.3 Les organisations multi lignes parallèles

Les systèmes à organisation multi lignes sont des systèmes hybrides avec des ressources dédiées. A chaque ligne de production est assignée une certaine gamme de produits. Un exemple d'organisation multi lignes parallèles est illustré par la figure 8.

Dans certains cas, il arrive que les lignes soient des organisations hybrides. Ces problèmes ont deux aspects, une organisation linéaire (Flow Shop Simple, le cas où on a une seule ligne de production) et l'autre de machines parallèles (le cas où l'on a plus d'une ligne et chaque ligne dispose d'une seule machine). La plupart de ces problèmes sont généralement résolus en associant des heuristiques et l'expérience humaine.

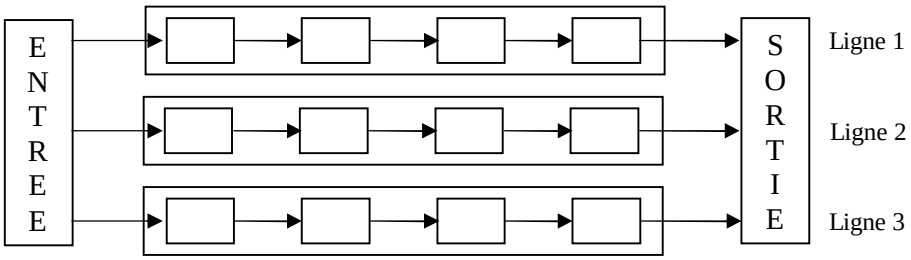


Figure 8: Exemple d'une organisation multi-lignes parallèles.

6. Les critères de performances

Les critères de performances sont classés en quatre catégories :

6.1 Les critères basés sur la date d'achèvement

C_{max} = max(C_i) : est la plus grande date d'achèvement, ou la durée totale (le Makespan), le temps nécessaire pour finir toutes les tâches.

$F_i = C_i - r_i$: est la durée de séjours (Flow time).

F_{max} = max(F_i) : est le maximum du temps de séjours.

$W_i = C_i - r_i - \sum p_{ij}$ **:** est le temps d'attente.

$W_{max} = \max\{W_i / i \in [1,n]\}$: est le maximum des temps d'attente.

$\sum C_i$ **:** est la somme des dates de fin d'exécution.

$\sum w_i C_i$: est la somme pondérée des dates de fin d'exécution.

Les critères suivants sont également basés sur les dates de fin des tâches : $\overline{C}, \overline{F}$ *et* $\overline{W}$. Ce sont les moyennes arithmétiques, respectivement des dates de fin, des temps de séjours et des temps d'attente. Dans certains cas, ces critères sont associés à une pondération w des tâches.

6.2 Les critères basés sur la date de livraison

$L_i = C_i - d_i$: est le retard algébrique.

L_{max} = max(L_i) **:** est le plus grand retard algébrique (maximum lateness).

$T_i = \max(0, L_i)$: est le plus grand retard absolu (Tardiness).

$E_i = -\min(0, L_i)$: est l'avance absolue (Earlyness).

Nous trouvons aussi des critères sous forme de moyennes arithmétiques, notée $\overline{L}, \overline{T} et\ \overline{E}$, ou bien encore pondérés par une pénalité. Le critère $N = \sum_{1=1}^{n} U_i$, représentant le nombre de tâches en retard, est également des plus utilisés.

6.3 Les critères basés sur les volumes des en-cours

Le critère $\overline{C}$ est des plus utilisés pour la minimisation des volumes des en-cours (l'en-cours est le plus souvent caractérisé par le temps moyen des travaux [Gia 88]). C'est la moyenne arithmétique de l'ensemble des dates de fin d'exécution.

Une opération est toujours caractérisée par le numéro de la tâche et l'atelier sur lequel elle passe. Soient s_o la date de début de l'opération o et p_o sa durée d'exécution. A chaque instant t on peut calculer les mesures suivantes :

$N_p(t) = \sum_{i=1}^{n} e_i(t)$ est le nombre de tâches en cours d'exécution.

Avec

$$e_i(t) = \begin{cases} 1 \quad si \quad (s_0 \le t < s_0 + p_0) \\ 0 \quad \text{sin } on \end{cases}$$

La maximisation de ce critère permettra l'exploitation de façon quasi optimale des ressources. Plus la valeur de ce critère est grande, plus le nombre de machines en attente est faible.

$N_w(t) = \sum_{i=1}^{n} e_i(t)$ est le nombre de tâches en attente.

Avec

$$e_i(t) = \begin{cases} 1 \quad si \quad (s_0 + p_0 \le t < s_{0+1}) \\ 0 \quad \text{sin } on \end{cases} .$$

Minimiser ce critère permet de diminuer le nombre de tâches en attente dans les stocks intermédiaires des machines et par conséquent, de réduire la capacité intrinsèque des stocks.

$N_f(t) = \sum_{i=1}^{n} e_i(t)$ est le nombre de tâches terminées sur la dernière machine.

Avec

$$e_i(t) = \begin{cases} 1 & si \quad (c_i \le t) \\ & \\ 0 & \text{sin } on \end{cases}.$$

La maximisation de ce critère permet de répondre au plus vite à la demande des clients.

Nous pouvons observer que $N_p(t) + N_w(t) + N_f(t) = N$ (N est le nombre de tâches à ordonnancer pour un horizon fixé) et, par conséquent, les objectifs de production dans ce cas peuvent être complètement antagoniste [Hen 99] et [Rin 76].

6.4 Les critères basés sur l'utilisation des ressources

L'exploitation des ressources peut être aussi un souci majeur du gestionnaire. En plus du critère de minimisation de la plus grande date de fin C_{max} (Makespan), les critères suivants sont parmi les mieux appropriés, soient M_v le nombre de machines disponibles dans l'atelier v et C_{max} la plus grande date de fin de l'ordonnancement.

$\overline{U} = \left(\left(\sum_{i=1}^{n} \sum_{j=1}^{v} p_{i,j} \right) * \left(\sum_{j=1}^{v} M_j * C_{\max} \right)^{-1} \right)$ représente l'utilisation moyenne des ressources.

Elle permet de signaler éventuellement la sous-utilisation des ressources de fabrication de l'entreprise. Elle indique par conséquent, les périodes pleines et les périodes creuses de la chaîne de fabrication.

$I_v = \left(C_{\max} - \sum_{i=1}^{n} p_{i,v} \right) * M_v$ est le temps d'inactivité (ou temps de repos, d'inoccupation) de l'atelier v (Idle time). Cet indicateur donne le temps d'inactivité de l'atelier correspondant et sa minimisation permettra une plus grande exploitation des ressources.

$A_p(t) = \sum_{s_o \le t \le s_o + p_o} p_o$ est la quantité de travail en marche dans l'atelier à l'instant t. Maximiser le $\overline{A}_p$ conduit à une utilisation efficiente des machines.

Les théorèmes suivants présentent les équivalences entre les critères [Rin 76]:

Théorème 1 : les critères suivants sont équivalents : C_{max}, $\overline{N}_P$, $\overline{A}_P$, $\sum I_v$, $\sum w_v I_v$ et $\overline{U}$.

Théorème 2: les critères suivants sont équivalents : $\sum C_i$, $\sum F_i$, $\sum W_i$, $\sum L_i$, $\overline{C}$, $\overline{F}$, $\overline{W}$ et $\overline{L}$.

Théorème 3: les critères suivants sont équivalents : $\sum w_i C_i$, $\sum w_i F_i$, $\sum w_i W_i$ et $\sum w_i L_i$.

Théorème 4 : un ordonnancement optimal pour le critère L_{max} est aussi optimal pour le critère T_{max}.

Théorème 5: les critères suivants sont équivalents : $\overline{N}_p + \overline{N}_w$, $\overline{N}_f$ et $\overline{C} / C_{\max}$.

Théorème 6: les critères suivants sont équivalents : $\overline{N}_w$ et $\overline{W} / C_{\max}$.

Pour la démonstration de ces théorèmes, voir [Rin 76] et [Gaz 03].

Nous nous intéressons dans ce mémoire à la plus grande date d'achèvement (Makespan) et le nombre de tâches exécutées.

7. Classification des problèmes d'ordonnancement

Les classifications les plus connues sont :

- Ø Classification de Conway, Maxwell et Miller (1967) [Con 67]
- Ø Classification de Graham & al. (1979)
- Ø Classification de Lawler & al. étendue par Vignier & al. (1995)[Vig 95]

7.1 Classification de Conway, Maxwell et Miller 1967

La première classification a été proposée par Conway, Maxwell et Miller en 1967, et fut étendu par Lenstra en 1976. Elle distingue quatre types d'informations :

- Les tâches à effectuer,
- Le nombre et le type de machines disponibles,
- L'organisation des ressources et les contraintes à respecter,
- Le critère avec l'ordonnancement est évalué.

Elle utilise quatre champs, à chacun correspond un paramètre, pour identifier les problèmes d'ordonnancement : A/B/C/D

Le champ A décrit le processus d'arrivée des tâches. Pour un problème dynamique, A identifie la probabilité de distribution des arrivées des tâches, et pour un problèmes statique, le nombre de tâches n.

Le champ B indique le nombre m de machines dans l'organisation.

Le champ C représente le type de l'organisation (I, U ou R pour les machines parallèles, F pour le 'Fow Shop', O pour 'l'Open Shop', G pour le 'Job Shop'). Dans le cas d'une seule machine, il n'y a pas d'organisation spécifique et ce troisième paramètre est omis.

Le champ D décrit le critère avec lequel un ordonnancement est évalué (la fonction objectif à maximiser ou à minimiser).

N/6/I/C_{max} est un problème d'ordonnancement à machines identiques en parallèle, de n tâches avec six machines, dont le critère à optimiser est la durée totale d'exécution de toutes les tâches C_{max} (la plus grande date de fin).

Dans sa version originale, Classification de Conway, Maxwell et Miller, la notion hybride ou d'atelier avec plus d'une machine est inexistante. Cette notation a été étendue par Lenstra en 1976. L'auteur a rajouté à la notation de Conway, Maxwell et Miller un cinquième paramètre en éclatant le troisième paramètre en deux nouveaux paramètres $C^{'} et C^{''}$, pour inclure les contraintes. Toutefois, cette extension reste insuffisante pour prendre la dimension hybride de certain problème d'ordonnancement.

7.2 Classification de Graham & al 1979

La classification de Graham & al. (1979) est basée sur trois paramètres notés α/β/γ.

Le premier paramètre α est composé de deux sous-paramètres $\alpha_1 et \alpha_2$ $(\alpha = \alpha_1 \alpha_2)$. $\alpha_1 = \{\Phi, P, Q, R, O, F, J\}$ caractérise l'organisation des machines dans le système (Φ : une seule machine ;P :machine identique, Q :machines uniformes, R : machines différentes (indépendante), O : machines dédiées pour les systèmes du type 'Open Shop', F : machines dédiées pour les systèmes du type 'Flow Shop', J : machines dédiées pour les systèmes du type 'Job Shop'). $\alpha_2 \in \{\Phi, k\}$ est le nombre de machines (dans le cas où $\alpha_2 = \Phi$, le nombre de machines est supposé variable).

Le deuxième paramètre β caractérise les restrictions éventuelles. Il est composé de cinq sous-paramètres $\beta_1, \beta_2, \beta_3, \beta_4 et \beta_5$ principalement.

$\beta_1 \in \{\varnothing$, prmp$\}$ précise que la préemption des tâches est autorisée ou non. β_2 identifie les ressources additionnelles du système. Les contraintes de précédence sont notées dans β_3. $\beta_4 \in \{\varnothing, r_i\}$ représente les dates de disponibilité des tâches. β_5 définit les particularités des durées opératoires s'il y en a.

Le troisième paramètre γ est l'indicateur de performance (critère d'optimisation ou la fonction objectif).

La notion hybride n'a pas été prise dans cette classification. Sriskandarajah & Sethi (1989) ont proposé une version étendue du travail de Graham et Lawler. Les auteurs ont modifié les définitions des paramètres α (α = Fm : pour Flow Shop à m machines) et β

($\beta = \{1,2,3,\ldots,m\}$), le nombre de machines dans chaque atelier. La modification ainsi faite ne prend pas en considération les contraintes supplémentaires [Hen 99].

7.3 Classification de Lawler & al. étendue par Vignier & al 1995

La notation de Vignier & al (1995) est inspirée par celle proposée par Graham & al. en 1979 puis étendue par Lawler & al. en 1989 et Blazewicz & al. en 1993. Cette classification a été adoptée au Flow Shop hybride. Elle est basée sur les trois paramètres $\alpha|\beta|\gamma$.

Le paramètre α est de la forme $\alpha = [\alpha_1, \alpha_2], \left\{\left(\alpha_3, \alpha_4^{(1)}\right)\right\}\begin{bmatrix}\alpha_2 \\ l\end{bmatrix}$ où :

- α_1=FH est la représentation du Flow Shop hybride.
- α_2 est le nombre d'ateliers.
- $\alpha_3 \in \{\Phi, P, Q, R, O, F, J\}$ caractérise l'organisation des machines dans le système.
- $\alpha_4 = (M^{(t)}, M^{(t)}(t))$ représente le nombre de machines dans chaque atelier.

Le paramètre β est la concaténation de plusieurs sous-paramètres (β_1, β_2, β_3, …,β_n) reflétant les particularités de chaque problème étudié.

Le paramètre γ correspond à la classique mesure de performance (fonction objectif).

Par exemple, FH5,($P3^{(1)}$, $P3^{(2)}$, $P2^{(3)}$, $P2^{(4)}$, $P2^{(5)}$) est l'identification du Flow Shop hybride à cinq atelier composé de trois machines identiques aux deux premiers ateliers et de deux machines identiques pour les autres (le même problème peut être noté de également ainsi : $FH5, \left(\left(P3^{(l)}\right)_{l=1}^{2} \left(P2^{(l')} \right)_{l'=3}^{5} \right)$, [Vig 95]. L'utilisation des indices ne facilite pas la tâche de lecture ni celle d'écriture. Si au lieu d'avoir 5 ateliers on en a 10 dans l'exemple ci-dessus, l'écriture sera plus fastidieuse, difficile à lire et prendra plus de place. Dans cette dernière classification le nombre de tâche n'apparaît pas. C'est moins grave dans un ordonnancement statique que dans un ordonnancement dynamique où on doit spécifier la densité de distribution de l'arrivée des tâches [Gaz 03] [Hen 99].

7.4 Contraintes et restrictions supplémentaires

Les notations et définitions suivantes représentent quelques contraintes et restrictions que nous trouvons dans les champs C'' (respectivement β) de classification de Conway, Maxwell et Miller 1967, étendue par Lenstra en 1976 (respectivement de la classification de Graham & al. 1979, étendue par Lawler & al. 1989 et celle de Lawler et al. Etendue par Vignier & al. 1995).

- $r_i \geq 0$: La date de disponibilité des données pour chaque tâche.
- Job-spl : Le découpage égal du job sur un nombre égal de machines.
- Pmtn : La prémption est autorisée. Un ordonnancement est dit préemptif quand une tâche peut voir son exécution suspendue au profit d'une autre tâche. En ordonnancement non préemptif, il faut attendre qu'une tâches termine (ou se bloque, dans le cas d'ordonnancement temps-réel, en attente d'une ressource) pour pouvoir passer à la tâche suivante.
- a_{ij} : Le lag minimal (minimum time lag). Un lag minimal a_{ij} représente la différence minimale (éventuellement négative) séparant le début de l'exécution de la tâche i sur la machine $j + 1$ de la fin de son exécution sur la machine j (par exemple, temps de séchage, de refroidissement, de transport, etc.).
- A_{ij} : Le lag maximal (maximum time lag). Un lag maximal A_{ij} représente la différence maximale (éventuellement négative) séparant le début de l'exécution de la tâche i sur la machine $j + 1$de la fin de son exécution sur la machine j.
- S_{sd} (respe. S_{nsd}) : temps de montage dépendant de la séquence « sequence dependent job setup times » (resp.temps de montage indépendant de la séquence « non sequence dependent job setup times »).
- R_{nsd} : temps de démontage indépendant de la séquence « non sequence dependent job removal times »).
- $b_{j,j+1}$: capacité de stockaeg limitée (limited buffer storage capacity) entre les machines j et $j+1$.
- $B_{j,j+\beta}$: nombre de ressources auxiliaires utilisables limité pendant les $\beta+1$ opérations successives d'une tâche.
- no-wait : aucune attente n'est autorisée entre deux opérations successives d'une tâche.
- j-pres : contrainte de précédence générale.
- prec : contrainte de précédence générale des opérations.
- in-tree : contrainte de précédence où toute opération peut avoir un ou plusieurs prédécesseurs, mais au plus successeur.
- out-tree : contrainte de précédence où toute opération ne doit avoir qu'un prédécesseur au plus, mais peut avoir un ou plusieurs successeurs.

Annexe II

ABC *Activity Based Costing :* L' ABC est une méthode de calcul du prix de revient basée sur le concept d'activité. Celles-ci consomment des ressources qui sont des sources de coût.

ABM *Activity Based Management* : L'ABM est l'utilisation des concepts de la méthode ABC dans le cadre du management d'un ensemble d'activités.

***Activité* (1) :** C'est une transformation (de matière, produit ou information, en général d'un flux) permettant de transformer des éléments d'entrée en éléments de sortie selon une règle reproductible. Cette transformation est supporté par un ensemble de ressources (humaines, techniques, etc.).

***Activité* (2) :** Ensemble de tâches homogènes caractéristiques d'un processus de réalisation de la chaîne de valeur et consommateur de ressources (par exemple, entretien des outillages pour un atelier). Au sein d'une unité organisationnelle analysée (filiale, usine, service, canal de distribution, etc.), il est possible de ne retenir que les activités principales (2 à 5). Plusieurs unités peuvent avoir le même type d'activité (par exemple, maintenance) ou participer à un même processus (par exemple, facturation, lancement d'un produit, etc.). Il existe plusieurs types d'activités. Par exemple, il est possible de distinguer entre les activités la hiérarchie suivante : les activités liées aux volume produits, celles relatives aux séries, celles concernant les unités de support et celles nécessaires à l'ensemble de l'entreprise. D'autres hiérarchies peuvent être établies en fonction des orientations stratégiques et des objectifs prioritaires d'une entreprise (flexibilité, satisfaction des clients, qualité, etc.).

***Activité* (3) :** Terme qui couvre beaucoup de sens plus ou moins proches. Utilisé en projet comme faculté d'exercer une action, et surtout par déviation, comme action d'une certaine ampleur, appartenant à un domaine technique déterminé, dont l'exécution peut être confiée à une personne physique ou morale compétente, action considérée comme unique dans le temps et l'espace et, en conséquence non répétitive. En planification, le terme « tâche » est souvent considéré comme synonyme d'activité.

***Activité* (4) :** Une activité constitue un faire mettant en œuvre un savoir-faire, elle peut être définie comme tout ce que l'on peut décrire par des verbes dans la vie de l'entreprise.

Activité/ Tâche : La tâche est un but donné dans des conditions déterminées. Elle indique ce qui est à faire, l'activité ce qui se fait. La notion de tâche véhicule avec elle l'idée de prescription, sinon d'obligation. La notion d'activité renvoie, elle, à ce qui est mis en jeu par le sujet pour exécuter ces prescription, pour remplir ces obligations.

***Amélioration continue* :** Effort permanent pour éliminer la cause première de dysfonctionnements ou de gaspillages par une suite de petites améliorations proposées et mises en œuvres par tout le personnel de l'entreprise.

Axe de progrès : L'une des directions (qualité, coût, délai, socle humain et social, etc.) de la démarche de progrès.

L'axe de progrès majeur, sera pour un processus, un système ou un sous système, la dimension sur laquelle porteront prioritairement les efforts.

Chaîne de valeur : La chaîne de valeur comprend les activités créatrices de valeur, c'est-à-dire les activités physique et technologiques par lesquelles une entreprise crée un produit qui possède une valeur pour ses clients. Ces activités sont de deux type : les activités principales et les activités de soutien.

Cohérence : Caractéristique d'ensembles d'idées, de faits, de systèmes, de sous systèmes ou d'éléments afin qu'ils forment un tout logique et que leurs actions ne soient pas en conflit.

Cohérence horizontale : La cohérence horizontale représente la coordination des différentes fonctions ou des différentes activités nécessaires à la production selon le cycle de vie du produit.

Notion de cohérence appliquée à des systèmes, des sous système ou des fonctions se situant sur un processus d'activités.

Cohérence verticale : Notion de cohérence appliquée à la structure hiérarchique d'une organisation à travers les différents niveaux décisionnels, du niveau stratégique au niveau opérationnel.

Coopération : Pour des systèmes, fonctions ou personnes, fait d'agir conjointement pour mener à bien une activité, en ayant un objectif commun.

Compétitivité : La compétitivité consiste à présenter, à un instant donné, la meilleure offre qui résulte de la conjonction de la productivité avec la performance commerciale et la performance technique des produits.

Aptitude d'une entreprise à donner à ses produits les avantages qui lui permettront de rester présente sur le marché et d'améliorer durablement ses positions.

Coût : Charge ou dépense supportée par un intervenant économique par suite de la production ou de l'utilisation d'un produit ou de l'ensemble des deux. Il ne faut pas confondre le coût et le prix. Ce dernier est égal au coût majoré de la marge de l'entreprise.

Coût d'acquisition : Il est obtenu en ajoutant au prix d'achat les frais accessoires ou charges directes liées à l'acquisition et à la mise en état d'utilisation du bien ainsi que les charges indirectes dans la mesure où elles peuvent être rattachées à cette acquisition.

Coût caché : Coût qui existe mais dont la réalité n'apparaît pas dans les évaluations comptables classiques. L'ensemble des coûts caché constitue « l'usine fantôme ».

Coût global : Coût relatif à l'ensemble de la vie d'un produit pour un usage donné.

COQ (coût d'obtention de la qualité) : Ce coût comprend les coûts de non-conformités (défaillances internes et externes) et les coûts de conformité (prévention et évaluation).

Cycle de vie d'un produit : Le cycle de vie d'un produit est le temps qui s'écoule entre l'idée du produit jusqu' à sa destruction en passant par son installation chez le client. Il comprend les étapes suivantes : marketing, conception, industrialisation, fabrication, livraison, maintenance et recyclage.

Dynamique de progrès : Attitude d'une société qui développe un ensemble d'actions lui permettant d'améliorer sa compétitivité.

Comportement et état d'esprit d'un ensemble de personnes voulant atteindre le même objectif de progrès et menant des actions en conséquence.

Effectivité : L'effectivité se prononce sur le tryptique objectifs / moyen / résultats. Selon une formulation de Jean–Louis Lemoigne[1], il s'agit alors de vérifier si « l'on fait effectivement ce que l'on veut faire ».

Efficacité (1) : L'efficacité est relative à l'utilisation des moyens pour obtenir des résultats donnés dans le cadre d'objectifs fixés.

Efficacité (2) : Adéquation des résultats obtenus aux objectifs recherchés.

Efficacité (3) : L'efficacité (technique, industrielle, commerciale, financière, économique) a un caractère global et stable dans le temps. Elle résulte de la compétitivité et de la productivité de l'entreprise. Elle traduit son aptitude à dégager des marges et à être économiquement viable. Elle représente le degré d'atteinte des objectifs. C'est « faire les bonnes choses ».

Efficience (1) : L'efficience élargit l'analyse en portant appréciation sur le couple moyens / résultats, sans pour autant remettre en cause les objectifs proprement dits. Granstetdt[2], la défini comme « le rapport entre l'effort et les moyens totaux déployés dans une activité d'une part, et l'utilité réelle que les gens en tirent sous la forme de valeur d'usage d'autre part ».

Efficience (2) : Adéquation des moyens mis œuvre aux résultats obtenus.

Efficience (3) : Mesure de la performance généralement exprimée en pourcentage dans l'exécution d'une tâche. L'efficience est le rapport entre les heures standards passées divisées par les heures travaillées réelles. L'efficience peut aussi représenter le rapport entre les quantités réalisées et les quantités prévues. L'efficience est une mesure du respect des standards préétablis. Sur une période donnée, elle peut être calculée par machine, opérateur, groupe de machines, département, etc. Elle représente le coût de l'atteinte des objectifs. C'est « faire les choses bien ».

Facteur-clé de succès : Les facteurs-clés de succès (FCS) traduisent la performance perçue et attendue de l'entreprise. Ce sont les éléments par rapport auxquels seront définis les objectifs globaux de succès. Il faut distinguer les FCS gagnants permettant d'augmenter les parts de marché (par exemple « réduire le délai d'élaboration des devis ») des FCS qualifiants permettant de se présenter sur un marché (par exemple « la qualité du produit »). Ils se traduiront par des indicateurs globaux de succès (ou indicateurs stratégiques) tels que *Time to market, Radio D'incertitude* ®, taux de service, etc.).

La détermination des objectifs se fait en rapport avec la situation concurrentielle.

Les IP stratégiques seront « perçus » (taux de service, délai de réponse, etc.) ou « internes » (« Ratio d'incertitude ® »).

Facteur-clé de performance : Eléments qualifiant les outputs caractéristiques de chaque processus, système ou sous système et par rapport auxquels seront définis les objectifs locaux de succès (par exemple, ratio de tension des flux, temps de réponse à une demande, etc.). Ils se déduisent des indicateurs (globaux) de succès par « analyse des contributions » et se traduisent par des indicateurs locaux de succès.

Facteur-clé de progrès : Parmi les variables d'actions, celles qui influencent de manière significative la contribution de chaque activité à l'atteinte de la performance globale de l'entreprise.

Ils résultent de l'analyse « cause-effets » des indicateurs de succès des processus.

Ils se traduiront par des indicateurs de progrès dont les variables d'actions pourront elles-mêmes devenir facteurs de progrès de « niveau inférieur », etc.

Fiabilité : Aptitude d'une entité à accomplir une fonction requise dans des conditions données et sur un horizon donné.

Flexibilité : Capacité des ressources à s'adapter en mix et / ou en volume pour permettre à l'entreprise de répondre à une sollicitation extérieure dans un temps donné.

Flux : Déplacement d'un ensemble d'éléments dans l'espace et dans le temps. En gestion industrielle, les trois grandes catégories d'éléments déplacés sont les décisions, les informations et les éléments physiques.

Gestion de production sur commande : Concerne des produits généralement coûteux, spéciaux et unitaires. Ces produits ne sont pas disponibles immédiatement et nécessitent un délai de livraison dépendant du produit à réaliser.

---

[1] J.L. Lemoigne, « L'évaluation de l'effectivité des systèmes complexes ». ECOSIP, Paris, Juin 1989.
[2] I. Grantedt, « L'impasse industrielle ». Le Seuil, Paris, 1980.

Gestion de production sur stock : Concerne surtout les produits standard à large consommation. La quantité de production est souvent par lot de taille importante et dans ce cas on parle de la production de masse.

Horizon : L'horizon est l'intervalle de temps qui caractérise la validité de l'ensemble des décisions pour un niveau décisionnel donné.

Intégration : Action d'organiser ou d'arranger un ensemble d'activités discrètes en un processus le plus continu possible, par la suppression de cloisons ou d'activités sans valeur ajoutée. L'intégration est une notion fondamentale où *l'optimum* à atteindre est celui de l'entreprise (et non celui de chacune de ses fonctions) et où la relation entre les différentes fonctions devient une relation de coopération entre les acteurs de ces fonctions. L'intégration doit permettre la convergence des objectifs de chacune des fonctions vers les objectifs globaux de l'entreprise.

Indicateur : Un indicateur est un outil de gestion complexe qui comprend un ensemble d'informations :

- sa propre définition ;
- sa raison d'être : l'objectif stratégique auquel il se rattache, la cible chiffrée et datée qui lui est impartie, éventuellement des références comparatives, par exemple le résultat d'un benchmarking ;
- la désignation d'un acteur chargé de le produire (celui qui accède le plus facilement aux informations requises) ;
- la désignation d'un acteur responsable du niveau de l'indicateur (celui qui maîtrise le mieux le levier d'action correspondant) ;
- la périodicité de production et de suivi de l'indicateur ;
- sa définition en extension : la formule et les conventions de calcul ;
- les sources d'information nécessaires à sa production (applications informatiques, bases de données, saisies manuelles) ;
- les modes de segmentation, pour décomposer une forme agrégée de l'indicateur en formes plus détaillées (par exemple : la segmentation géographique, décomposition par territoires, segmentation par types de marché, par lignes de produits, par centres de responsabilité, etc.);
- les modes de suivi (budgété, réel, écart budgété / réel, historique sur N mois, comparaison même période année antérieure, cumul depuis le début de l'année, etc.) ;
- le mode de présentation (chiffres, tableaux, graphiques, courbes, etc.) ;
- une liste de diffusion.

Indicateur de performance (1) : Un indicateur de performance est une donnée quantifiée qui mesure l'efficacité de tout ou partie d'un processus ou d'un système par rapport à une norme, un plan ou un objectif déterminé et accepté dans le cadre d'une stratégie d'entreprise.

Indicateur de performance (2) : Un indicateur de performance est une donnée quantifiée, qui mesure l'efficacité des variables de décision par rapport à l'atteinte de l'objectif défini au niveau de décision considéré, dans le cadre des objectifs globaux de l'entreprise.

Indicateur de pilotage : Un indicateur de pilotage sert à la propre gouverne de l'acteur qui le suit, pour l'aider à piloter son activité. L'indicateur de pilotage doit guider une action en cours, et n'a pas nécessairement vocation à remonter aux niveaux hiérarchiques supérieurs pour permettre un contrôle a posteriori. Pour la plupart, les indicateurs de pilotage ne doivent par remonter. En effet, si trop d'indicateurs remontent, les niveaux hiérarchiques supérieurs sont engorgés et perdent la vision de leurs propres objectifs. Les indicateurs de pilotage sont liés, soit au suivi d'actions en cours, soit à des points sur lesquels le responsable veut maintenir un état de vigilance en contrôlant régulièrement les résultats atteints. Il peut s'agir, selon le cas, d'un indicateur de suivi ou d'un indicateur de résultats.

Indicateur de résultat (1) : Un indicateur de résultat pour une action donnée, mesure le résultat final de l'action (degré de performance client atteint, degré de réalisation d'un objectif). Par exemple, le taux de défauts sur le produit fini est un indicateur de résultat pour la fabrication ; c'est évidemment une mesure a posteriori, un constat. Par définition, l'indicateur de résultat arrive trop tard pour l'action, puisqu'il permet de constater que l'on atteint ou non les objectifs : c'est un outil pour formaliser et contrôler les objectifs.

Indicateur de succès (ou de résultat) (2) : Un indicateur de succès (ou de résultat) est une donnée quantifiée reflétant la performance attendue ou constatée d'une activité ou d'un ensemble d'activités.

Indicateur de suivi : Un indicateur de suivi d'une action donnée jalonne l'action, en mesure la progression et permet de réagir (actions correctives) avant que le résultat soit consommé. Par exemple, les caractéristiques dimensionnelles d'échantillons prélevés par sondage régulièrement au cours de la fabrication permettent d'anticiper sur d'éventuels problèmes de qualité avant que ceux-ci soient consommés et mesurés en fin de parcours par un taux de défauts. Un indicateur de suivi doit révéler les évolutions tendancielles dans les processus et fournir une capacité d'anticipation ou de réaction à temps.

Mesure : Evaluation d'une grandeur par comparaison avec une autre grandeur de même espèce prise pour unité.

Moyen d'action : Ressource (temps, finances, méthodes, etc.) consacrée à l'action sur les variables d'actions.

Niveau de service : Qualité avec laquelle une entreprise assure une livraison conforme aux demandes du client en quantité, qualité et prix.

Note : souvent exprimée en pourcentage, cette mesure prend différentes formes selon qu'il s'agit de livraison sur stock ou à la commande.

Objectif (1) : C'est le résultat, la cible, que doit atteindre le système, le processus ou l'activité pilotée.

Objectif (2) : Un objectif traduit l'intention de passer de l'état de performance existant à l'état de performance souhaité pour le système physique piloté par le centre de décision.

Cet objectif doit s'exprimer sous la forme d'un verbe exprimant la variation souhaitée associée à un domaine de performance. Lorsque l'objectif est quantitatif, l'état de performance souhaité sera quantifié. Lorsque l'objectif est qualitatif, la variation souhaitée sera exprimée sous la forme d'une tendance.

Objectifs opérationnels : Ils sont liés à l'activation des ressources et des processus opérants.

Objectifs stratégiques : Ils concernent l'évolution de l'entreprise, ses orientations et son positionnement dans son environnement.

Objectifs tactiques : Ils concernant la préparation des activités industrielles, en déployant la performance par processus.

Performance (1) : Synonyme d'action (anglais : *to perform*, ancien français « performer », (Larousse).

1) Résultat obtenu par un athlète dans une épreuve, chiffre qui mesure ce résultat (Larousse).
2) Réussite remarquable, exploit (Larousse).
3) Résultat obtenu dans l'exécution d'une tâche (Larousse).
4) Ensemble des indications chiffrées caractérisant les possibilités optimales d'un matériel (Larousse).
5) Est performance dans l entreprise, tout ce qui, et seulement ce qui, contribue à améliorer le couple valeur- coût (a contrario, n'est pas forcément performance ce qui contribue à diminuer le coût ou à augmenter la valeur isolément).
6) Est la performance dans l'entreprise, tout ce qui, et seulement ce qui, contribue à atteindre les objectifs stratégiques.

Performance (2) : La performance représente le degré de réalisation des objectifs établis.

Résultat obtenu dans un domaine précis, par quelqu'un ou par une machine, un véhicule.

Performance externe :

La performance externe représente le niveau de performance perçu par le client.

Au niveau global, elle s'apparente à la compétitivité.

Performance globale : La performance globale représente le degré d'atteinte des objectifs globaux de l'entreprise.

Performance industrielle : C'est faire mieux que le « concurrent » (référence choisie pour la comparaison) sur le moyen et long terme, dans l'idéal sur un ensemble des paramètres définissant la performance, au minimum sur ceux des paramètres jugés être le plus significatifs de la satisfaction des clients.

Performance interne : La performance interne s'apparente à la productivité.

Pertinence : Adéquation des moyens mis en œuvre aux objectifs recherchés.

Période : La période est l'intervalle de temps au bout duquel il est nécessaire de remettre en cause les décisions élaborées sur l'horizon considéré.

Pilotage (1) : Ce terme est souvent assez mal compris, parce que vague. Il serait bon de le réserver aux activités décisionnelles indirectes et discontinues, par exemple lorsqu'un comité de pilotage (ou un supérieur hiérarchique) donne au chef de projet des instructions générales à traduire en décisions effectives.

Pilotage (2) : Piloter, c'est définir et mettre en œuvre des méthodes qui permettent d'apprendre, ensemble :

- à agir ensemble de manière performante ;
- à agir ensemble de manière de plus en plus performante.

Pilotage (3) : Piloter, c'est accomplir de manière continue deux fonctions complémentaires : déployer la stratégie en règles d'action opérationnelles (déploiement) et capitaliser les résultats et les enseignements de l'action pour enrichir la réflexion sur les objectifs (retour d'expérience).

Pilotage (4) : Le pilotage représente l'ensemble des activités décisionnelles relatives à la planification, à la programmation, à la coordination et au contrôle d'un système.

Proactivité (1): Capacité d'un système à anticiper ou à susciter une sollicitation extérieure, par exemple une demande du marché.

Proactivité (2) : (Qualifiée également par l'AFGI de réactivité de d'ordre 2). Elle a pour but de réagir en anticipant face à des défis futurs. Elle se situe dans une perspective de moyen-long terme et peut influer sur l'environnement de l'entreprise.

Proactivité (3) : La proactivité d'un système de production se caractérise par ses capacités d'anticipation (prévoir et/ou provoquer) les changements d'état, d'apprentissage et d'enrichissement des connaissances (pour améliorer sa réactivité), d'adaptation des règles de fonctionnement et par sa capacité de réorganisation reposant sur une architecture décentralisée et une délégation de responsabilité.

Processus (1) : Ensemble d'activités séquencées et finalisées par un objectif global et dont le résultat peut être matériel (produit) ou immatériel (information). Dans le cas d'informations, on parle également de « procédure ».

Processus (2): Ensemble d'activités liées en vue d'atteindre un objectif commun (par exemple, ensemble des activités nécessaires à la facturation d'un client, à la fabrication d'un téléviseur, à la conception d'un niveau produit, etc.). Les performances de ces activités liées sont souvent indépendantes. L'analyse par processus permet de mieux maîtriser une gestion transversale de l'entreprise (par exemple la gestion des projets, la gestion des commandes, etc.).

Processus (3): Il correspond à tout changement dans le temps de matière, d'énergie ou d'information.

Productivité (1) : Un concept visant à engager le minimum de ressources afin de réaliser les produits répondant juste au besoin. C'est une mesure globale de l'efficience de la production qui comprend deux facteurs : rendement (comment travaille une ressource) et utilisation (combien de temps elle est utilisée). La productivité est alors le produit du rendement par l'utilisation ou le *ratio* de la production réalisée sur le temps total prévu pour cette production.

Productivité (2), Compétitivité : La productivité peut donc s'apprécier pour une entreprise isolée alors que la compétitivité ne peut résulter que de la comparaison entre entreprises concurrentes.

La productivité et la compétitivité peuvent être bonnes à l'instant considéré mais c'est l'efficacité qui garanti sa pérennité.

Produit (1) : Ce qui est (ou sera) fourni à un utilisateur pour répondre à son besoin, Le produit peut être ici un matériel, un service ou toute combinaison des deux, un processus industriel ou administratif (procédé, logiciel, procédure).

Produit (2) : Les produits (matériels ou immatériels, biens ou services) apportent des réponses à des besoins de clients, qu'il s'agisse des besoins des clients directs (fonctionnalités des produits) ou des besoins plus généraux de la société (recyclabilité, non-pollution, par exemple).

Produits finis : Ce sont les produits qui ont atteint un stade définitif dans le cycle de production.

Projet : Ensemble d'actions à réaliser pour satisfaire un objectif défini, dans le cadre d'une mission précise, et pour réalisation desquelles on a identifié non seulement un début, mais aussi une fin. On distingue souvent les projets *ouvrage*, dont la finalité est d'obtenir un résultat considéré pour lui – même (ouvrage d'art, bâtiment, usine, navire, déménagement, etc.). et le projet *produit* dont la finalité est la mise au point d'un produit, qui fera par la suite l'objet d'une production répétitive, destinée à un marché (par exemple, automobile, électroménager, produit chimique ou pharmaceutique). Le premier est aussi appelé « projet d'ingénierie » ou « projet client » car il destiné à un client unique, et le second « projet de développement » ou « projet marché ».

Qualité : C'est « l'ensemble des propriété et caractéristique d'un produit ou d'un service qui lui confère l'aptitude à satisfaire des besoins exprimés ou implicite »

Réactivité (1) : Capacité d'un système à répondre, dans un temps donné à une sollicitation extérieure. En production, cette sollicitation correspond à la demande des clients.

Réactivité (2) : La réactivité se défini comme la capacité à réagir le plus rapidement possible et efficacement aux sollicitations internes ou externes en maximisant la flexibilité intrinsèque du système.

Réactivité (3) : Aptitude d'un système à retrouver un fonctionnement maîtrisé dans un temps requis, suite à une sollicitation ou à une perturbation. La réactivité se situe dans une perspective à court terme et a un caractère passif : elle n'influe généralement pas sur l'environnement de l'entreprise.

***Réactivité* :** La réactivité d'un système de production est définie comme l'aptitude à répondre (réagir) dans un temps requis aux changements de son environnement interne ou externe (aléa, situation nouvelle, perturbation, sollicitation, …) par rapport au régime (fonctionnement) permanent (stable).

Régulation : Mode de fonctionnement d'un système dans lequel la grandeur réglée tend à se rapprocher d'une grandeur de référence et examine l'aptitude du système à effacer les perturbations.

Reporting : C'est l'action d'informer les niveaux hiérarchiques supérieurs de la performance d'un système ou d'une activité à un niveau hiérarchique donné.

Ressource (1) : Tout moyen à disposition d'une entreprise pour la production et la livraison d'un produit ou d'un service.

Ressource (2) : Une ressource est un objet (par exemple une machine, un ordinateur, un lingot de métal) ou un service (par exemple le travail d'un ouvrier, une étude sous-traitée), porteur de valeur, consommé ou consommable par l'entreprise dans le cadre de l'un de ses processus. La consommation de la ressource est mesurable physiquement (heures, kilos) ou monétairement. La mesure monétaire d'une consommation de ressource est un coût.

Robustesse : La robustesse d'un système de production se définit par son aptitude à produire conformément aux résultats attendus. Cela suppose la garantie de l'obtention des performances souhaitées en présence d'incertitudes dans le système.

Stratégie d'entreprise : La stratégie d'entreprise est l'ensemble des décisions et des actions relatives aux choix des moyens et à l'articulation des ressources en vue d'atteindre les objectifs globaux de l'entreprise.

Système (1) : Ensemble d'éléments en interaction organisés en fonction d'un but et en relation avec un environnement.

Système (2) : Totalité organisée, faite d'éléments solidaires ne pouvant être définis que les uns par rapport aux autres en fonction de leur place dans cette totalité. Ensemble d'éléments en interaction dynamique organisé en fonction d'un but.

Système complexe (1) : Système que l'on tient pour irréductible à un modèle fini, aussi compliqué, stochastique sophistiqué que soit ce modèle, quelle que soit sa taille, le nombre de ses composants, l'intensité de leurs interactions.

Système complexe (2) : Par construction, système manifestant quelque forme d'autonomie : si ses comportement devaient être complètement dépendants d'intervention extérieures ou exogènes (sur lesquelles il n'exerce aucun contrôle), il ne seraient pas complexes, mais au contraire complètement prévisibles.

Système complexe (3) : Tout système complexe est constitué d'une grande variété d'élément possédant de fonctions spécialisées. Ces éléments sont organisés en niveaux hiérarchiques internes. Les différents niveaux, les différents éléments sont reliés par une grande variété de liaisons, se qui se traduit par une grande densité d'interconnexion.

Système de décision : Le système de décision est composé d'un système décisionnel périodique et d'un système décisionnel événementiel. Il est composé des moyens et des hommes qui contribuent à la prise de décisions pour conduire le système physique.

Système d'indicateurs de performance : Ensemble cohérent d'indicateurs de performance déployés et répartis uniformément sur l'ensemble des fonctions du cycle de vie du produit et sur l'ensemble des niveaux décisionnels.

Système d'information : Le système d'information est l'ensemble des ressources techniques dont l'objectif est le stockage, le traitement et la mise à disposition de toutes les informations nécessaires au fonctionnement du système physique.

Système pilotage : Le système de pilotage est l'ensemble des ressources humaines et techniques dont l'objectif est la conduite du système physique. Il est composé du système de décision et du système d'information.

Système physique : Le système physique est l'ensemble des ressources humaines et techniques dont l'objectif est de transformer des entrants (matière, information, etc) en produit ou prestations.

Tableau de bord (1) : Les indicateurs de pilotage sont regroupés en tableaux de bord, qui en assurent une présentation lisible et interprétable, avec une périodicité régulière adaptée aux besoins du pilotage. Chaque tableau de bord correspond à une unité de pilotage donnée (centre de responsabilité, processus, projet, fonction, produit, marché) sur laquelle ont été définis un schéma de responsabilité et une animation de gestion, en vue d'atteindre des objectifs de performance.

Tableau de bord (2) : Ensemble des indicateurs, à l'usage d'un responsable, nécessaires et suffisants au pilotage par celui-ci des actions et au *reporting* des résultats. Ce responsable peut-être une personne en position hiérarchique, un chef de projet, un animateur, une équipe constituée, etc.

Valeur (1) : La valeur représente ce pour quoi le client paye.

Valeur (2) : Jugement porté sur le produit par l'utilisateur sur la base de ses attentes et de ses motivations, exprimé par une grandeur qui croît lorsque, toutes les choses égales par ailleurs, la satisfaction du besoin de l'utilisateur augmentant et /ou que la dépense afférente au produit diminue. Ce jugement résulte d'une observation objective, le jugement porté par l'utilisateur en fonction de l'utilité qu'il retire du produit (la valeur d'usage) et d'une évaluation subjective, la considération affective que l'utilisateur attache au produit (valeur d'estime) . Lorsqu'il n'est pas ou pas encore possible de connaître les attentes et motivations de l'utilisateur même, elles sont exprimées par ceux qui ont mission de le représenter. Cette grandeur implique une relation entre satisfaction et dépense. Elle s'apparente au « rapport qualité/prix ». Cette relation traduit le caractère à la fois fonctionnel et économique de la démarche. Par « dépense afférente au produit », on peut entendre soit le coût d'acquisition (prix + dépenses annexes) ou le coût global, considéré par l'acheteur ou l'utilisateur, soit le coût considéré par l'industriel.

Valeur ajoutée : Augmentation de la valeur d'un produit pendant son processus de production, depuis le moment où ses composants sont réceptionnées jusqu' à sa mise à disposition au client.

Valeur ajoutée client : Qualité supplémentaire d'un produit reconnue et appréciée par le client.

« Ce pourquoi le client paye ».

Variable d'action : Ce sont les variables sur lesquelles les acteurs du système agissent pour atteindre les objectifs.

Variable de décision : Une variable de décision est une entité qui agit sur une activité du système conduit afin de faire évoluer la performance de l'activité dans le sens du (ou des) objectif (s). La variable de décision a toujours une latitude finie qui dépend des contraintes imposées au décideur.

Annexe III

Les tâches programmables

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
r_i	0	1	2	3	4	6	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
d_i	2	4	5	8	9	10	14	20	15	18	15	17	19	20	21	20	22	25	24	25	30

i	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
r_i	23	26	27	29	30	31	35	37	39	40	41	45	48	52	57	59	63	67	68	70	75
d_i	28	29	32	34	60	45	72	42	46	47	50	53	58	61	82	69	74	72	86	75	80

i	43	44	45	46	47	48	49	50	51	52	53	54	55
r_i	78	80	82	84	86	87	88	90	91	94	96	98	99
d_i	81	85	88	89	90	94	95	92	97	98	99	100	100

Tableau 1 : Les caractéristiques des tâches programmables.

Les tâches aléatoires

j	1	2	3	4	5	6	7
rj	19.6	39	58.21	77.23	96.06	114.7	133.16

Tableau 2 : Les caractéristiques des tâches aléatoires pour s = 0.05.

.

j	1	2	3	4	5	6	7	8	9	10	11	12
r_j	9.9	19.75	29.55	39.3	49	58.65	68.25	77.8	87.31	96.77	106.18	115.55

j	13	14	15	16	17	18	19	20	21	22	23
r_j	152.55	179.81	206.67	224.39	276.44	293.41	301.84	335.15	359.69	383.87	407.69

Tableau 3 : Les caractéristiques des tâches aléatoires pour s = 0.1.

j	1	2	3	4	5	6	7	8	9	10	11	12
r_j	4.97	9.93	14.88	29.65	34.55	49.16	58.85	63.67	78.07	87.6	92.35	101.82

j	13	14	15	16	17	18	19	20	21	22	23
r_j	153.05	180.4	211.8	242.66	260.04	290.07	315.37	332.07	368.97	393.12	405.06

Tableau 3 : Les caractéristiques des tâches aléatoires pour s = 0.2.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
r_j	1,996	3,99	5,98	7,97	9,96	11,94	13,93	15,91	17,89	19,87	21,84	23,82	25,79	27,76	29,73	31,69	33,66	35,62	37,58	41,5	43,45

j	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
r_j	45,4	49,3	51,25	55,14	57,08	60,95	64,82	66,75	68,68	70,61	74,46	78,3	80,22	85,96	89,78	91,68	97,39	99,29	101,19

Tableau 4 : Les caractéristiques des tâches aléatoires pour s = 0.5.

.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
r_j	1,66	3,32	4,98	6,64	8,3	9,96	11,61	13,27	14,92	16,57	18,22	19,87	23,16	24,81	28,09	33,01	36,28	39,55	42,81	46,06	49,31

j	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
r_j	50,94	54,18	57,41	59,03	62,26	65,48	68,7	71,91	75,12	79,92	81,51	84,7	87,89	91,07	95,83	97,41	99	100,58	102,16

Tableau 5 : Les caractéristiques des tâches aléatoires pour s = 0.6.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
r_j	1,42	2,85	4,27	5,7	7,12	8,54	9,96	11,38	14,21	17,05	18,46	19,87	22,7	25,52	28,33	32,55	35,35	38,16	40,95	43,75	47,93

j	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
r_j	49,32	52,1	56,27	57,66	61,81	63,19	65,95	72,84	75,59	76,96	82,44	83,81	86,54	93,36	96,07	98,79	100,14	101,5	102,85

Tableau 6 : Les caractéristiques des tâches aléatoires pour s = 0.7.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
r_j	1,24	2,49	3,74	4,98	6,23	7,47	8,72	9,96	12,44	14,92	16,16	18,64	19,88	21,11	24,82	27,28	30,97	34,66	39,56	40,78	44,45

j	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
r_j	48,11	51,76	54,19	56,62	60,26	66,31	69,93	71,13	73,54	77,14	83,14	85,53	89,11	91,5	93,88	97,44	101,01	105,74	110,47

Tableau 7 : Les caractéristiques des tâches aléatoires pour s = 0.8.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
r_j	1,1	2,21	3,32	4,43	5,54	6,65	7,75	8,86	9,96	11,07	14,38	16,58	18,78	20,98	25,37	29,75	31,93	34,12	37,39	40,65	43,91

j	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
r_j	45	48,25	50,42	54,74	56,9	57,98	60,13	65,51	68,73	70,87	75,15	78,35	80,48	84,74	88,99	90,05	93,23	97,45	102,73

Tableau 8 : Les caractéristiques des tâches aléatoires pour s = 0.9.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
r_j	1	1.99	2.99	3.99	4.99	5.98	6.98	7.97	8.97	9.96	12.94	14.93	18.89	19.88	21.86	24.82	27.78	30.73	33.68	38.59	42.51

j	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
r_j	45.44	48.36	51.29	55.18	59.06	61	64.87	69.7	73.55	76.44	79.32	82.2	85.07	89.85	92.71	97.46	99.36	100.31	110.73

Tableau 9 : Les caractéristiques des tâches aléatoires pour s = 1.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
r_j	0.66	1.99	2.66	3.32	4.65	5.23	6.65	7.31	8.64	9.97	10.63	11.29	16.58	19.23	21.2	24.5	29.1	31.72	35	38.92	44.14

j	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
r_j	46.1	49.35	52.6	57.13	58.43	62.31	66.17	67.46	70.04	74.53	78.38	80.3	83.49	88.59	90.5	95.59	98.75	100.02	105.71

Tableau 10 : Les caractéristiques des tâches aléatoires pour s = 1.5.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
r_j	0,5	1	1,99	2,99	3,49	4,99	5,98	6,48	7,98	8,97	9,47	10,96	12,95	16,92	18,9	20,88	25,32	28,78	32,22	34,19	39,09

j	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
r_j	40,56	43,5	47,41	50,33	54,71	57,63	60,05	65,38	68,76	71,65	75,05	77,91	81,75	84,14	87,97	91,31	94,65	96,07	105,57

Tableau 11 : Les caractéristiques des tâches aléatoires pour s = 2.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
r_j	1,19	2,39	3,59	4,79	5,98	6,78	7,58	8,77	9,97	10,76	13,94	17,91	22,26	25,82	29,76	30,55	35,27	39,98	41,55	45,07	49,75

j	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
r_j	50,92	54,42	59,86	61,8	66,06	69,92	72,24	77,62	79,62	81,85	83	85,68	91,03	95,6	97,51	99,79	102,45	105,1	110,4

Tableau 12 : Les caractéristiques des tâches aléatoires pour s = 2.5.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
r_j	0,66	1,99	2,66	3,99	4,65	5,98	6,31	7,64	8,97	9,3	10,96	14,6	19,89	21,54	26,8	28.12	32,06	37,3	39,59	40,24	44,8

j	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
r_j	48,71	51,63	55,2	58,76	62,64	67,8	68,44	73,91	75,51	78,07	80,95	84,47	89,89	92,75	96,56	97,83	100,99	104,15	112,66

Tableau 13 : Les caractéristiques des tâches aléatoires pour s = 3.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
r_j	0,86	1,14	2	3,14	4,57	5,7	6,27	7,41	8,55	9,97	11,11	12,81	18,76	21,03	24,7	27,52	31,74	34,55	38,75	42,95	43,51

j	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
r_j	47,7	50,21	53,54	56,6	60,2	63,8	65,74	70,15	74,01	76,76	82,24	86,89	88,53	90,17	92,62	98,06	101,05	102,4	103,22

Tableau 14 : Les caractéristiques des tâches aléatoires pour s = 3.5.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
r_j	0,75	1,99	2,49	3,99	4,24	5,99	6,73	7,48	8,72	9,97	10,22	14,69	19,4	23,6	26,32	29,52	30.5	36,4	37,88	40,08	42,77

j	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
r_j	45,22	52,78	56,91	58,12	63,45	64,66	67,81	71,42	73,35	77,92	83,2	84,4	87,75	93	95,62	99,89	103,45	106,77	109.37

Tableau 15 : Les caractéristiques des tâches aléatoires pour s = 4.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
r_j	0,44	1,1	2,44	3,1	4,65	5,54	6,87	7,75	8,86	9,3	10,19	13,5	18,79	20,11	25,6	28,01	32,83	35,88	39,15	41,98	42,64

j	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
r_j	46,33	51,75	53,69	57,58	61,46	64,26	68,99	72,63	75,84	76,91	80,53	83,94	86,5	91,8	93,5	96,25	98,79	101,11	103,64

Tableau 16 : Les caractéristiques des tâches aléatoires pour s = 4.5.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
r_j	0,8	1,79	2,59	3,79	4,39	5,99	6,98	7,18	8,77	9,37	12,55	15,33	17,52	22,66	24,05	26,22	33,51	37,43	38,81	43,31	44,68

j	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
r_j	47,42	53,27	55,6	59,87	64,52	65,88	69,93	71,48	74,17	78,98	81,67	83,2	85,5	89,52	90,09	95,81	97,71	101.7	112.3

Tableau 17 : Les caractéristiques des tâches aléatoires pour s = 5.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
r_j	0.77	2.55	3.99	5.1	6.98	7.09	8.75	9.2	10.08	11.3	14.5	18.9	20	24.4	25.5	29	32.5	36	38.4	40.9	45.03

j	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
r_j	49.7	51	54.57	58.02	60.5	63.3	66.74	72.1	75.2	79.05	81.5	83.95	88.1	90.01	94.36	97.95	100.7	101.96	103.23

Tableau 18 : Les caractéristiques des tâches aléatoires pour s = 9.

Annexe IV

Les tâches programmables

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
r_i	0	1	2	4	7	7	8	10	11	15	18	22	25	29	30	33	33	38	42	48	51
d_i	3	8	7	10	12	19	32	25	20	20	24	30	40	42	35	50	39	65	60	57	55

i	22	23	24	25	26	27	28	29	30
r_i	55	63	64	73	76	80	87	97	99
d_i	80	92	70	79	94	85	90	99	100

Tableau 1 : Les caractéristiques des tâches programmables.

Les tâches aléatoires

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
r_j	0	1	3	5	9	16	19	21	23	30	40	43	52	64	68	72	79	83	89	98

Tableau 2 : Les caractéristiques des tâches aléatoires pour le premier cas.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
r_j	0	1	3	5	9	16	19	21	23	30	40	43	52	64	68	72	79	83	89	98
p_j	1	1.5	2	2.5	1	1.5	2	2.5	1	1.5	2	2.5	1	1.5	2	2.5	1	1.5	2	2.5

Tableau 2 : Les caractéristiques des tâches aléatoires pour le second cas.

Références

Bibliographiques

Ouvrages

[Afg 92] Association Française de Gestion Industrielle, Evaluer pour évoluer, les indicateurs de performance au service du pilotage industriel. Ouvrage collectif AFGI, 1992.

[Bel 57] R. Bellman, Dynamic Programming. Princeton University Press. Princeton, 1957.

[Ben 98] J. Bénassy, La gestion de production. Ed. Hermès, 1998.

[Bot 99] G. Botta, Gestion de Production. INSA, Département de Génie Productique. 1998-1999.

[Car 88] J. Carlier et P. Chrétienne, Problèmes d'Ordonnancement : Modélisation, Complexité et Algorithmes. Paris, Masson, 1988.

[Che 77] A. Chevalier, La programmation dynamique. Dunod, 1977.

[Chu 96] C. Chu et J.M. Proth, L'Ordonnancement et ses applications, Masson, 1996.

[Con 67] R. W. Conway, W.L. Maxwell and L.W. Miller, Theory of Scheduling. Addision-Wesley, Reading, MA, 1967.

[Dup 97 a] L. Dupont, Gestion Industrielle : concepts et outils. E.N.S.G.I. 1997.

[Esq 99] P. Esquirol et P. Lopez, L'Orordonnancement. Economica, 1999.

[Gia 88] V. Giard, Gestion de production. Economica, 1988.

[Gra 02] A. Gratacap, La Gestion de la Production. Dunod, 2002.

[Jav 97] G. Javel, Organisation et Gestion de la production. Masson, 1997.

[Kar 72] R. Karp, Complexity of computer Computations. R.E. Miller, J.W. Thatcher (eds.), Plenum Press, New-York, 1972.

[Kel 98] W.D. Kelton, R.P. Sadowski and D.A. Sadowski, Simulation With ARENA. Mac Graw-Hill, 1998.

[Lam 91] P. Lamy, Ordonnancement et Gestion de la Production. Paris, Hermès, 1991.

[Lop 01] P. Lopez et F. Roubellat, Ordonnancement de la production. Paris, Hermès, 2001.

[Mit 96] M. Mitchell, In Introduction to Genetic Algorithms. A Brad For Book, London, 1996.

[Pin 95] M. Pinedo, Scheduling: Theory, Algorithms and Systems. Prentice Hall, 1995.

[Pri 86] A.A.B. Pritsker, Introduction to simulation and SLAM II. Halsted Press, New York, NY, 3rd Edition, 1986.

[Rin 76] A.H.G. Rinnooy Kan, Scheduling Problems: Classification, Complexity and Computations. Martinus Nijhoff, The Hague, 1976.

[Woo 65] J. Woodward, Industrialisation Organisation: theory and practice. Oxford, University Press, 1965.

Mémoires et Thèses

[Art 97] C. Artigues, Ordonnancement en temps-réel d'ateliers avec temps de préparation des ressources. Thèse de doctorat, Université Paul Sabatier, Toulouse, 1997.

[Bak 96] M. Bakalem, Modélisation et simulation orientées objet des systèmes manufacturiers. Thèse de doctorat en Électronique –Électrotechnique- Automatique, Université de Savoie, juillet 1996.

[Ben 87] E. Bensana, Utilisation des techniques d'intelligence artificielle pour l'ordonnancement d'atelier. PhD thesis, Thèse de l'ENSAE, Toulouse, 1987.

[Cas 04] P. Castagna ; Contribution à la modélisation, la simulation et la commande de systèmes de production et de transitique ; HDR, Université de Nantes, France, 2004

[Fox 83] M.S. Fox, Constraint-directed search: a case study in job shop scheduling. Ph.D. Thesis, Carnegie Mellon University, 1983.

[Gal 94] T. Galinho, Algorithme heuristique de placement pour l'ordonnancement : Etude comparative et recherche d'expertise sur les stratégies de contrôle. PhD thesis, Université de Rouen, 1994.

[Gaz 03] S. H. Gazoby, Planification des tâches dans un environnement d'ateliers à flux tirés. Mémoire de Magister, EMP, Algérie, 2003.

[Hab 01] G. Habchi, Conceptualisation et Modélisation pour la simulation des systèmes de production. HDR, LLP/CESALP – ESIA – Université de Savoie, 2001.

[Hen 99] H. Hentous, Contribution au pilotage des systèmes de production de type Job Shop. Thèse de Doctorat, INSA Lyon, 1999.

[Leb 98] L. Leboucher, Algorithmique et Modélisation pour la Qualité de Service des Systèmes Répartis Temps-Réel. Thèse de doctorat, Ecole Nationale Supérieure des Télécommunications de Paris, 1998.

[Let 01] A. Letouzey, Ordonnancement interactif basé sur des indicateurs : Applications à la gestion de commandes incertaines et à l'affectation des opérateurs. Ecole Nationale d'Ingénieurs de Tarbes, 2001.

[Meb 95] N. Merbaki, Une approche d'ordonnancement temps-réel basée sur la sélection dynamique de règles de priorité. Thèse de doctorat, Université Claude Bernard Lyon I, Lyon, France, 1995.

[Mok 83] A.K. Mok, Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environnement. PhD Thesis, Departement of electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, 1983.

[Our 01] S. Ourari, Ordonnancement temps-réel en présence d'évènements aléatoires. Mémoire de Magister, EMP, Algérie, 2001.

[Riv 98] N. Rivierre, Ordonnancement temps-réel centralisé, les cas préemptifs et non préemptifs. Thèse de doctorat, Université de Versailles Saint Quentin, 1998.

[Sac 02] A. Saci, Ordonnancement de tâches dans un environnement de machines en parallèle. Mémoire de Magister, EMP, Algérie, 2002.

[Ter 04] F. Tercinet, Méthodes arborescentes pour la résolution des problèmes d'ordonnancement, conception d'un outil d'aide au développement. Thèse de doctorat, Université François Rabelais Tours, France, 2004.

Articles et Rapports

[Afn 88] AFNOR, Concepts fondamentaux de la gestion de production. Technical Report X50-310. Agence Française de NORmalisation, 1988.

[All 99] A. Allahversi, J.N.D Gupta and T. Aldowaisan, A review of scheduling research involving setup considerations, Omega. International Journal of Management Science, (27) p. 219-239, 1999.

[And 93] D.P.Anderson, MetaScheduling for Distributed Continuous Media. ACM Trans. on Computer Systems, 11(3):226-252, 1993.

[Axs 81] S. Axsäter, Aggregation of product data for hierarchical production planning. Operations Research, 29: 754-756, 1981.

[Bab 04] P. Babu, L. Péridy, E. Pinson et D. Rivreau, Borne inférieure et méthode arborescente pour la minimisation du nombre pondéré de tâches en retard sur machines en parallèle. MOSIM '2004, Nantes, France, 2004.

[Bar 90] S.K. Baruah, A.K. Mok and L.E. Rosier, Preemptively scheduling hard real-time sporadic tasks on one processor. Pages 182-190. 11th Real-Time System Symposium, 1990.

[Bla 82] J.H. Blackstone, J.D. T.Phillips and G.L. Hogg, A state-of-the-art survey of scheduling rules for manufacturing job-shop operations. International Journal of Production research, Vol.20, n°1, 1982.

[Bla 76] J. Blazewicz, Scheduling Dependant Tasks with Different Arrival Times to Meet Deadlines. In. Gelende. H. Beilner (eds), Modeling and Performance Evaluation of Computer Systems, Amsterdam, North-Holland, 1976.

[Blu 94] R. D. Blumofe and C. E. Leiserson, Scheduling multithreaded computations by work stealing. Proc. of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, pp. 356-368, 1994.

[Bru 01] P. Brucker, J. L. Hurink and S. Knust, A Polynomial algorithm for $P|p_i=1,r_i,\text{outtree}|SC_i$ and $P|p_i=1,r_i,\text{outtree,pmtn}|SC_i$. University of Twente, University of Technical and Social Sciences, Faculty of Mathematical Sciences. ISSN 0169-2690, 2001.

[Buc 93] M.C. Buchanan and P.T. Zellweger. Automatically Generating Consistent Schedule For Multimedia Applications. Multimedia Systems Journal, 1(2): 55-67, 1993.

[Bux 89] G. Buxey, Production Scheduling : Practice and theory. European Journal of operational Research, 38: 16-31, 1989.

[Car 94] C. Cardeira and Z. Mammeri, Ordonnancement de tâches dans les systèmes temps-réel et répartis. APII, 28(4):353-384, 1994.

[Car 82] J. Carlier et P. Chrétienne. Les problèmes d'ordonnancement : un domaine très ouvert. RAIRO, Recherche opérationnelle / Operations Research, pp.42-47, 1982.

[Car 87] J. Carlier, Scheduling jobs with release dates and tails on identical machines to minimize the makespan. European Journal of Operational Research, 29: 298–306, 1987.

[Car 98] J. Carlier and E. Pinson, Jackson's pseudo preemptive schedule for the $Pm|r_j,q_j|C_{max}$ scheduling problem. Annals of Operations Research, 83:41–58, 1998.

[Che 90] H. Chetto, M. Silly and T. Bouchentouf, Dynamic Scheduling of Real-time Tasks under Precedence Constraints. Real-Time Systems, The International Journal of Time-Critical Computing Systems, 2(3):181-194, 1990.

[Cof 72] E. Coffman and R. Graham, Optimal scheduling for two processors systems. Acta Informatica 1, pp. 200-213, 1972.

[Cof 76] E. Jr Coffman and R. A. Sethi, generalized bound on LPT sequencing. RAIRO-Informatique, n°10, pp.17-25, 1976.

[Cof 78] E. Coffman, M. Garey and D. Johnson, An application of bin-packing to multi-processor scheduling. SIAM Journal of Coputing 7, pp. 1-16, 1978.

[Cot 98] F. Cottet, M. Courtes et M. Holle, Traitement de la gigue temporelle pour les ordonnancements temps-réel par échéance. Pages 63-77. Real Time Systems, Paris, 1998.

[Cou 97] G. Coulson and A. Mauthe, Scheduling and Admission Testing for Jitter Constrained Periodic Threads. ACM Multimedia Systems Journal, 5(5):337-346, 1997.

[Dan 95] S. P. Dandamundi and P. S. P. Cheng, A Hierarchical Task Queue Organization for Shared-Memory Multiprocessor Systems. IEEE Transaction on Parallel and Distributed Systems, Vol. 6, No. 1, pp 1-16, 1995.

[Dav 94] L. Davis & G. Williams, Evaluating and selecting simulation software using the analytic hierarchy process, Integrated Manufacturing Systems, Vol. 5, No.1, pp.23-32, 1994.

[Der 01] A. Derbala, Minimizing schedule length for Makespan criteria for parallel processor scheduling. Proceedings CIP'2001, Conférence Internationale sur la Productique, Alger, Algérie, 2001.

[Dra 98] G. Draghici, N. Brinzei et I. Filipas, La modélisation et la simulation en vue de la conduite des systèmes de production. Les Cahiers des Enseignements Francophones en Roumanie, 1998.

[Dup 97 b] L. Dupont, Y. Crama et G. Finke, Recherche opérationnelle et gestion de la production. Article préparé pour publication dans la revue: Nouvelles de la science et des technologies, 1997.

[Dur 04] C. Duron, M.A. Ould Louly et J. M. Proth, Problème d'ordonnancement à une machine : insertion d'une tâche en temps-réel. MOSIM'04, Nantes, France, 2004.

[Far 90] F. Farhoodi, A knowledge-based approach to dynamic job-shop scheduling. International Journal of Computer Integrated Manufacturing, vol.3, n°2, pp. 84-95, 1990.

[Gar 79] M.R. Garey and D. S. Johnson, Computers and intractability : a guide to the theory of NP-Completeness. W.H. Freeman and Company, New York, 1979.

[Ger 87] S.B. Gershwin, A hierarchical framework for discrete event scheduling in manufacturing systems. In IIASA Worshop on discrete Evenet Systems: Models and Applications, Sopron, Hungary, 1987.

[Gha 02] A. Gharbi and M. Haouari, Minimizing makespan on parallel machines subject to release dates and delivery times. Journal of Scheduling, 5:329–355, 2002.

[Gha 04] A. Gharbi and M. Haouari, Optimal parallel machines scheduling with initial and final availability constraints. Proceedings of the Ninth International Workshop on Project Management and Scheduling PMS, p. 218–21, 2004.

[Gha 05] A. Gharbi and M. Haouari, Optimal parallel machines scheduling with availability constraints. Discrete Applied Mathematics,148:63–87, 2005.

[Gla 94] C.A. Glass, C. Potts and P. Shade, Unrelated parallel machines scheduling using local search. Mathl. Comput. Modelling 20,2, pp.41-52, 1994.

[Glo 86] F. Glover, Future paths for integer programming and links to artificial intelligence. Computers and Operations Research 13: 533-549, 1986.

[Glo 89] F. Glover, Tabu search, part 1. Orsa Journal on Computing, vol.1, n°3, pp. 190-206, 1989.

[Glo 90] F. Glover, Tabu search, part 2. Orsa Journal on Computing, vol.2, n°1, pp. 4-32, 1990.

[Got 93] Groupe d'Ordonnancement THéorique et Appliqué, Les problèmes d'ordonnancement, R.A.I.R.O. Recherche opérationnelle/Operational Research, vol.27, n°1, pp. 77-150, 1993.

[Gra 98] B. Grabot, Objective satisfaction assessment using neural nets for balancing multiple objectives. International Journal of Production Research, vol.36, n°9, pp. 2377-2395, 1998.

[Gra 66] R.L. Graham, Bounds for certain multiprocessing anomalies. Bell System Technology Journal, n°45, pp.1563-1581, 1966.

[Gus 84] D. Gusfield, Bounds for naive multiple machine scheduling with release times and deadlines. Journal of Algorithms, 5: 1–6, 1984.

[Ha 97] S. Ha and E. A. Lee, Compile-Time Scheduling of Dynamic Constructs in Dataflow Program Graphs. IEEE Transactions on Computers, Vol 46, No 7, pp. 768-778, 1997.

[Ham 95] B. Hamidzadeh and D. L. Lilja, Dynamic Scheduling Strategies for Shared-Memory Multiprocessors. Proceedings of International Conference on Distributed Computing Systems, pp. 208-215, 1996.

[Han 96] C. C. Han, K. J. Lin, and C. J. Hou, Distance-constrained Scheduling and Its Applications to Real-time Systems. IEEE Transactions on computers, 45(7):814-826, 1996.

[Han 86] P. Hansen, The steepest ascent mildest descent heuristic for combinatorial programming. Congress on Numerical Methods in Combinatorial Optimisation, Capri, Italie, 1986.

[Hao 99] J.K. Hao, P. Galinier et M. Habib, Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. Revue de l'intelligence artificielle, 1999.

[Hao 04] M. Haouari and A. Gharbi, Lower Bounds for Scheduling on Identical Parallel

Machines with Heads and Tails. Annals of Operations Research 129, 187–204, 2004.

[Hen 97] H. Hentous and A. Guinet, A Constraint hybrid Flow Shop problem. Workshop on Production and Planning Control Proceedings, Mons, Belgique, pp. 303-306, 1997.

[Her 97] J.F. Hermant, L. Leboucher and N. Rivierre, Real-time Fixed and Dynamic Priority Driven Scheduling Algorithms : Theory and Experience. Rapport de recherche de l'INRIA numéro 3081, 1997.

[Hu 61] T. Hu, Parallel sequencing and assembly line problems. Operations Research 9, pp. 841-848, 1961.

[Iba 77] O. Ibarra and C. Kim, Heuristic algorithms for scheduling independent tasks on nonidentical processors. J. Journal of Assoc. Comput. Machinery 24,2, pp. 280-289, 1977.

[Jac 55] J.R. Jackson, Scheduling a production line to minimize maximum tardiness. Technical report, Management Science Research Project, University of California, Los Angeles, 1955.

[Jef 91] K. Jeffay, D.L. Stone, and D.E.Poirier, YARTOS: Kernel support for efficient, predictable real-time systems. Proc. joint Eight IEEE Workshop on Real-Time Operating Systems and Software and IFAC/IFIP Workshop on Real-Time Programming, Atlanta. Real-Time Systems Newsletter, 7(4):7-12, 1991.

[Jef 99] K. Jeffay and D. Bennett, A Rate-Based Execution Abstraction For Multimedia Computing. In Lectures Notes in Computing Science, T. D. C. Little and R. Gusella, Springer-Verlag, Heidelberg, 1018: pp. 64-75, 1995.

[Joh 54] S.M. Johnson, Optimal two and three stage production schedules included. Naval Research Logistics Quarterly, 1: 61-68, 1954.

[Joh 93] T. Johnson, T. A. Davis and S. M. Hadfield, A Concurrent Dynamic Task Graph. International Conference on Parallel Processing, Parallel Computing Vol 22, No 2, pp. 327-333, 1996.

[Kim 88] S. J. Kim and J. C. Browne, A General Approach to Mapping of Parallel Computations Upon Multiprocessor Architectures. IEEE Conf. on Parallel Processing, Vol III, pp. 1-8, 1988.

[Kir 83] S. Kirkpartick, C.D. Gelatt and P.M. Vecchi, Optimization by simulated annealing. Science 220, pp. 671-680, 1983.

[Law 93] E. Lawler, J. Lenstra, A. Rinnooy Kan and D. B. Schmoys, Logistics of production and inventory - Handbooks in OR and MS. S.C. Graves & al. (eds.), Elsevier Science Publishers, ch. Sequencing and scheduling : algorithms and complexity, pp. 445-522, 1993.

[Leh 90] J. P. Lehoczky, Fixed priority scheduling of periodic task sets with arbitrary deadlines. pp. 201-209, In Proc. 11th IEEE Real Time Systems Symposium, Lake Buena Vista, 1990.

[Le M 74] J.L. Le Moigne, Les systèmes de décision dans les organisations, Presses universitaires de France, 1974.

[Le M 77] J.L. Le Moigne, La théorie du système en général. Théorie de la modélisation, Presses universitaires de France, 1977.

[Len 77] J.K. Lenstra, A.H.G. Rinnooy Kan and P. Brucker, Complexity of machine scheduling problems. Annals of Discrete Mathematics, 1: pp. 343–362, 1977.

[Len 78] J. Lenstra and A. Rinnoy Kan, Complexity of scheduling under precedence constraints. Operations Research 26, pp. 22-35, 1978.

[Len 90] J. Lenstra, D. Schmoys and E. Tardos, Approximation algorithms for scheduling unrelated parallel machines. Mathematical Programming 46, pp. 259-271, 1990.

[Leu 80] J. Y. T. Leug and M. L. Merril, A note on preemptive scheduling of periodic real time tasks. Information Processing Letters, 11(3): pp.115-118, 1980.

[Lin 87] F.C.H. Lin and R. M. Keller, The Gradiant Model Load Balancing Method. IEEE Transactions on Software Engineering, Vol 13, pp. 32-38, 1987.

[Liu 73] C. L. Liu and J. W. Layland, Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environnment. Journal of the Association for Computing Machinery, 20(1):46-61, 1973.

[Mc N 59] R. Mc Naughton, Scheduling with deadlines and loss functions. Management Science 6, 1-12, 1959.

[Met 53] N.A. Metropolis, A. Rosenbluth, A. Teller and E. Teller, Equation of state calculations by fast computing machines. Journal of Chemestry and Physics, vol.6, n°21, 1953.

[Mok 04] E. Mokotoff, An exact algorithm for the identical parallel machine scheduling problem. European Journal of Operational Research, 2004.

[Mon 98] D. Montana, G. Bidwell and S. Moore, Using genetic algorithm for complex real time scheduling applications. IEEE, 1998.

[Mun 70] R. Muntz and E. Coffman, Preemptive scheduling or real-time tasks on multiprocessor systems. J. Assoc. Comput. Mach 17, 324-338, 1970.

[Pan 77] S.S. Panwalker and W. Iskander, A survey of scheduling rules. Operations Research, vol.9, pp. 45-61, 1977.

[Pap 82] C. H. Papadimitriou and P. C. Kanellakis, Flow Shop scheduling with limited temporary storage. Journal of the Association for Computing machinery, 27, pp. 533-554, 1980.

[Par 97] G. L. Park, B. Shirazi, J. Marquis and H. Choo, Decisive Path Scheduling: A New List Scheduling Method. International Conference on Parallel Processing, pp. 472-480, 1997.

[Per 03] L. Péridy, Thecniques d'optimisation combinatoire. Personnal communication 2003.

[Pie 97] H. Pierreval and N. Mebarki, Dynamic selection of dispatching rules for manufacturing systems scheduling. International Journal of Production Research, vol.35, n°6, pp. 1575-1591, 1997.

[Por 96] M. C. Portmann, Genetic algorithms and scheduling: A state of the art and some propositions. Workshop on Production and Planning Control Proceedings, Mons, Belgique, 1996.

[Rib 94] C.C. Ribeiro and N. Maculan (Eds), Applications of combinatorial optimization. Annals of Operations Research 50, 1994.

[Rio 01] G. Riotteau, O. Scaloni and E. Neron, Ordonnancement sur des machines identiques avec splitting et temps de préparation dépendant de la séquence. 3^{éme} Conférence Francophone de MOdélisation et SIMulation "Conception, Analyse et Gestion des Systèmes Industriels" MOSIM'01, Troyes, France, 2001.

[Ros 95] J. Rost, F.J. Markus and L. Yan-Hua, Agency Scheduling, A Model for Dynamic Task Scheduling. International Conference on Parallel Processing, Euro-Par 95, Sweden, pp. 635-646, 1995.

[Rot 66] M. Rothkopf, Scheduling independent tasks on parallel processors. Management Science 12, pp.347-447, 1966.

[Sak 95] M. Saksena, R. Gerber and W. Pugh, Parametric Dispatching of Hard Real-time Tasks. IEEE Trans. on Computers, 44(3):471-479, 1995.

[Sar 93] V. Sarkar and B. Simons, Parallel Program Graphs and their Classification. Sixth Workshop on Languages on Compilers for Parallel Computing, Portland, Oregon, Springer-Verlag Lecture Notes in Computer Science, 768, pp. 633-655, 1993.

[Ser 96] P. Seraphini, Scheduling jobs on several machines with the job splitting property. Operations Research (44), p. 617-628, 1996.

[Sha 90] L. Sha, R. Rajkumar and J.P. Lehoczky, Priority Inheritance Protocols: An Approach to real-time Synchronization. IEEE Transactions on computers, 39(9):1175-1185, 1990. Real Time Systems, The International Journal of Time-Critical Computing Systems, 2(3):181-194, 1990.

[Shi 90] B. Shirazi, M. Wang and G. Pathak, Analysis and Evaluation of Heuristic Methods for Static Task Scheduling. Journal of Parallel and Distributed Computing, Vol 10, pp. 222-232, 1990.

[Shu 96] W. Shu and M.Y. Wu, Runtime Incremental Parallel Scheduling (RIPS) on Distributed Memory Computers. IEEE Transactions on Parallel and Distributed Systems, Vol. 7, No. 6, pp 637-649, 1996.

[Smi 56] W.E. Smith, Various optimizers for single-stage production. Naval Research Logistics Quarterly, 3: pp. 59-66, 1956.

[Sta 95] J. Stankovic, M. Spuri, M. Di Natale and G. Buttazzo, Implications of Classical Scheduling Results for Real-Time Systems. IEEE Computer, 28(6):16-25, 1995.

[Ull 76] J. Ullman, Scheduling in Computer and JobShop Systems. E.G. Coffman Jr (eds.), J. Wiley, New-York, ch. Complexity of Sequencing problems,1976.

[Vig 95] A. Vignier, J.C. Billaut et C. Proust, Les problèmes d'ordonnancement de type Flow Shop Hybride : Etat de l'art. Actes des journées d'étude: Affectation et Ordonnancement. CNRS/ GdR Automatique / Pôle SED/GT3, Tours, pp. 7-47, 1995.

[Wil 93] M. Willebeek-LeMair and A.P. Reeves, Strategies for Dynamic Load Balancing on Highly Parallel Computers. IEEE Transactions on Parallel and Distributed Systems, vol. 4, N^{o} 9, pp. 979-993, 1993.

[Wu 97] M. Wu, W. Shu and J. Gu, Local Search for DAG Scheduling and Task Assignment. International Conference on Parallel Processing, pp.174-180, 1997.

[Yal 99] F. Yalaoui and C. Chu, Parallel machine scheduling with job splitting and setup times: a heuristic approach and performance analysis. Proceedings of the International Conference on Industrial Engineering and Production Management ,Glasgow, vol. 2, p. 297-305, 1999.

[Yan 94] T. Yang and A. Gerasoulis, DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors. IEEE Trans. on Parallel and Distributed System, Vol. 5, N^{o} 9, pp. 951-967, 1994.

[Yan 97] R. Yanpin, Z. Zuo and W. Qiufeng, An analysis of scheduling rules. IEEE International Conference on Intelligent Processing System, 1997.

[Yug 04] C. Yugma et L. Dupont, Sélection et ordonnancement de tâches en dynamique sur une machine. MOSIM '2004, Nantes, France, 2004.

ملخص

تعالج هذه الأطروحة مشكل جد و لة الأشغال في زمن واقعي على وسيلتين متماثلتين على التّوازي.
الغا ية هي تخفيض مدّة البرمجة الكليّة و زيا د ة عد د الأشغال الغير متوقعة المبرمجة ، مع مراعاة أزمنة الظّهو ر ، تخصيص الوسا ئل مع الأخذ بعين الاعتبار أزمنة النّهاية و مدّة المعالجة .
الطريقة المقترحة تميّز مرحلتين ، برمجة الأشغال المقرّرة و جد ولة ا الأشغال الغير متوقعة. هذه الأخيرة يتوجّب د مجها في جد ول الأشغال المبرمجة.
حالتين ، من بين العد يد عند جد ولة الأشغال الغير متوقعة ، تؤ خذ بعين الاعتبار، وجو د صفّ واحد يخصّ جميع الوسا ئل و تو فر كل وسيلة على صفّ يخصّها . في كلا الحالتين ، وضعت قوا عد للجد ولة.

المفا تيح: الجدولة ، زمن واقعي، وسائل متماثلة على التوازي ، أزمنة النهاية ، مدة المعالجة.

Résumé

Dans le cadre de cette étude nous traitons le problème d'ordonnancement temps-réel sur deux ressources en parallèle. Les ressources, dans ce cas, sont identiques.

L'objectif est de minimiser la durée totale d'ordonnancement, Makespan, ainsi que la maximisation du nombre de tâches aléatoires exécutées, sous contraintes de dates de disponibilité des tâches, contraintes disjonctives des ressources et les dates échues dont il est impératif de tenir compte ainsi que les durées opératoires.

L'approche proposée considère deux étapes, l'ordonnancement prévisionnel des tâches programmables et l'ordonnancement des tâches aléatoires. Lors de ce dernier, les tâches aléatoires sont insérées dans le programme d'exécution des tâches prévisionnelles.

Deux approches de résolution, parmi d'autres, sont à considérer en ordonnançant les tâches aléatoires : l'existence d'une file d'attente de ces tâches commune à toutes les ressources et le cas où chaque ressource possède sa propre file d'attente. Dans les deux approches, des algorithmes d'ordonnancement sont proposés et appliqués à des exemples.

Mots clés : Ordonnancement, Temps-Réel, Ressources Identiques en Parallèle, Dates de Disponibilité, Dates Echues, Durées Opératoires.

Abstract

This survey deals with the real-time two-identical parallel resources scheduling problem (on-line scheduling problem).

The objective is minimizing the Makespan and maximizing the executed random tasks number under release dates and disjunctive resource constraints with dues dates and processing times to consider.

The proposed approach considers two steps, the off-line scheduling of tasks available at the beginning and the on-line scheduling problem. In the real-time scheduling, random tasks are inserted in the first schedule.

To consider, in real-time scheduling, two cases: when one random tasks queue is common for all resources and each resource have its own queue of these tasks. For both of these cases, scheduling algorithms are proposed.

Keywords: Scheduling, Real-Time, Parallel Identical Resource, Release Time, Due Dates, Processing Time.