Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira de Béjaïa

Faculté des Sciences Exactes

Département Recherche Opérationnelle

Mémoire

En vue de l'obtention du diplôme de Master en Recherche Opérationnelle

Option : Modélisation Mathématique et Techniques de Décision

Thème

LES JEUX D'ORDONNANCEMENT

Soutenu devant le jury composé de :

Président	M^{elle}	BOUCHEBAH Kahina
Examinateur	M^{r}	KHIMOUM Nourdine
Examinateur	M^{r}	TOUATI Sofiane
Promotrice	M^{elle}	BOUCHAMA Kahina
Co-promoteur	M^{r}	RADJEF M.Said

Présenté par :

M^{elle} *Baouche* Salima

M^{elle} *Benanoune* Souad

Promotion 2013/2014

Dédicaces

Comme fruit de nos cinq ans d'étude

Je remercie Dieu pour le courage, la patience et la volonté

Je tiens en tous coeur pour dédier ce modeste travail à mes adorables

Parents qui ont toujours été à mes cotés et veiller sur moi, pour leurs conseils, bontés, amours, leurs assistances et leurs affections qui m'ont été d'un grand appui

A mes chères soeurs (Linda ,Kahina) pour leurs gentillesses, compréhension.

A ma belle soeur Samia que dieu te protège.

A mes chèrs frères(Nassim, Salim, Boubekeur, M.Cherfi, Yanis) qui ont étés d'un soutien moral,

A mes nièces poupées (Nawel, Malak, Nesrine, Imane).

A mes beaux frères (Farhat, A.Krim) qui m'ont encouragés tout au long de ce parcours.

A toutes personnes ayant participées et aider à franchir ce stade. Surtout Mes meilleurs(es) amis(es). Mes copines de chambre, Mes proches et mes collègues de promotion RO.

Et à tout ce qui me connaissent. A toutes ces personnes, sincèrement : Merci !.

Salima

Dédicaces

À la mémoire de mes défunts grands-parents paternels,

À ma Mami que j'aime beaucoup, quoique je puisse dire, je ne peux exprimer mes sentiment d'amour et de respect à votre égard. Que le tout puissant vous garde pour nous.

À mon grand-père qui m'a toujours soutenu et aidé.

À mes bijoux si précieux... Merci maman, Merci papa pour tout ce que vous avez fait pour moi.

À mon frère Yacine et mes soeurs : Silia et Djouja.

À mes deux oncles : Kamel et Nourdine je n'oublierai jamais ce que vous avez fait pour moi...Merci.

À mes tantes (Farida, Samia, Hassiba et Tassadit).

À mes amours (Rayan, Doudoum et Lina).

À tout mes cousins et cousines.

À tout mes amis (Djoudjou, Djallal, Massi, Idir, Mohand, Karima, Dalila, Djawida, Hakim, Ouali, Sabrina, Lala, Mess, Asma, Adel, Salah, Lyes, Tarik, Jouba et Jimou)

À loulou d'être tout simplement là...Merci.

Souad

Remerciements

Louange à Allah le miséricordieux de nous avoir donné le courage la force et la volonté pour la réalisation de ce mémoire.

Nos vifs remerciements sont adressés à Mlle K.BOUCHAMA pour sa disponibilité, sa gentillesse et sa patience. Qu'elle trouve ici l'expression de nos sincères remerciements.

Nous adressons également nos vifs remerciements à Mr M.RADJEF pour nous avoir proposé le thème, Merci Monsieur.

Nous tenons également à remercier les membres du jury qui ont aimablement acceptés de nous honorer en lisant ce mémoire pour évaluer et apprécier notre travail, nous espérons qu'ils en soient satisfaits.

Nous remercions les personnes qui nous ont orientés au cours de ce projet, ainsi que l'ensemble des enseignants qui nous ont formés au cours de notre cursus à l'Université de BEJAIA, sans oublier nos collègues et amis.

"Avoir une stratégie de travail, c'est effectuer la recherche
la plus logique, la plus exhaustive et la plus rapide"
Hêleine Prévoteaux Jeam Claude Ultard

Table des matières

Table des figures

Liste des tableaux

Résumé

Ce mémoire s'intéresse aux problèmes d'ordonnancement du point de vue théorie des jeux. Après avoir fait une synthèse des travaux ayant abordés cette problématique, nous nous somme intéressé au problème de job shop en le modélisant sous forme d'un jeu non coopératif avec contraintes (méta-jeux).

La résolution de ce jeu revient à retrouver son équilibre social. Ce dernier correspond à la meilleure solution pour le problème du job shop. Notons que dans la littérature, très peu de travaux se sont intéressés aux méta-jeux et nous n'avons pas pu trouver d'algorithme les résolvant. Par conséquent, nous avons proposé un algorithme fondé sur sa définition tout en respectant les contraintes de notre problème d'origine et nous l'avons testé sur une instance à 3 jobs et comparé les résultats obtenus à ceux fourni par la résolution en utilisant un algorithme génétique.

Mots clés : Ordonnancement, Équilibre de Nash, Job shop, Équilibre social, Théorie des jeux, Méta-jeux, Heuristiques, Makespan.

Abstract

This thesis focuses on scheduling problems in a point of view of game theory. After synthesis of works that discussed this problematic, we have been interested to job shop problem by modeling it in a form of a non cooperative game with constraints (meta-game).

The resolution of this game comes to find its social equilibrium. The latter correspond to the best solution for the job shop problem. In the literature, very little works have been interseted to meta game and we could not find an algorithm that resolve it. Consequently, we proposed an algorithm based on its definition by respecting constraints of our problem and we have tested it on three jobs and compared the obtained results to those provided by the resolution using a genetic algorithm.

Key words : Scheduling, nash equilibrium, Job shop, social equilibrium, Game theory , Meta-game, Heuristic, Makespan.

Liste des Acronymes

AG : Algorithme Génétique.

EN : Equilibre de Nash (Nash equilibrum).

FJSP :Le Problème de Job Shop Flexible.

FH : Flow Shop hybride.

HGA :Algorithme Génétique Hybride.

J : Job Shop.

RL :Rocherche Locale.

RT :Recherche Tabou.

O : Open Shop.

PERT : Technique D'évaluation et de Controle des programmes.

PNE :Equilibre de nash en stratégies pures.

Introduction générale

Actuellement, les organisations industrielles sont de plus en plus complexes, et la nécessité de produire à moindre coût, des produits de haute qualité, et d'avoir des systèmes de production plus flexibles sont les objectifs de tout gestionnaire. Pour faire face à son environnement concurrentiel, l'entreprise doit organiser au mieux sa production en mettant en œuvre un système de gestion informatisé pour chacune de ses fonctions, particulièrement pour la fonction "ordonnancement " qui n'est pas de moindre importance.

L'ordonnancement d'atelier de production est un problème dont la résolution a fait l'objet de plusieurs travaux dans la littérature. La NP-difficulté de ce problème fait que la plupart des méthodes utilisées pour sa résolution sont approximatives. Malheureusement, elles ne permettent pas de prendre en charge la résolution de certains problèmes imposant la prise en comptes de critères hétérogènes sur les jobs à ordonnancer. Dans de telles situations, il serait plus approprié de faire appel aux outils de la théorie des jeux qui ont prouvé leur efficacité dans l'étude et l'analyse de situations d'interactions entre un ensemble d'agents, ayant chacun un but à atteindre. En effet, on recense ces dernières années un certain nombre de travaux qui abordent le problème d'ordonnancement sous l'angle de la théorie des jeux. L'existence de liens entre les concepts d'équilibre pour les modèles de jeux obtenus et la solution du problème d'ordonnancement, permet de trouver une solution exacte à ce problème tout en considérant les objectifs des jobs à titre individuel, ce qui est un avantage considérable comparé à la résolution par les méthodes approximatives.

Le problème d'ordonnancement de tâches est l'un des premiers problèmes qui ont été modélisé sous forme d'un jeu d'ordonnancement. Les principaux résultats démontrés pour cette classe ont été synthétisés par J. Cohen dans [15]. Le problème d'ordonnancement de la production à une seule machine a également été représenté sous forme d'un jeu non coopératif [10], où chaque joueur correspond à un job et devra choisir pour stratégie une durée d'attente minimale avant son accès à l'unique machine

sans la congestionner. A base de ce modèle, une étude des conditions d'existence de l'équilibre de Nash, des propriétés de l'espace des solutions réalisables et des performances des solutions obtenues ont été établies par CJ. Wang, YG. XI en 2004 [10]. Un jeu pour le problème à machines parallèles a également été proposé par E.T.O.Opiyo et al [14]. Un autre travail qui n'est pas de moindre importance et qui illustre l'approche théorie des jeux pour la représentation des problèmes d'ordonnancement est le modèle proposé en 2009, par G.Zhoue et al [21]. Dans ce papier, il était question de rechercher le meilleur ordonnancement réalisable pour le problème du jobshop flexible (PJSF) en passant par le calcul de l'équilibre de Nash du jeu associé. Ce jeu ne modélisant que la phase d'affectation du (PJSF), les contraintes de gamme opératoire et de séquencement sur les machines ne sont prises en compte qu'implicitement. Pour remédier à cela, nous avons considéré un cas particulier du (PJSF) où l'affectation des opérations aux machines est supposée bien défini, ç.à.d on considère un problème de jobshop. Ensuite, nous proposons de représenter ce problème par un méta-jeu, où l'on associe à chaque joueur i un job J_i. Une stratégie pour un joueur (job) serait alors de fixer une date de début pour chacune des opérations qui le constituent, tout en respectant les contraintes de gamme et de séquencement qui le lie au choix des autres joueurs. Le gain d'un joueur i est alors de minimiser la date de fin du job J_i. L'équilibre social correspond à une solution optimale pour ce problème de job shop. Malheureusement, il n'existe aucun algorithme dans la littérature permettant de retrouver cet équilibre. Par conséquent, notre second objectif est de proposer une procédure de calcul nous permettant de retrouver un équilibre social qui représentera une solution pour notre problème d'ordonnancement.

Pour présenter notre travail, nous l'avons réparti en quatre chapitres :

• Le premier chapitre est une introduction aux problèmes d'ordonnancement, il aborde les éléments fondamentaux de l'ordonnancement, ainsi que les différentes méthodes de résolution.

• Le deuxième chapitre donne un aperçu sur la théorie des jeux.

• Le troisième chapitre évoque quelques travaux dans la littérature faisant appel à l'approche théorie des jeux pour la résolution des problèmes d'ordonnancement.

• Le quatrième chapitre présente notre procédure de recherche de l'équilibre social, puis nous comparons nos résultats avec les résultats fourni par un algorithme génétique pour le même jeu de données.

Chapitre 1

Généralités sur les problèmes d'ordonnancement

1.1 Introduction

Un problème d'ordonnancement est composé, de façon générale, d'un ensemble de tâches soumises à certaines contraintes, et dont l'exécution nécessite des ressources. Résoudre un problème d'ordonnancement consiste à organiser ces tâches, c'est-à-dire à déterminer leurs dates de début et d'achèvement, et à leur attribuer des ressources, de telle sorte que les contraintes soient respectées, ces problèmes sont très variés.

Ce chapitre est dédié à la présentation des éléments essentiels autour des problèmes d'ordonnancement.

1.2 Éléments fondamentaux

1.2.1 L'ordonnancement

L'ordonnancement vise à améliorer les performances d'une entreprise tels que les coûts de production et les délais de livraison.

Ordonnancer un ensemble de tâches, c'est programmer leur exécution en leur allouant les ressources requises et en fixant leurs dates de début.

L'ordonnancement est lié à plusieurs secteurs de recherches et d'activités. En informatique, le choix des tâches à envoyer aux processus se modélise comme un problème d'ordonnancement. En production, l'ordonnancement consiste à déterminer les séquences d'opérations à réaliser sur les différentes machines de

l'atelier. En gestion de projet, ordonnancer, c'est déterminer les dates d'exécution des activités constituant le projet. Dans un problème d'ordonnancement interviennent deux notions fondamentales : les tâches et les ressources.

1.2.2 Les tâches

Une tâche est un travail mobilisant des ressources et réalisant un progrès significatif dans l'état d'avancement du projet compte tenu du niveau de détail retenu dans l'analyse du projet.

D'une façon plus schématique, une tâche qu'on note i est une entité élémentaire de travail localisée dans le temps par une date de début t_i et de fin c_i dont la réalisation nécessite une durée p_i telle que $p_i = c_i - t_i$ et qui utilise des ressources.

Généralement, une tâche est caractérisée par trois paramètres :

◦ La durée opératoire p_i (*processing time*) : c'est la durée d'exécution de la tâche.

◦ La date de disponibilité r_i (*release time*) : c'est la date de début au plus tôt.

◦ La date d'échéance d_i (due date) : c'est la date de fin au plus tard [27].

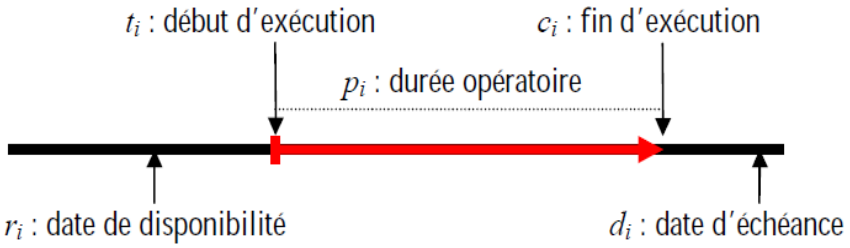


FIGURE 1.1 – Caractéristiques d'une tâche i.

On distingue deux types de tâches [24] :

◦ **les tâches morcelables** (préemptives) qui peuvent être exécutées en plusieurs fois, facilitant ainsi la résolution de certains problèmes.

◦ **les tâches non morcelables** (indivisibles) qui doivent être exécutées en une seule fois et ne peuvent être interrompues.

1.2.3 Le job

Un job est un ensemble de tâches séquencées suivant un certain ordre formant la gamme opératoire [9].

1.2.4 Les ressources

Une ressource est un moyen technique ou humain utilisé pour réaliser une tâche. On trouve plusieurs types de ressources :

◦ **Les ressources renouvelables** : c'est des ressources, après avoir été allouées à une tâche, redeviennent disponibles en même quantité (machines, personnel, etc).

◦ **Les ressources consommables** : ce sont des ressources qui ne sont pas disponible en même quantité (argent, matières premières, etc) [24].

Dans le cas des ressources renouvelables, on distingue principalement, les ressources disjonctives qui ne peuvent exécuter qu'une tâche à la fois et les ressources cumulatives qui peuvent être utilisées par plusieurs tâches simultanément mais en nombre limité [33].

◦ **Les ressources doublement contraintes ou limitées** : qui combinent les contraintes liées aux deux catégories précédentes [33].

1.2.5 Les contraintes

Une contrainte exprime des restrictions sur les valeurs que peuvent prendre simultanément les variables représentant les relations reliant les tâches et les ressources. On distingue deux types de contraintes : les contraintes temporelles et les contraintes de ressources [33].

• **Les contraintes temporelles**

Les contraintes temporelles concernent les délais de fabrication imposés. On cite :

◦ **Les contraintes de dates butoirs** : certaines tâches doivent être achevées avant une date préalablement fixée [24].

◦ **Les contraintes de précédence** : une tâche i doit précéder une tâche j.

◦ **Les contraintes de dates au plus tôt** : liées à l'indisponibilité de certains facteurs nécessaires pour commencer l'exécution des tâches [33].

• **Les contraintes de ressources**

Ces contraintes concernent la limitation de la quantité de ressources de chaque type. Dans ce cadre, deux types de contraintes de ressources sont distinguées :

◦ **Les contraintes disjonctives** : elles induisent une contrainte de réalisation des tâches sur des intervalles temporels disjoints pour une même ressource.
◦ **Les contraintes cumulatives** : elles impliquent la limitation du nombre de tâches à réaliser en parallèle [33].

1.2.6 Les objectifs d'ordonnancement

En résolvant un problème d'ordonnancement, nous pouvons avoir différents objectifs, à savoir :
◦ **Les objectifs liés au temps** : On trouve par exemple la minimisation du temps total d'exécution, du temps moyen d'achèvement, des durées totales de réglage ou des retards par rapport aux dates de livraison.
◦ **Les objectifs liés aux ressources** : telle que la maximisation de la charge d'une ressource ou minimiser le nombre de ressources nécessaires pour réaliser un ensemble de tâches sont des objectifs de ce type.
◦ **Les objectifs liés au coût** : il s'agit généralement de minimiser les coûts de lancement, de production, de stockage, de transport, etc [27].

1.3 Classification des problèmes d'ordonnancement

Les problèmes d'ordonnancement peuvent être classifiés selon le nombre de machines et leur ordre d'utilisation pour fabriquer un produit qui dépend de la nature de l'atelier. Un atelier se définit par le nombre de machines qu'il contient et par son type. En ordonnancement d'atelier, les tâches représentent les opérations élémentaires de fabrication, les ressources correspondent aux machines utilisées et un travail correspond à un ensemble d'opérations de fabrication nécessaires à sa réalisation [33].
Selon la structure de l'atelier de production, nous pouvons distinguer différentes classes de problèmes d'ordonnancement :

1.3.1 Les problèmes à une seule machine

Pour ces problèmes, chaque job est constitué d'une seule opération.

1.3.2 Les problèmes à machines parallèles

Il s'agit d'une généralisation des problèmes à machine unique. Dans ce cas, l'atelier dispose de machines en plusieurs exemplaires pour le traitement d'un ensemble de travaux. Ainsi, le choix de la machine est à déterminer. Ces dernières peuvent être de différentes natures :

- Les machines parallèles identiques.
- Les machines parallèles uniformes.
- Les machines parallèles indépendantes[33].

1.3.3 Les problèmes à machines multiples

Trois ateliers spécialisés sont différenciés, selon que la gamme de fabrication (l'ordre de fabrication) est commune à tous les travaux, on parle alors d'atelier à cheminement unique ou flow shop. Dans le cas où la gamme opératoire est spécifique à chaque job, on parle d'atelier à cheminement multiples ou job shop, quand cette gamme n'est pas définie alors l'atelier est à cheminement libre où open shop [33].

1.3.3.1 Le flow shop

C'est un cas particulier du problème d'ordonnancement d'atelier pour lequel le cheminement des travaux est unique : les n travaux utilisent les m machines dans un même ordre [33].

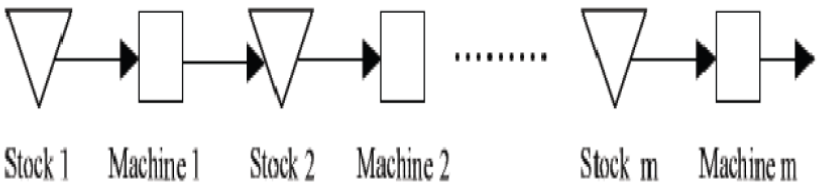


FIGURE 1.2 – Représentation d'un atelier flow shop [24].

1.3.3.2 Le flow shop hybride

Il s'agit d'ordonnancer un ensemble de n jobs qui sont représentatifs des scénarios de production possibles $J_1, J_2, ..., J_n$ dans un atelier de production composé de plusieurs étages [37]. Chaque étage est composé d'un certain nombre de machines parallèles. Les produits doivent être traités sur une seule machine de chaque étage et l'ordre des étages est le même pour tous les produits [20].

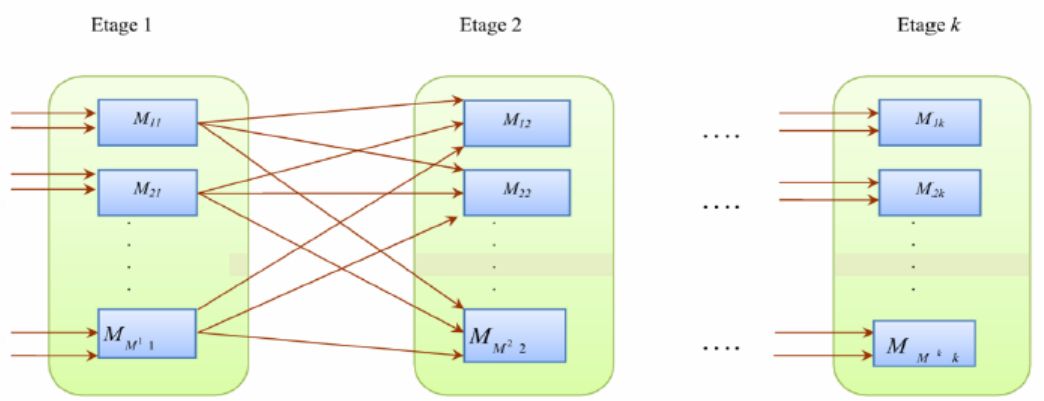


FIGURE 1.3 – Représentation d'un atelier flow shop hybride [37].

1.3.3.3 Le job shop

Les problèmes d'ordonnancement de type job shop font partie des problèmes les plus étudiés dans la littérature. Cela s'explique par deux raisons : l'importance théorique du modèle, et de nombreuses applications pratiques qui peuvent être modéliser de cette façon.

les premiers résultats théoriques concernant le job shop datent des années cinquante avec la résolution du problème à 2 machines et d'un cas particulier du problème à 3 machines. On trouve sa définition formelle dans l'ouvrage "Industrial Scheduling" de Muth et Thompson [16], paru en 1963, où trois exemples de job shop, considérés depuis comme des données tests sont proposés. Les deux premiers exemples ont été résolus dans les années soixante-dix . En revanche, il a fallu attendre les années quatre-vingt pour la résolution du troisième.

Le cas du job shop se rencontre dès qu'un atelier fabrique plusieurs produits présentant des caractéristiques différentes sur un nombre limité de ressources. En effet, l'ordonnancement de type job shop consiste à réaliser un ensemble de n jobs $J = \{J_1, J_2, \ldots, J_n\}$ sur un ensemble de m ressources

$M = \{M_1, M_2 \dots M_m\}$. Chaque job $J_i \in J$ est composé d'une série de n_i opérations élémentaires, devant être exécutées sur les différentes ressources selon un ordre préalablement défini. Cet ordre est appelé gamme opératoire [3]. Il traduit les contraintes de précédence entre les différentes opérations du job. Par ailleurs, chaque opération possède une durée de traitement connue à l'avance et ne peut être exécutée que par une et une seule ressource. De plus, chaque job doit être réalisé dans un intervalle de temps déterminé par la date de début au plus tôt du job avant laquelle le traitement ne peut pas commencer, et par sa date de fin au plus tard déterminant une limite supérieure pour la fin de réalisation.

La théorie de complexité a permis de classifier le problème job shop en un problème NP-difficile [33]. Par conséquent, il n'existe pas de méthodes universelles permettant de le résoudre efficacement sans que le temps de résolution n'augmente de façon exponentielle avec la taille de celui-ci.

L'espace des solutions pour une instance du problème a une grande importance en effet il existe $(n!)^m$ différentes solutions pour un problème de n tâches à réaliser sur m machines, énumérer toutes ces possibilités pour arriver à la solution optimale n'est pas une chose réaliste [27].

Il est à noter que ce nombre évolue plus vite en fonction du nombre des tâches qu'en fonction du nombre des machines.

Autrement dit l'ajout d'une tâche a beaucoup plus d'impact que l'ajout d'une machine.

La difficulté du job shop est fonction du nombre de tâches, du nombre de machines, du nombre d'opérations par tâche, et de la durée des opérations.

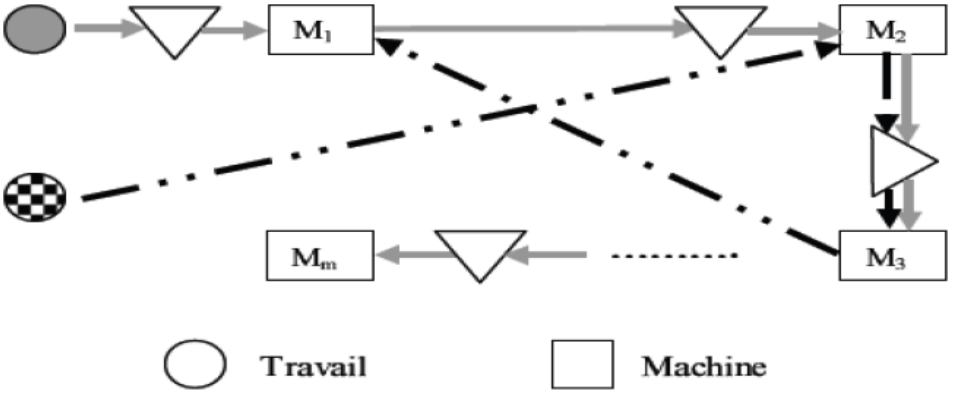


FIGURE 1.4 – Représentation d'un atelier job shop [24].

1.3.3.4 Le job shop flexible

Le problème d'ordonnancement de type job shop flexible, est une généralisation du problème de job shop où une opération donnée peut être réalisée par une ou plusieurs ressources, et y possède une durée de traitement dépendant de la ressource utilisée. Les contraintes relatives à ce type de problème sont de type précédence, temporel ou disjonctif [23].

1.3.3.5 L'open shop

C'est un modèle d'atelier moins contraint que le job shop et le flow shop, car l'ordre d'exécution des tâches d'un travail y est libre.

L'open-shop modélise des problèmes courants, comme l'organisation d'examens médicaux dans un hôpital ou la planification d'un atelier de réparation automobile. Dans ces deux exemples, les travaux sont respectivement les patients et les voitures, les tâches sont les examens médicaux et les réparations à effectuer, les machines étant les salles ou appareils d'examen (radio, endoscopies...) et les postes de travail (graissage, lavage, etc...).
Un problème de type open shop est un job shop dans lequel les contraintes de précédence sont relâchées. C'est-à dire, les opérations peuvent être effectuées dans n'importe quel ordre [20].

1.3.3.6 Mixed Shop

Quelques tâches seront affectées d'autres non [11].

1.3.4 Notations

Dans la littérature, un formalisme de notation a été proposé pour représenter un problème d'ordonnancement d'une manière simple. Il consiste à représenter le problème par trois champs $\alpha,\ \beta\ et\ \gamma(\alpha/\ \beta/\ \gamma)$
- le champ α permet de décrire le type d'atelier et le nombre de machines disponibles [2].
- le champ β caractérise les conditions d'exécution des jobs ainsi que les états des ressources présentes dans l'atelier. Il indique en particulier la présence ou non des contraintes de précédence entre les tâches et la possibilité de tolérer la préemption.
- le champ γ, dédié au(x) critère(s) à optimiser.

Le paramètre $\alpha_1 \in \{1\ ou\ \varnothing, P, Q, R, O, F, J, FH\}$ caractérise le type de machines utilisées :
$\alpha_1 = 1$ auquel cas nous avons un problème à une seule machine.

$\alpha_1 = P$ machines identiques parallèles.

$\alpha_1 = Q$ machines parallèles uniformes.

$\alpha_1 = R$ machines parallèles quelconques (unrelated).

$\alpha_1 = O$ il s'agit d'un open shop.

$\alpha_1 = F$ système Flow shop.

$\alpha_1 = J$ système Job shop.

$\alpha_1 = FH$ flow shop hybride.

$\alpha_2 \in \{\varnothing, k\}$ est un entier qui représente le nombre de machines dans le problème.

$\alpha_2 = \varnothing$ le nombre de machines est supposé être variable.

$\alpha_2 = k$ le nombre de machines est égal à k ($k \neq 0$).

Le second champ$\beta = \beta_1\beta_2\beta_3\beta_4\beta_5\beta_6\beta_7\beta_8$ décrit la tâche et les caractéristiques de la ressource.

Le paramètre $\beta_1 \in \{\varnothing, pmtn\}$ indique la possibilité de la préemption de la tâche.

$\beta_1 = \varnothing$ la préemption n'est pas autorisée.

$\beta_1 = pmtn$ la préemption est autorisée.

Le paramètre $\beta_2 \in \{\varnothing, res\}$ caractérise les ressources additionnelles.

$\beta_2 = \varnothing$ aucune ressource additionnelle n'existe.

$\beta_2 = res$ il y a des contraintes de ressources spécifiées.

Le paramètre $\beta_3 \in \{\varnothing, prec, uan, tree, chains\}$ reflète les contraintes de précédence.

$\beta_3 = \varnothing, prec, uan, tree, chains$ représentent respectivement tâches indépendantes, contraintes de précédence générales, réseau d'activité uniconnexe, contraintes de précédence formant un arbre ou une chaîne.

$\beta_4 \in \{\varnothing, r_j\}$ les temps au plus tôt sont zéro, soit les temps au plus tôt différent par tâche.

$\beta_5 \in \{\varnothing, p_j = p, \overline{p} \leq p_i \leq \underline{p}\}$ les tâches ont des temps d'exécution arbitraires, le temps d'exécution dans l'intervalle défini.

$\beta_6 \in \{\varnothing, d_i\}$ décrit les deadlines (dates d'écheances) : aucune date d' écheance n'est supposée dans le système avec pénalité(dates au plus tard peuvent être définies si le critère utilisé pour évaluer l'ordonnancement est celui de date au plus tard), deadlines sont imposées sur la performance de l'ensemble des tâches.

$\beta_7 \in \{\varnothing, n_j \leq k\}$ décrit le nombre maximum de tâches constituant un job dans le cas du système job shop : le nombre est arbitraire ou le problème d'ordonnancement n'est pas du job shop ; le nombre de tâches pour chaque job n'est pas plus grand que k.

$\beta_8 \in \{\varnothing, no - wait\}$ décrit une propriété sans attente de l'ordonnancement sur des machines dédiées :

la région tampon est de capacité illimitée ; sans attente les capacités d'attente sont nulles. Une tâche qui termine son exécution sur une machine passe instantanément sur une autre machine [2].
Le troisième champs γ signifie le critère à utiliser par exemple :
$\gamma = C_{max}$ la durée totale (makespan).
$\gamma = L_{max} = max\{L_i = C_i - d_i\}$ le plus grand retard algébrique.
$\gamma = T_{max} = max\{0, C_i - d_i\}$ le plus grand retard vrai.
$\gamma = \sum U_i$ le nombre de travaux en retard.
$\gamma = \sum[w_i C_i]$ la durée moyenne ou pondérée des travaux.

1.4 Méthode de résolution des problèmes d'ordonnancement

Le problème d'ordonnancement étant combinatoire, il est improbable de trouver un algorithme polynomial pour le résoudre dans le cas général. Pour des problèmes de grandes tailles, on ne cherche que des solutions approchées. Les algorithmes qui fournissent des solutions optimales ne s'appliquent qu'aux problèmes de petite taille.

1.4.1 Les méthodes de résolution exactes

Les méthodes de résolution exactes permettent d'obtenir une solution dont l'optimalité est garantie, dans certaines méthodes.
parmi elles on cite :
-La programmation linéaire.
-La programmation dynamique.
-Les procédures par séparation et évaluation (branch and bound, ...) [33].

1.4.2 Les méthodes de résolution approchées

Les méthodes approchées, dites aussi d'approximation, constituent une alternative intéressante pour traiter les problèmes d'ordonnancement de grande taille [33]. Elles sont définies comme étant des procédures exploitant au mieux la structure du problème considéré afin de trouver une solution de qualité raisonnable en un temps de calcul aussi faible que possible.
Lorsque ces méthodes sont conçues de manière simple, rapide et ciblée sur un problème particulier, on

les appelle heuristique. Une heuristique est un algorithme approché qui permet d'identifier en temps polynomial au moins une solution réalisable rapide, pas obligatoirement optimale. L'usage d'une heuristique est efficace pour calculer une solution approchée d'un problème.
Lorsque celles-ci sont générales, adaptables et applicables à plusieurs catégories de problèmes d'optimisation combinatoire, elles portent le nom de méta-heuristique, Une méta-heuristique est un algorithme qui est plus complet et complexe qu'une simple heuristique, et permet généralement d'obtenir une solution de très bonne qualité pour des problèmes issus des domaines de la recherche opérationnelle ou de l'ingénierie dont on ne connaît pas de méthodes efficaces pour les traiter ou bien quand la résolution du problème nécessite un temps élevé ou une grande mémoire de stockage.

1.4.2.1 La recherche locale (RL)

La recherche locale (RL) est un algorithme fort simple qui fouille l'espace de solutions d'un problème en visitant pas à pas chaque solution. Une opération élémentaire est effectuée sur la solution s pour donner une solution voisine s' [24].

1.4.2.2 Le recuit simulé

Le recuit simulé est un algorithme stochastique itératif qui progresse vers l'optimum par échantillonnage d'une fonction objectif, cet algorithme est inspiré d'un processus utilisé en métallurgie. Après avoir fait subir des déformations au métal, on réchauffe celui-ci à une température, de manière à faire disparaître les tensions internes causées par la déformation, puis on laisse refroidir lentement. L'énergie fournie par le réchauffement lent fige peu à peu le système dans une structure d'énergie minimale [4].
L'idée principale est alors d'accepter des déplacements dans le voisinage d'une solution en dépit du fait qu'ils dégradent le coût de la solution et cela suivant une probabilité calculée à partir d'une température T du système à un moment donné [24].

1.4.2.3 La recherche tabou

La recherche tabou se comporte comme toute méthode de voisinage et améliore à chaque étape la valeur objectif. Lorsque l'on atteint par contre un minimum local, on se déplace vers le voisin le moins mauvais.
L'inconvénient de cette dernière démarche est que parfois, une itération ne suffit pas pour s'en sortir d'un minimum local et un déplacement peut provoquer un déplacement inverse à une étape ultérieure, ce qui provoquera un cycle autour du minimum local [27]. C'est pour cette raison que la recherche tabou

garde en mémoire les dernières solutions visitées pour éviter d'y revenir, c'est ce qu'on appelle la liste tabou.

1.4.2.4 L'optimisation par colonie de fourmis

Cette méta-heuristique été introduite en 1991 par Colorni, Dorigo et Maniezzo pour résoudre le problème du Voyageur de commerce. Elle s'est popularisée, puis a été l'objet d'améliorations dès 1995 et a été appliquée avec succès à d'autres problèmes d'optimisation combinatoire dès 1994 [7].

Comment les fourmis virtuelles peuvent être exploitées pour résoudre un problème d'optimisation combinatoire ?

Les fourmis sont capables de résoudre collectivement des problèmes complexes, comme trouver le plus court chemin entre deux points dans un environnement accidenté. Pour cela, elles communiquent entre elles de façon locale et indirecte, grâce à une hormone volatile, appelée phéromone : au cours de leur progression, les fourmis déposent une trace de phéromone ; elles choisissent ensuite leur chemin de façon aléatoire, selon une probabilité dépendant de la quantité de phéromone précédemment déposée.

Ce mécanisme, qui permet aux fourmis de résoudre collectivement des problèmes complexes, est à l'origine des algorithmes à base de fourmis artificielles, comme une approche multi-agents pour résoudre des problèmes d'optimisation combinatoire. L'idée est de représenter le problème à résoudre sous la forme de la recherche d'un meilleur chemin dans un graphe, puis d'utiliser des fourmis artificielles pour rechercher de bons chemins dans ce graphe. Le comportement des fourmis artificielles est inspiré des fourmis réelles : elles déposent des traces de phéromone sur les composants du graphe et elles choisissent leurs chemins relativement aux traces de phéromone précédemment déposées ; ces traces sont évaporées au cours du temps. Intuitivement, cette communication indirecte fournit une information sur la qualité des chemins empruntés afin d'attirer les fourmis, dans les itérations futures, vers les zones correspondantes de l'espace de recherche.

Ces caractéristiques du comportement des fourmis artificielles définissent la 'méta heuristique' d'optimisation par une colonie de fourmis qui a permis de résoudre différents problèmes d'optimisation combinatoire, comme le problème du voyageur de commerce, le problème d'affectation quadratique, ou le problème de routage de véhicules [18].

1.4.2.5 Les réseaux de neurones

Ils ont été appliqués à la résolution approchée des problèmes d'optimisation en général et d'ordonnancement en particulier. Leur avantage est la possibilité d'utiliser des circuits analogiques dont l'évolution

converge vers des états stables correspondant à une solution approchée du problème d'ordonnancement. Leur inconvénient est le blocage à la rencontre d'un minimum local [24].

1.4.2.6 Les algorithmes génétiques

Les Algorithmes Génétiques (A.G.) sont des algorithmes itératifs dont le but est d'optimiser une fonction prédéfinie, appelée fitness. Pour réaliser cet objectif, l'algorithme travaille sur un ensemble de points, appelés population d'individus. Chaque individu ou chromosome représente une solution possible du problème donné. Il est constitué d'éléments, appelés gènes, dont les valeurs sont appelées allèles. L'utilisation d'un algorithme génétique nécessite la définition, au préalable, d'un espace de recherche dont les éléments de base sont les chromosomes et d'une fonction définie sur cet espace dite fonction fitness.

◦ Organigramme d'un algorithme génétique

Les AG se caractérisent par le fait que :
- Ils utilisent un codage des paramètres, et non les paramètres eux-mêmes.
- Ils travaillent sur une population de points, au lieu d'un point unique.
- Ils n'utilisent que les valeurs de la fonction étudiée ; pas sa dérivée ou une connaissance auxiliaire.
- Ils utilisent des règles de transition probabilistes et non déterministes[23].

◦ **Les étapes de l'algorithme génétique** :

◦ **Codage**

Le premier pas dans l'implantation des algorithmes génétiques est de créer une population d'individus initiaux. Chaque individu de la population est codé par un chromosome. Une population est donc un ensemble de chromosomes [37].

Généralement, un chromosome est une chaîne ou un ensemble de chaînes, souvent de bit ou d'entiers. Chaque chromosome code un point de l'espace de recherche. L'efficacité de l'algorithme génétique va donc dépendre du choix du codage d'un chromosome.

◦ **Evaluation** : fitness

Les performances de chaque individu sont évaluées en utilisant une fonction d'évaluation. Cette fonction permet d'évaluer la capacité d'un individu à survivre en lui affectant un poids appelé fitness. Cet fitness est calculée afin que les plus forts soient retenus dans la phase de sélection, puis modifiés dans la phase

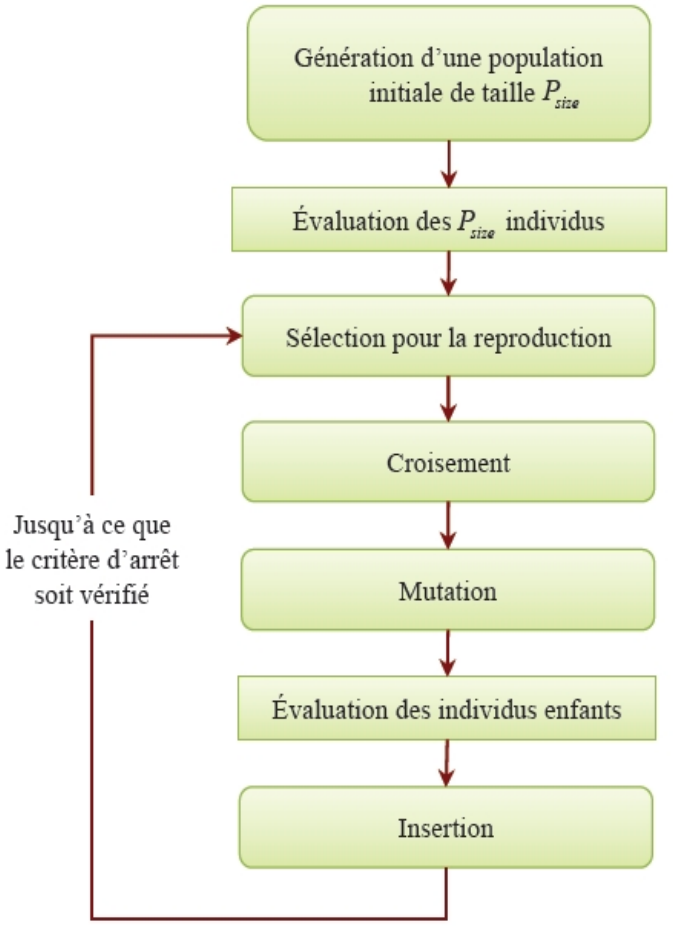


FIGURE 1.5 – Organigramme d'un algorithme génétique [37]

de croisement et mutation [37].

- **La sélection**

La phase de sélection consiste à choisir parmi les N individus de la population courante les plus forts à partir desquels la génération suivante sera créée. Soit pour un ordonnancement i réalisable, on calcul la valeur de la fonction objectif « Makespan, par exemple ». On associe à chaque solution une fonction d'évaluation pour calculer sa force d'adaptation[26].

- **Croisement**

Le croisement a pour but d'enrichir et de diversifier la population en manipulant la structure des chromosomes [23].

-L'opérateur de croisement en un point proposé consiste à choisir deux parents et un point de croisement. Le parent 1 (respectivement le parent 2) reçoit les chromosomes du parent 2 (respectivement du parent 1) qui suivent le point de croisement, comme le montre la figure suivante.

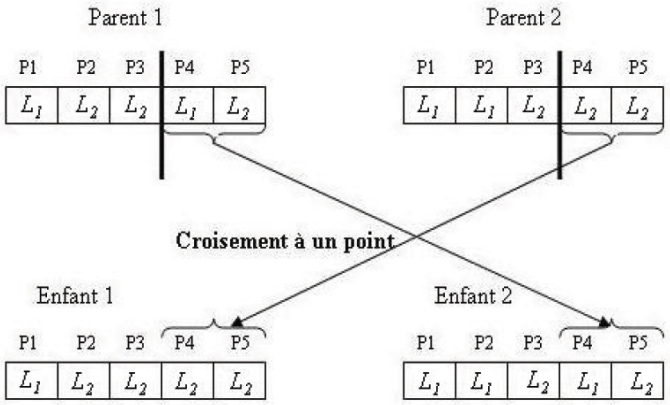


FIGURE 1.6 – Exemple d'un croisement à un point

◦ **Mutation**

Une mutation est une perturbation introduite sur la composante de l'individu afin de garantir la diversité et élargir le champ d'exploration, elle consiste à remplacer, de façon aléatoire, un gène au sein d'un chromosome par un autre, c'est donc une modification légère dans le chromosome muté. La mutation a pour rôle d'éviter la convergence trop rapide de l'algorithme vers un minimum local. La figure suivante montre un exemple de mutation .

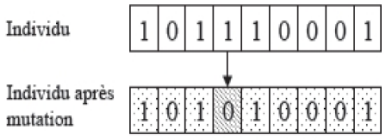


FIGURE 1.7 – Exemple d'une mutation [24]

◦ **Phase de remplacement**

Cette étape constitue la population de la génération suivante à partir des parents et des enfants de la génération courante. Une fraction de la population est remplacée par sa descendance à chaque

génération. L'écart entre les générations indique la proportionnel de parents remplacés par des enfants.

◦ **Critère d'arrêt**

Le critère d'arrêt peut être de diverse nature comme par exemple un certain temps de calcul à ne pas dépasser ou un certain nombre d'itérations ou un certain taux d'adaptation de la population à atteindre...[5].

1.5 Représentation des problèmes d'ordonnancement

Il existe trois sortes de représentations possibles d'un problème d'ordonnancement : le diagramme de Gantt, le graphe Potentiel-Tâches et la méthode PERT.

1.5.1 Représentation à l'aide d'un graphe potentiels-tâches

Cet outil graphique a été développé grâce à la théorie des réseaux de Pétri qui ont surtout servi à modéliser les systèmes dynamiques à événements discrets. Dans ce genre de modélisation, les tâches sont représentées par des noeuds et les contraintes par des arcs [23]. Ainsi, les arcs peuvent être de deux types :

- Les arcs conjonctifs illustrant les contraintes de précédence et indiquant les durées des tâches, - Les arcs disjonctifs indiquant les contraintes de ressources .

1.5.2 Représentation à l'aide d'un diagramme de Gantt

Le diagramme de Gantt constitue un formalisme graphique qui a été mis au point par Henry Gantt en 1910. Il s'agit d'un outil couramment utilisé pour représenter la solution d'un problème d'ordonnancement. Il permet de visualiser l'utilisation des machines, l'avancement de l'exécution des opérations sur celles-ci ainsi que les dates de début et de fin de chaque opération [33].

1.5.3 Représentation à l'aide de la méthode PERT

Le Program Evaluation and Review Technic (Technique d'Evaluation et de Contrôle des Programmes) est une méthode mise au point en 1958 par Willard Frazard. Elle permit à l'US NAVY de gagner 2 ans sur la fabrication des fusées Polaris (projet établi initialement sur 7 ans).

il a pour rôle de :

◦ Présenter d'une façon visuelle, l'enchaînement logique des tâches en vue de :

- Faciliter la coordination et le contrôle.

- Améliorer les prévisions de durée et de coût.

◦ Connaître le chemin critique, c'est-à-dire le chemin le plus long entre la première et la dernière étape, et par conséquent :

- La durée totale du projet.

- Les tâches pour lesquelles tout retard entraîne l'allongement du projet.

1.6 Conclusion

Dans ce chapitre nous avons rappelé les principales définitions et notations utilisées dans la résolution des problèmes d'ordonnancement en précisant les différents éléments qui les déterminent ainsi que leur classification.

De même, ce chapitre nous a permis de passer en revue un ensemble de méthodes existantes pour résoudre les différents problèmes d'ordonnancement. Elles sont divisées en deux grandes classes, selon qu'elles les résolvent de manière exacte ou approchée. La première classe requière des ressources informatiques importantes dès que la taille du problème augmente. La deuxième permet d'offrir le plus souvent un bon compromis entre la qualité des solutions et le temps de calcul.

Chapitre 2

Théories des jeux

2.1 Introduction

La théorie des jeux est une théorie mathématiques qui a pour objet d'établir et d'étudier les interactions stratégiques entre plusieurs agents rationnels. Les mots important dans cette définition sont :

• Interaction : il y a plusieurs agents (appelés aussi joueurs, "decision makers", etc...), et ils interagissent : le contentement (appelé aussi paiement, gain, utilité, bien-être) de chacun ne dépend pas que de lui même, mais aussi du choix de ses adversaires.

• Stratégique : les joueurs ont le choix entre plusieurs options.

• Rationnel : un joueur ne joue pas n'importe comment, il cherche à optimiser son paiement. Un des buts de la théorie des jeux est d'abord de créer des modèles mathématiques de base . Ces modèles synthétisent tous les éléments essentiels pour décrire l'interaction, puis introduisent des concepts de solution pour décrire les issues possibles d'un jeu, ils interprètent les résultat pour mieux comprendre les phénomènes étudiés [19].

L'objectif de ce chapitre est de donner les éléments de base de la théorie des jeux, en commençant par définir les jeux et leurs caractéristiques ainsi que leurs concepts de solutions.

Le but de ce chapitre est de donner un aperçu sur la théorie des jeux en dernier nous allons donner la définition d'un méta-jeu qui sera utilisée dans le dernier chapitre.

2.2 Définitions

Définition 2.1. *Un jeu est une situation dans laquelle les joueurs sont conduits à faire des choix stratégiques parmi un certain nombre d'actions possibles,*

vérifiant les règles du jeu [30].

Le résultat de ces choix constitue une issue du jeu à laquelle est associé un gain pour chacun des participants.

Définir un jeu revient à donner l'ensemble des joueurs, l'ensemble de choix possibles (ensemble des stratégies) pour chaque participant et la fonction qui calcule leurs gains (fonction utilité) dans toutes les éventualités possibles [29].

• **Les Joueurs :** Un joueur peut être une personnes, une société,...etc. Si dans un jeu, il y'a N joueurs $N \geq 2$ qui participant au jeu, on notera par $I = \{1, 2, ..., N\}$ l'ensemble des joueurs [30].

• **Les Stratégies :**

Définition 2.2. *Une stratégie pure du joueur i est une action où un plan d'actions choisit avec "certitude" par ce joueur afin de faire face à toute les situations possibles du jeu.*
On note par X_i, l'ensemble des stratégies pures du joueur i et par x_i une stratégie pure de ce joueur [29].

Définition 2.3. *Une stratégie mixte est définie comme une distribution de probabilité sur l'ensemble des stratégies pures [29].*
Les stratégies mixtes du joueur i seront alors notées par $\alpha_i \backslash \alpha_i \in \triangle_{|i|}$ où

$$\triangle_i = \{\alpha_i = (\alpha_{i1}, .., \alpha_{i|X_i|}) \in \mathbb{R}^{|X_i|}, \ \ \alpha_i j \geq 0, \ ;\forall i = \overline{1, |X_i|}, \ \ \sum_{j=0}^{|X_i|} \alpha_{ij} = 1\} \tag{2.1}$$

où α_{ij} est la probabilité que le joueur i joue sa stratégie pure $x_j \in X_i$.

• **La fonction d'utilité (gain) :** L'utilité est une mesure individuelle du contentement de l'agent. Utiliser une fonction d'utilité sert à définir les préférences des joueurs [6, 30]. pour tout $i \in I = \{1, ..., N\}$, on associe une fonction U_i définie sur l'ensemble $X = \prod_{j=1}^{N} X_j$ des issues possibles du jeu :

$$\begin{gathered} U_i : X = \prod_{j=1}^{N} X_j \longrightarrow \mathbb{R} \\ x \in X \longrightarrow U_i(x) \end{gathered}$$

Où : $x = (x_1, x_2, ..x_N)$ une issue du jeu [30].

2.3 Les classes de jeux

Nous allons présenter ici les différentes classes de jeux.

2.3.1 Les jeux statiques et dynamiques

La première distinction que nous allons faire est celle entre un jeu statique et dynamique.

Définition 2.4. *[6, 25] Un jeu statique est un ensemble de règles qui encadre le comportement des joueurs et qui détermine leurs gains selon les actions entreprises. Formellement, un jeu J est constitué d'un ensemble de joueurs $I = \{1, ..., N\}$, l'ensemble des profils de stratégies ou issues du jeu $X = X_1 \times \ldots \times X_n$ où X_i représente l'ensemble des choix possibles (stratégies) du joueur i. Ce qui suit que l'ensemble des X_i est défini. On définit également une fonction d'utilité ou de gains de chaque joueur en fonction de l'issue du jeu :*
$u_i : S \longrightarrow \mathbb{R}$. Le joueur i préfère strictement l'issue x à l'issue $\acute{x}$ si $u_j(\acute{x}) > u_j(x)$ si $u_j(x) = u_j(\acute{x})$ alors i est indifférent entre ces deux issues.
Un profil de stratégies, appelé aussi issue du jeu, est une combinaison de stratégies individuelles $x = (x_1, \ldots, x_N) \in X_1 \times \ldots \times X_N$.

Définition 2.5. *Un jeu dynamique est un jeu qui se déroule en plusieurs étapes. On se place ici dans le cadre des jeux dynamiques en information complète, c'est-à-dire que l'on admet que toutes les actions passées sont observables et connues de tous les joueurs [6].*

Dans ce cadre, en intervenant à des étapes antérieures du jeu, certains joueurs ont le pouvoir d'affecter directement les gains d'autres joueurs de manière irréversible.
Ces actions ne sont pas connues et chacun des joueurs intervenant à cette étape se comporte un peu comme dans un jeu statique, à la différence que dans ce cas, l'histoire du jeu influence le choix de chacun.

2.3.2 Les jeux coopératifs et non coopératifs

Un jeu coopératif (appelé aussi jeu coalitionnel) est un jeu dans lequel les joueurs peuvent former des coalitions et agir de concert [6].

Définition 2.6. *Une coalition est un sous-ensemble de joueurs : $C \subseteq I = \{1, ..., N\}$. Si C est une coalition d'un seul joueur ($C = i$), C est appelé singleton.*
Si C est la coalition formée de tous les joueurs ($C = N$), C est appelé grande coalition [6].

Un jeu est dit non coopératifs lorsque chaque joueur essaye de maximiser sa fonction d'utilité en tenant compte de la stratégie des autres, il n'est donc pas possible de former des coalitions [6].

2.3.3 Jeux finis et Jeux infinis

On dit qu'un jeu est fini si l'ensemble des stratégies X_i de chaque joueur i est fini. Un jeu est dit infini s'il existe un $i \in I$ tel que l'ensemble X_i est infini [30].

2.3.4 Jeux à information compléte et incompléte

Un jeu est est dit à information complète, si tous les joueurs connaissent la structure du jeu, c'est-à-dire : l'ensemble des joueurs, les préférences des joueurs, les règles du jeu et le type d'information de chaque joueur et l'histoire du jeu [30].

Un jeu est est dit à information incomplète si au moins un des joueurs ne connait pas parfaitement la structure du jeu.

2.3.5 Jeux à information parfaite et imparfaite

Un jeu est en information parfaite si chaque joueur connaît l'ensemble des actions choisies par tous les joueurs qui ont intervenus avant qu'il ne sélectionne sa stratégie, et qu'il connaît toutes leurs stratégies possibles. Il est le seul joueur à prendre une décision à cette étape [6, 30].

Si plusieurs joueurs choisissent leurs actions simultanément à une étape donnée, où si les joueurs ne connaissent pas toutes les stratégies des autres joueurs, le jeu est dit à information imparfaite [6].

2.3.6 Jeux séquentiels et simultanés

Un jeu séquentiel est un jeu où les joueurs interviennent les uns aprés les autres, la représentation la plus adéquate est la forme extensive [30].

Un jeu est dit simultané lorsque les joueurs choisissent simultanément leurs actions, et reçoivent ensuite leurs gains respectifs [6], la représentation la plus adéquate est la forme normale.

2.3.7 Jeux à somme nulle et non nulle

Les jeux à somme nulle sont tous les jeux où la somme "algébrique" des gains des joueurs est constante, ce que gagne l'un est nécessairement perdu par un autre [6, 30].

Un jeu est est dit à somme non nulle, si au moins pour une situation du jeu la somme des gains des joueurs n'est pas nulle [30].

2.4 Représentation des jeux

Un jeu peut être représenté de deux façons différentes :

-Les jeux sous forme normale (dite aussi stratégique).

-Les jeux sous forme extensive (dite aussi développée) [6].

2.4.1 La forme extensive

Un jeu sous forme extensive est défini par un arbre de décision décrivant les actions possibles des joueurs à chaque étape du jeu, la séquence de tours de jeu des joueurs ainsi que l'information dont ils disposent à chaque étape pour prendre leur décision.
Chaque nœud de l'arbre spécifie le joueur qui doit choisir une action à ce moment du jeu, l'information dont il dispose, ainsi que les gains engendrés.

La forme extensive permet une description "dynamique" du jeu parce qu'elle spécifie les séquences de décisions prises par les joueurs, lorsqu'un jeu implique plusieurs joueurs et de multiples choix, l'arbre peut devenir complexe à représenter. Or, lorsque les stratégies des agents sont choisies en parallèle, c'est-à-dire sans que l'un des joueurs observe les décisions des autres, comme c'est le cas dans le dilemme du prisonnier, cette construction basée sur un modèle dynamique n'est pas nécessaire. Dans ce cas, on simplifiera la présentation du jeu en utilisant une représentation sous forme normale, absolument équivalente à la forme extensive associée [6].

2.4.2 La forme normale

Un jeu sous forme normale est la donnée de l'ensemble des joueurs, de l'ensemble des stratégies pour chaque joueur $i \in I$ et des paiement associé à chaque combinaison de stratégies $x \in X$ [29].
Un jeu J sous forme normale (ou bien stratégique) s'écrit de la manière suivante :

$$J = \langle I, \{X_i\}_{i \in I}, \{U_i\}_{i \in I} \rangle \tag{2.2}$$

où :

I=$\{1,...,N\}$ représente l'ensemble des joueurs et X_i représente l'ensemble des stratégies du joueur i, $i \in I$.
Pour chaque joueur i, on définit une fonction d'utilité : $U_i : \prod_{j=1}^{N} X_j \longrightarrow \mathbb{R}$, qui à chaque ensemble de stratégies associe le gain du joueur i.

2.4.3 Forme normale des jeux sous forme extensive

A chaque jeu sous forme extensive correspond un jeu sous forme stratégique dans lequel les joueurs choisissent simultanément les stratégies qu'ils mettront en oeuvre [34].

2.5 Concepts de solution

Nous allons présenter ici les concepts de solution qui s'appliquent pour les jeux statiques non coopératifs où $U_i(x_i,x_{-i})$: est le gain du joueur i, lorsqu'il joue sa stratégie $x_i \in X_i$ et que les autres joueurs jouent leurs stratégies x_{-i}.
$C_i(x_{-i}) = \{x_i \in X_i, \ \ U_i(x_i, x_{-i}) = \sup_{y_i \in X_i} U_i(y_i, x_{-i})\}, \ \ \forall x_{-i} \in X_{-i}$, est dite règle de décision canonique pour le joueur $i, i \in I$.

2.5.1 L'équilibre de Nash

Équilibre de Nash en stratégies pures

Considérons le jeu (2.2) non coopératif à N joueurs, avec $I=\{1, ..., N\}$ l'ensemble de joueurs.

Définition 2.7. *[6] Un profil de stratégie $x^* = \{x_1^*, x_2^*, ...x_N^*\} \in X$ est un équilibre de Nash en stratégies pures si et seulement si :*

$$U_i(x_i^*, x_{-i}^*) \geq U_i(x_i, x_{-i}^*), \forall x_i \in X_i, \forall i \in I$$

Où :

$x_{-i} = (x_1, ..., x_{i-1}, x_{i+1}, ..., x_N)$

cela signifie qu'aucun joueur n'a intérêt à dévier s'il suppose que les autres joueurs ne dévieront pas non plus [6].

L'existence d'équilibre de Nash :

Théorème 2.5.1. *[31] L'équilibre de Nash en stratégie pure existe sous les conditions suivantes :*

1. $\forall i \in I$, *les ensembles* X_i *sont convexes et compacts,*
2. $\forall i \in I$, *les fonctions* $U_i(.) : S \longrightarrow \mathbb{R}$ *sont continues,*
3. $\forall i \in I$, *les fonctions* $U_i(., x_{-i}) : X_i \longrightarrow \mathbb{R}$ *sont concaves,*$\forall x_{-i} \in X_i$,

Alors le jeu (2.2) admet un équilibre de Nash en stratégie pure.

Équilibres de Nash en stratégies mixtes

Définition 2.8. *[30] Un équilibre de Nash en stratégies mixtes pour le jeu (2.2) est un ensemble de stratégies mixtes* α^* *tel que :*

$$U_i(\alpha_i^*, \alpha_{-i}^*) \geq U_i(\alpha_i, \alpha_{-i}^*), \forall \alpha_i \in \Delta_{|X_i|}, \forall i \in I$$

Où $\Delta_{|X_i|} = \{\alpha = (\alpha_1, ..., \alpha_{|X_i|}) \in \mathbb{R}^{|X_i|}, \alpha_j \geq 0, \forall j = \overline{1, |X_i|}, \sum_{i=1}^{|X_i|} \alpha_j = 1\}$

Théorème 2.5.2. *Tout jeu fini à N joueurs, admet au moins un équilibre de Nash en stratégies mixtes [5].*

2.5.2 La Dominance

Soit deux stratégies A et B envisageables par un joueur dans un jeu à information parfaite donné. B domine A si $U(B) \geq U(A)$ pour toute stratégie de l'adversaire.

Une stratégie $x_i \in X_i$ du joueur i est dite strictement dominée s'il existe une stratégie $x_i^{'} \in X_i$ telle que :

$$\forall x_{-i} \in X_{-i}, U_i(x_i, x_{-i}) < U_i(x_i^{'}, x_{-i})$$

On dira dans ce cas que x_i est strictement dominée par $x_i^{'}$ ou que $x_i^{'}$ domine strictement x_i [6].

Une stratégie $x_i \in X_i$ du joueur i est dite faiblement dominée, si :
$\exists x_i^{'} \in X_i$, telle que,

$$\forall x_{-i} \in X_{-i}, U_i(x_i, x_{-i}) \leq U_i(x_i^{'}, x_{-i})$$

Une situation $x = (x_1, x_2, ..., x_N) \in X$ est appelée équilibre en stratégies dominantes si $\forall i \in I$, chaque composante $x_i \in X_i$ est une stratégie dominante pour le joueur i, c-à-d :

$$\forall i \in I, \forall x_i^{'} \in X_i, U_i(x_i, x_{-i}) > U_i(x_i^{'}, x_{-i})$$

2.6 Jeux non coopératifs avec contraintes (Méta-jeux)

Considérons le jeu (2.2) on désigne par $C_i : X_{-i} \longrightarrow Xi$ la correspondance qui associe à chaque combinaison $x_{-i} \in X_{-i}$ des choix des $N\backslash\{i\}$ joueurs, un sous-ensemble de stratégies réalisables $C_i(x_{-i}) \subseteq X_i$ pour le joueur $i \in I$.
On associant à chaque joueur une fonction de gain U_i, on pourrait noter un jeu sous contraintes dit aussi méta-jeu pure, sous la forme normale suivante [5] :

$$J_{s.c} = \langle I, \{X_i\}_{i\in I}, \{C_i\}_{i\in I}, \{U_i\}_{i\in I}\rangle \tag{2.3}$$

Les règles de décision d'un joueur sont données par l'ensemble :

$$D_i(x_{-i}) = \{x_i \in X_i, x_i \in C_i(x_{-i}) \;\; U_i(x_i, x_{-i}) = \sup_{y_i \in C_i(x_i)} U_i(y_i, x_{-i})\} \tag{2.4}$$

Définition 2.9. *Une issue admissible pour un jeu sous contraintes (méta-jeu) est une situation $x = (x_i, x_{-i}) \in X$ telque :*

$$x_i \in C_i(x_{-i}), \forall i \in I.$$

Définition 2.10. *Une issue $x^* \in X$ cohérente par rapport aux règles de décision (2.4) i.e : $x_i \in C_i(x_{-i})$, $\forall i \in I$ est appelée équilibre social du méta-jeu (2.3) [5].*

2.7 Conclusion

Ce chapitre est dédié à la présentation des éléments fondamentaux de la théorie des jeux ainsi que quelques concepts de solution des jeux appartenant la classe de jeux non coopérative.

Chapitre 3

L'ordonnancement par la théorie des jeux

3.1 Introduction

La NP difficulté des problème d'ordonnancement fait que leurs résolution nécessite souvent l'utilisation des méta-heuristiques.Ces dernières années, plusieurs études se sont intéressés à la modélisation de ces problèmes en passant à la théorie des jeux. En effet, établir les liens entre les concepts d'équilibre pour les modèles de jeux obtenus et la solution du problème d'ordonnancement, permet de trouver une solution en prenant en compte les objectifs visés individuellement. Suite à cela ce chapitre est dédié à la présentation de quelques problèmes d'ordonnancement modélisés par la théorie des jeux.

3.2 Jeux d'ordonnancement de tâches

L'ordonnancement de tâches est l'affectation de n tâches sur m machines de telle sorte que certaines contraintes soient respectées et un objectif fixé soit atteint.
La théorie des jeux peut être utilisée pour modéliser un tel problème. En effet, J.Cohen, et al ont proposés dans [15] un modèle de jeu non coopératif pour le problème d'ordonnancement de tâches.
Pour cela, ils ont utilisés les notations suivantes :
∘ $I = \{1, \ldots, n\}$ l'ensemble des tâches à ordonnancer, chaque tâche est représentée par un joueur i.
∘ M_i l'ensemble des machines sur lesquelles peut être exécuté la tâche i. Cet ensemble représente l'ensemble des stratégie du joueur i.
∘ L'objectif de chaque joueur est la minimisation de sa date d'achèvement.

Il a été prouvé que la solution optimale du problème d'allocation des tâches correspond à l'équilibre

de Nash associé à ce modèle de jeu. Le théorème suivant met en evidence l'existence d'un tel équilibre.

Théorème 3.2.1 (Fotakis 2002, Even-Dar 2003)**.** *Dans un jeu d'allocation de tâches où les machines sont identiques, il existe au moins un équilibre de Nash en stratégie pure [29, 15].*

En définissant les variables :

$$x_{ij} = \begin{cases} 1 & \text{si la tâche i est affectée à la machine j} \\ 0 & sinon \end{cases}$$

$i = \overline{1,n}$, $j = \overline{1,m}$, m est le nombre de machines dans l'atelier.

p_i : représente la durée de la tâche i.

L_j : représente la charge de la machine j, $L_j = \sum_{i=1}^{n} x_{ij} p_i$

L'équilibre de Nash dans un jeu d'allocation de tâches sera définit comme suit :

Définition 3.1. *[5] Une affectation de n tâches est un équilibre de Nash, si pour toutes les tâches i, $i = \overline{1,n}$ on a :*

si $x_{ij} = 1$ alors $\forall k \in M_i$, on a :

$$L_j \leq L_k + p_i$$

Exemple 1

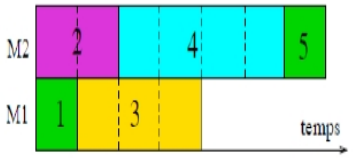
On considère que l'on doit affecter 5 tâches $(t_1, t_2, t_3, t_4, t_5)$ à 2 machines M_1 et M_2 identiques. Soit l'ordonnancement suivant :

Tâche (i)	1	2	3	4	5
Durée (p_i)	5	6	5	6	5

soit la solution suivante :

$$X = (x_{ij}) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$$

On vérifie si cette solution est un équilibre de Nash ou pas ?



La charge de la machine M_1 est : $L_1 = p_1 + p_3 = 10$.

La charge de la machine M_2 est : $L_2 = p_2 + p_4 + p_5 = 17$.

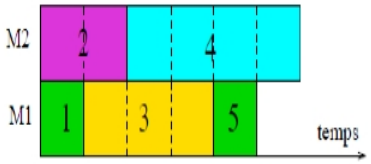
Il est clair que si le joueur 5 change sa stratégie, et que les autres joueurs maintiennent les leurs alors, on aura :

$$L_2 \geq L_1 + p_5$$

par conséquent, cette solution n'est pas un équilibre de Nash. Soit une autre solution :

$$\tilde{X} = (\tilde{x}_{ij}) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Vérifions si cette solution est un équilibre de Nash ou pas ?



comme on a déja trouvé précédemment :

La charge de la machine M_1 est : $L_1 = p_1 + p_3 + p_5 = 15$.

La charge de la machine M_2 est : $L_2 = p_2 + p_4 = 12$.

On remarque que si le joueur 4 change sa stratégie (une déviation) et que les autres joueurs gardent leurs stratégies alors, on aura :

$$L_2 < L_1 + p_4$$

par conséquent, cette solution est un équilibre de Nash pour ce problème.

Exemple 2

Le même modèle a aussi été proposé par *E.Koutsoupiasetal* dans [13] en prenant pour fonction d'utilité des joueurs la minimisation de la charge de d'une machine.

Définition 3.2. *Un placement de tâches A est un équilibre de Nash si et seulement si aucun joueur ne peut améliorer le coût unilatéralement en déplacement une tâche vers une autre machine [17].*

Définition 3.3. *Un placement de tâches A est un équilibre de Nash si et seulement si*
Pour chaque joueur i, et pour toutes les machines k,
nous avons : $c_i^{A[i]} \leq c_i^k$ *[17].*

3.3 Un modèle de jeu pour le problème d'ordonnancement

Dans [10], les auteurs se sont intéressé au problème d'ordonnancement dans le cas d'une seule machine et plusieurs jobs non préemptifs, ils ont opté pour une modélisation par un jeu non-coopératif. A base de ce modèle, il y'a eu étude de l'existence d'un équilibre de Nash pour ce jeu, des propriétés de l'espace des solutions, ainsi que l'étude des performance de ces solutions.

- **Les variables du modèle**

la figure suivante illustre les variables de bases de ce modèle :

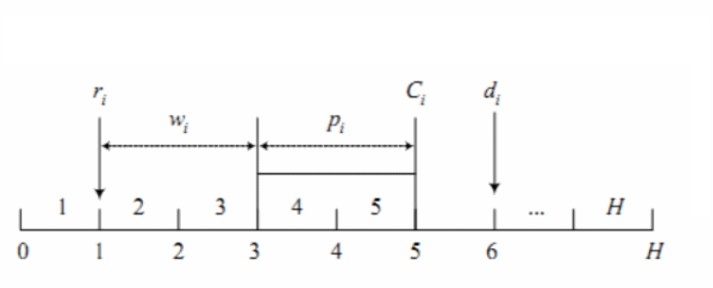


FIGURE 3.1 – Illustration des variables de bases et stratégies du job i [9].

r_i : date de début au plutôt du job i.

w_i : temps d'attente d'un job, avant le début de son exécution.

p_i : durée d'exécution du job sur la machine.

c_i : temps d'achèvement réel du job i.

d_i : date d'échéance du job i.

H : la longueur de l'horizon de l'ordonnancement.

où $p_i = C_i - r_i - w_i$

La forme normale associe est la suivante :

$$\langle I, \{X_i\}_{i\in I}, \{J_i\}_{i\in I}\rangle \tag{3.1}$$

Soient :

◦ $I = \{1, .., n\}$: l'ensemble des joueurs représentant ici les n jobs.

◦ $X_i = [0, d_i - p_i]$: l'ensemble des stratégies du joueur i, une stratégie d'un job i représente son temps d'attente w_i. Ainsi, une stratégie réalisable sera noté $w_i^{valide} \in X_i$.

Chaque job ayant choisit sa stratégie w_i, l'ordonnancement sera formé par :

$w = (w_1, w_2, .., w_n) \in W$ où $W = w_1^{valide} \times w_2^{valide} \times ... \times w_n^{valide}$.

W est un ensemble non vide, fermé et convexe.

La ressource (machine) étant unique, il y'aura forcement des issues du jeu où les jobs sont en collision.

Définition 3.4. *On dit qu'il y'a collision entre les stratégies du job i et du job j, si*

$$\triangle w_{ij} = \left|(w_j + r_j + \frac{1}{2}p_j) - (w_i + r_i + \frac{1}{2}p_i)\right| < \frac{1}{2}(p_i + p_j)$$

.

On définit alors l'ensemble

$$w_{ij}^{coll} = \{w \in W | \triangle w_{ij} < \frac{1}{2}(p_i + p_j)\}$$

L'ensemble des ordonnancement non réalisables sera alors :

$$W^{coll} = \bigcup_{i\neq j} w_{ij}^{coll}$$

et l'ensemble des issues réalisables pour ce jeu est donné par :

$$W^{valid} = W - W^{coll}$$

On peut aussi construire l'ensemble des séquencement réalisables $S(W^{valide})$ qui est une projection de l'ensemble W^{valide}. Un élément de $S(W^{valide})$ est $s(w) = (s_1(w), .., s_n(w))$, $w \in W^{valide}$, $s_i(w) \in \{1, .., n\}$ et $s_i(w) \neq s_j(w), \ \forall j, \ \forall i, \ i \neq j$.

Dans un modèle de jeu, il est nécessaire de définir la fonction *utilité* de chacun des joueurs. Ici, pour ce problème d'ordonnancement, la fonction objectif d'un job i est donnée par :

$$J_i(r_i, p_i, d_i, w_i) = f_i(r_i, p_i, d_i, w_i) + \sum_{i \neq j} W_{ij}(w). \tag{3.2}$$

$$W_{ij}(w) = \begin{cases} 0 & si \ \ w \in W^{valide} \\ +\infty & sinon. \end{cases} \tag{3.3}$$

La fonction $f_i(w_i)$ est continue monotone croissante en w_i.
Le terme $\sum_{i \neq j} W_{ij}(w)$ reflète l'influence mutuelle des choix des stratégies entre les différents joueurs. Tout choix de stratégie d'un job va forcement influencer le choix des autres jobs et donc l'ensemble des stratégies réalisables.

L'équilibre de Nash (E.N) :

Définition 3.5. *Une issue du jeu $w^* = (w_1^*, w_2^*, .., w_n^*)$, $w^* \in W$ est un équilibre de Nash, si elle vérifie la relation suivante :*

$$J_i(w_1^*, .., w_i^*, .., w_n^*) \leq J_i(w_1^*, .., w_i, .., w_n^*) \tag{3.4}$$

Face à une telle solution, aucun joueur n'a intérêt à changer de stratégie, i.e c'est une solution qui satisfait les objectifs de chaque job.

On note par $W^* = \{w^* \in W / w^* \ est \ un \ E.N\}$ l'ensemble des ordonnancement d'équilibre de Nash ($W^* \subset W^{valide}$).

Existence d'un E.N pour ce problème d'ordonnancement :

La question d'existence de l'E.N dans un jeu d'ordonnancement n'est pas toujours facile à discuter . Pour le problème traité ici, l'existence d'un terme de pénalité dans les fonctions objectifs rend difficile de garantir la continuité des fonctions $J_i(w)$ et sa concavité ou convexité.

Théorème 3.3.1. *L'ordonnancement optimal global pour notre problème* $1|r_i|\sum_{j=1}^{n}\alpha_j f_j(w_j)$ *est un équilibre de Nash pour le jeu non-coopératif (3.1).*

On note par $1|r_i|\sum_{j=1}^{n}\alpha_j f_j(w_j)$, le problème classique d'ordonnancement à une seule machine.

Nombre d'ordonnancement E.N :

Sous l'hypothèse de la monotonie croissante de $f_i(w_i)$ en fonction de w_i, on peut facilement montrer le théorème suivant :

Théorème 3.3.2. *Pour chaque* $w \in W^*$*, il existe un unique séquencement* $s(w) \in S(W^{valide})$ *et pour chaque* $s(w) \in S(W^{valide})$ *il existe au plus un ordonnancement* $w \in W^*$*.*

Ce théorème met en évidence, la relation entre l'ensemble des équilibres de Nash et l'espace des séquencements réalisables. Notons aussi que le nombre d'ordonnancements E.N ne dépasse pas $n!$

Analyse des performances des E.N :

La performance globale est mesuré par la fonction :

$$L(w) \;=\; \sum_{j=1}^{n} C_j(w) \tag{3.5}$$

Où $C_j(w)$ est le temps d'achèvement du job j lorsque la situation du jeu c'est w.

Ceci correspond au problème NP-difficile $1|r_j|\sum_{j=1}^{n}C_j$. On évalue la performance des ordonnancements E.N $w^* \in W^*$ du problème (3.1), en utilisant la fonction $L(w)$.

Posons w^{opt} l'ordonnancement optimal global, i.e celui qui minimise $L(w)$, alors, le taux de performance entre un ordonnancement qui est E.N et l'ordonnancement optimum global satisfait la relation

$$1 \leq \frac{L(w^*)}{L(w^{opt})} < n$$

Ce résultat indique que les ordonnancements E.N satisfaisant les objectifs individuels n'ont pas forcement une grande performance globale.

3.3.1 l'ordonnancement à machines parallèles

L'ordonnancement de n tâches sur m machines non identiques a fait l'objet de plusieurs travaux, plusieurs auteurs y sont intéressés en proposant des algorithmes optimisant des critères donnés. Le critère le plus fréquent est la durée total d'achèvement des tâches qu'on appelle aussi *makespan*. Dans cette section, nous allons présenter le travail de T.O.Elisha et al dans [36] qui se sont intéressés à la modélisation du problème à machines parallèles par un jeu non coopératif.

Dans ce modèle, les auteurs se sont intéressés à la résolution du problème $Q||C_{max}$ i.e les tâches seront ordonnancés sur des machines parallèles non identiques sans contraintes spécifiés tout en minimisant le *makespan*.

Les ordonnancements sont fait à l'aide des outils de la théorie des jeux et les systèmes multi-agents.

il l'ont modélisé comme suit :

Les tâches sont représentées par des joueurs, On a n tâches donc l'ensemble des joueurs sera égal à $N = \{1, 2, \ldots, n\}$, et les machines sont prise comme des stratégies i.e chaque tâche est affectée à la machine sur laquelle elle sera exécutée, On note par X_i l'ensemble de ces stratégies tel que

$$X_i = \{1, 2, \ldots, m\}.$$

Dans ce modèle les gains des joueurs est le *makespan* c'est un gain commun entre les joueurs, un ordonnancement avec un petit *makespan* est toujours préférable à un autre ordonnancement.

La fonction d'utilité est C_{max} est définit comme suit :

$C_{max} = \max C_i$, avec C_i est le temps d'achèvement (final) du job i, $i = \overline{1, n}$. Pour la résolution de ce modèle les auteurs ont fait appel aux jeux de dispersion, aux jeux de potentiel, et aux jeux à choix aléatoire.

3.3.2 Modèle de jeu pour le flow shop à trois étages

• Le modèle proposé

Dans [1], le problème $FH||C_{max}$ a été représenté par un jeu coopératif. Le problème consiste à ordonnancer n jobs sur m_i machines identiques à l'étage i, $i = \overline{1, 3}$

Les jobs arrivent à l'étage 1. Une fois que les opérations correspondantes sont achevées sur cet étage elles seront délivrées à l'étage 2 pour l'achèvement des opérations successives. De même, après l'achèvement

des opérations à l'étape 2, les jobs seront délivrés à l'étage 3 pour le traitement finale.

-Les jobs sont traités sans préemption.

l'objectif est d'affecter les jobs aux machines correspondantes et déterminer les séquences de traitement sur les machines de telle sorte que le temps d'exécution total soit minimisé.

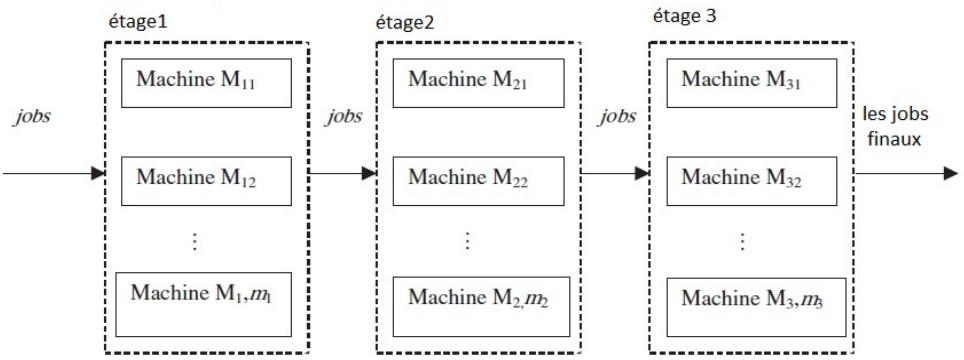


FIGURE 3.2 – La structure du système de fabrication [1]

• La modélisation par la théorie des jeux coopérative

Le problème de flow shop flexible à n jobs 3 étages sous forme d'un jeu coopératif est constitue des éléments suivants : $N = \{a_1, \ldots, a_n\}$ est l'ensemble des joueurs (tâches ou jobs).

x_j la stratégie du joueur a_j et x_j représente la machine assigné par les opérations.

V_S est la fonction d'utilité qui représente la valeur du makespan associe à l'ordonnancement tel que :

$V_S = C_{max}(x_1, x_2, \ldots, x_S)$;

où

$$C_{max}(x_1, x_2, \ldots, x_S) \tag{3.6}$$

est le meilleur makespan atteint. Dans le problème flow shop l'assignement des opérations peut se formuler par la programmation mixte en nombre entiers.

•La formulation de la programmation mixte en nombres entiers pour ce problème

Soient :

K nombre d'étages dans le système.

m_i nombre de machine dans l'étage i, $i = \overline{1,k}$;

t_i la durée de traitement de l'opération O_i, $i = \overline{1,k}$;

T_{ij} le temps d'achèvement sur la machine j, $j = \overline{1,m_i}$; à l'étage i, $i = \overline{1,k}$ avant que le job soit ordonnancé ;

CT_i durée de traitement de l'opération O_i à l'étage i, $i = \overline{1,k}$;

M un grand nombre ;

O_{ik} $i = \overline{1,n}$;$k = \overline{1,3}$ correspond à l'opération k du job i qui peut être exécutée sur une seule machine à l'étage k.

∘*les variables*

$$x_{ij} = \begin{cases} 1 & \text{si la tâche i est affectée à la machine j, i=1,2,..., k ; j=1,2,..., } m_i \\ 0 & \textit{sinon} \end{cases}$$

$i = \overline{1,k}$, $j = \overline{1,m_i}$

∘*La fonction objectif*

$$MinZ = CT_k \tag{3.7}$$

∘*les contraintes*

$$\sum_{j=1}^{m_i} x_{ij} = 1 \qquad i = \overline{1,k} \tag{3.8}$$

$$CT_{i+1} - CT_i \geq t_{i+1}, \qquad i = \overline{1,k-1} \tag{3.9}$$

$$T_{ij} + t_i x_{ij} - CT_i \leq M(1 - x_{ij}), \qquad i = \overline{1,k}; \overline{j = 1,m_i}; \tag{3.10}$$

$$x_{ij} = \overline{0,1}; \quad CT_i = \overline{1,k}; \quad j = \overline{1,k} \text{ sont non négatifs.} \tag{3.11}$$

La fonction (3.6) minimise la durée d'achèvement de la dernière opération.

La contrainte (3.7) assure que chaque opération O_i est affectée à une seule machine à l'étage i.

La contrainte (3.8) assure la succession des opérations i.e le traitement d'une opération ne commence pas avant que le traitement de la dernière ne soit achevé.

La dernière contrainte est celle d'intégrité et de la non négativité.

3.3.3 L'ordonnancement de projet par la théorie des jeux

Dans [8] les auteurs ont modelisé un projet d'ordonnancement par un jeu non coopératif, le modèle est présenté ci-dessous :

$T = \{0, \ldots, n+1\}$: l'ensembles des tâches. Par convention l'activité '0' et 'n+1' sont les activités de début et de fin respectivement, donc on a n activités.

p_i : La durée de traitment.

$\overline{p_i}$:La durée de traitement maximale .

$\underline{p_i}$:La durée de traitement minimale.

e_i : Le coût marginal de la tâche i par une unité de temps.

π : le coût de pénalité par unité de temps.

D : La date d'achèvement du projet.

$\overline{D}$: La date maximale d'achèvement du projet.

$\underline{D}$:La date minimale d'achèvement du projet.

$A = \{0, \ldots, m+1\}$: L'ensemble des joueurs ici représentant des tâches. Donc on a m joueurs car les joueurs qui représentent les noeuds de début et de fin sont supposés fictifs.

Chaque joueur A_u représente une ou un ensemble de tâches i.e T_u est l'ensemble de tâches gérés par un joueur.

$\kappa_u = \sum_{i \in T_u} \kappa_i$: Le coût fixe lors d'un traitement de durée maximale.

Le profit d'un joueur est donné par la formule suivante :

$$Z_u(p_u, D) = w_u \times \Pi(D) - y_u(p_u).$$

tel que :

$\Pi(D) = K - max(0, D - \underline{D}) \times \pi$

$y_u(p_u) = \kappa_u + \sum_{i \in T_u} (\overline{p_i} - p_i) \times e_i$.

P_u : est le vecteur des durées du joueur u. tel que u représente une où plusieurs tâches.

∘**Les joueurs** :

$A = \{0, \ldots, m+1\}$ est l'ensemble des joueurs.

∘**Les stratégies** :

les stratégies ici représentent les durées de traitement.

d'où l'issue du jeu : $S = (P_1, \ldots, P_n)$. tel que P_u est le vecteur des durées où chaque durée $p_i \in [\underline{p_i}, \overline{p_i}]$

A la fin on aura donc la date d'achèvement du projet noté par $D(S)$. Une issue S doit vérifier les deux

conditions suivantes :

1. $D(S) \in [\underline{D}, \overline{D}]$.
2. Pour chaque issue S, on associe un vecteur de profit $Z(S) = (Z_1(P_1, D(S)), \ldots, Z_m(P_m, D(S)))$ tel que $Z_u(P_u, D(S))$ doit tre positive.

Soient :

$\mathcal{S}_{TC}$ l'ensemble des stratégies qui vérifie (1).

$\mathcal{S}_{CC}$ l'ensemble des stratégies qui vérifie (2).

$\mathcal{S} = \mathcal{S}_{TC} \bigcap \mathcal{S}_{CC}$ est l'ensemble des stratégies valides qui vérifient les deux conditions citées précédemment.

La fonction d'utilité sert à minimiser la date d'achèvement D du projet.

3.3.4 Exemple illustratif

Soit la stratégie suivante :

$\mathcal{S} = (9, 7, 4, 9, 13)$ d'où $D(S) = 22$.

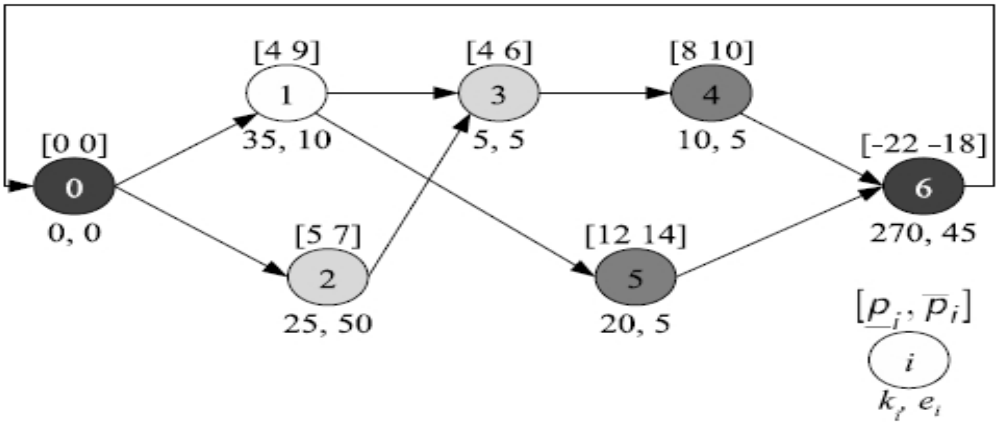


FIGURE 3.3 – Exemple d'un ordonnancement de projet [8]

dans cet exemple il y a 3 joueurs :

Le premier représente la tâche numéro (1).

Le deuxième représente l'ensemble de tâches {(2),(3)}.

Le troisième représente l'ensemble de tâches {(4),(5)}.

Soit la stratégie suivante :

$\mathcal{S} = (9, 7, 4, 9, 13)$ d'où $D(S) = 22$.

Le revenue est diviser d'une manière égale entre les joueurs (i.e $w_u = 1/3$).

On peut vérifier facilement que cette stratégie vérifie la première condition car ($22 \in [18, 22]$) le revenu de chacun est égal à 30. Par contre cette stratégie ne vérifie pas la condition (2) car $Z(S) = (-5, -10, -10)$.

Dans [8] les auteurs ont procédé par un calcul manuel pour le calcul de trois stratégies qui sont pareto optimale $\mathcal{S} = S_1^* = (6, 7, 4, 8, 13), S_2^* = (7, 7, 4, 8, 12), S_3^* = (6, 6, 4, 8, 12)$.

Les vecteurs de profits sont $Z(S_1^*) = (10, 35, 30), Z(S_2^*) = (20, 35, 25), Z(S_3^*) = (25, 0, 40)$.

les durées du projet sont respectivement $D(S_1^*) = D(S_2^*) = 19$ et $D(S_3^*) = 18$.

∘**Le calcul de l'équilibre de Nash** :

La stratégie S_3^* n'est pas un équilibre de Nash car $Z_2(S_3^*) < Z_2(S_2^*)$.

Les seules stratégies qui sont pareto optimal et un équilibre de Nash sont S_1^* *et* S_2^* avec D = 19.

3.4 Conclusion

Dans ce chapitre, nous avons présenté les problèmes d'ordonnancements qui ont été étudiés sous l'angle de la théorie des jeux à fin d'avoir les meilleurs solutions (EN), à savoir la complexité des problèmes.

Chapitre 4

Un méta-jeu pour le problème de job shop

4.1 Introduction

Dans ce chapitre, nous allons présenter une modélisation du problème d'ordonnancement $J||C_{max}$ par un méta-jeu, puis nous allons proposer une démarche pour la recherche de son équilibre social.

4.2 Le modèle proposé

Nous allons présenter les différentes notations utilisées pour la formulation du jeu sous contraintes représentant le problème du job shop [5].

$J_i(1 \leq i < n)$: représente le job i où n est le nombre de job à réaliser.

I : L'ensemble des joueurs (travaux).

X_i : L'ensemble de stratégie du joueur représentant le job J_i.

x_i : La stratégie de J_i.

m : Le nombre de machines dans l'atelier de production.

n_i : Le nombre d'opération du job i.

O_{ij} : L'opération j du job i.

p_{ij} : Durée d'exécution de l'opération O_{ij}.

x_{ij} : La date de début de l'opération O_{ij}.

$M_{ij} \in \{1, \ldots, m\}$ $1 \leq j \leq n_i$: L'indice de la machine sur laquelle s'exécute l'opération O_{ij}.

$y_{ij}^{i'j'} = 1$ si l'opération O_{ij} précède l'opération $O_{i'j'}$ sur la même machine, 0 sinon.

$y_{ij}^{i'j'}$ est définie pour tous les couples d'opérations (O_{ij},$O_{i'j'}$) partageant la même ressource.

I : L'ensemble des joueurs (jobs).

$X_i = ([0, +\infty[)^{n_i}$ est L'ensemble de stratégie du job i, tel que $x = (x_1, x_2, \ldots, x_i, \ldots, x_n)$, et

$x_i = (x_{i1}, x_{i2}, \ldots, x_{in_i})$

x_i : La stratégie de J_i.

F_i : La fonction de paiements du joueur $i \in I$.

La forme normale associée à ce meta-jeu est la suivante :

$$J_{sc} = \langle I, \{X_i\}_{i\in I}, \{C_i\}_{i\in I}, \{F_i\}_{i\in I}\rangle \tag{4.1}$$

Où :

I représente l'ensemble des joueurs à chaque job correspond un joueur i.

$X = (x_1, x_2, \ldots, x_n)$ est l'ensemble des stratégies du joueur $i \in I$. Elle consiste à fixer une date de début à chacune des opérations qui le constituent, i.e $x_i = (x_{i1}, \ldots, x_{in_i}) = (x_{ij})_{j=\overline{1,n_i}}, \quad x_{ij} \in \mathbb{R}^+$

La fonction de paiement pour chaque joueur $i \in I$ est définie par la relation suivante :

$$F_i(x) = F_i(x_1, \ldots, x_n) = x_{ij} + d_{in_i} \tag{4.2}$$

où $x = (x_1, \ldots, x_n)$ est une issue du jeu (4.1).

$C_i = C_i(x_i)$ est l'ensemble des contraintes auxquelles le joueur i est soumis lors du choix de sa stratégie.

$$C_i(x_{-i}) = x_i = (x_{ij})_{j=\overline{1,n_i}} \in X_i \quad / \quad (4.3) \quad et \quad (4.4) \quad \text{vérifiés}, \forall i \in I$$

$$x_{i,j+1} \geq x_{ij} + d_{ij}, \quad si \quad M_{ij} \neq M_{i'j'}, \quad j = \overline{1, n_{i-1}} \tag{4.3}$$

$$x_{ij}(1 - y_{ij}^{i'j'}) + y_{ij}^{i'j'} x_i'j' \geq (1 - y_{ij}^{i'j'})(x_i'j' + d_i'j') + y_{ij}^{i'j'}(x_{ij} + d_{ij}), \tag{4.4}$$

$$\forall i' \in I/i, \quad j' = \overline{1, n_i'}, \quad M_{ij} \neq M_i'j'$$

◦ **Signification des contraintes** :

• La contrainte (4.3) : cette équation décrit la contrainte de précédence entre les opérations d'un même job. Une opération ne peut commencer à être exécutée avant la fin du traitement de l'opération précédente du même job.

• La contrainte (4.4) : Elle exprime le partage de ressource entre deux opérations O_{ij} et $O_{i'j'}$ passant sur la même machine, i.e qu'une machine ne peut pas traiter deux opérations simultanément (contrainte

de disponibilité).

Remarque 4.1 : Lors du traitement des tâches :

-La préemption des opérations n'est pas autorisée.

-Il n'y a pas de date d'échéance sur les tâches.

La résolution du jeu (4.1) revient à rechercher une issue $x \in X$ du jeu qui soit un équilibre social. i.e Notons que la recherche d'un équilibre social pour le meta jeu (4.1) revient à trouver une issue du jeu qui soit admissible pour ce jeu et qui soit un équilibre de Nash sur l'espace des solutions admissibles.

4.3 La recherche d'un équilibre social

Le jeu d'ordonnancement étant défini on passe à la recherche de son équilibre social. Nous proposons les deux étapes suivantes :

Étape (1) : Elle consiste à rechercher une solution réalisable $x = (x_i, x_{-i})$ pour laquelle $x_i \in C_i(x_{-i})$, $\forall i \in I$. C'est à dire, cette solution doit vérifier les contrainte (4.3) et (4.4).

Étape (2) : A partir de la solution trouvée en (1) qu'on note $\tilde{x}^0$, on construit progressivement une autre solution x^* qui soit un équilibre de Nash tout en restant dans le domaine réalisable. On dira alors que cette solution représente l'équilibre social du jeu (4.1). En effet, à une itération donnée j, chacun des n joueurs doit vérifier si en déviant de sa stratégie actuelle $\tilde{x}_i^{(j)}$, sachant que les autres joueurs maintiennent toujours les leurs conduit à l'amélioration de son propre gain tout en gardant l'issue réalisable. Si on se retrouve dans une situation où plusieurs joueurs ont intérêt à dévier de leurs stratégies, seul le joueur pour lequel l'amélioration est plus grande pourra valider sa déviation. On obtient une nouvelle solution réalisable $\tilde{x}^{(j)}$. A ce niveau, si $\tilde{x}^{(j)} = \tilde{x}^{(j-1)}$ alors $\tilde{x}^{(j)}$ représente l'équilibre social recherché.

La règle SPT :

En mesure de réduire le *makespan* pour chaque job, on propose de suivre une règle dite la règle SPT qui tente de donner la priorité au job qui a la somme des durées la plus courte. Ensuite exécuter les opérations d'un même job suivant par ordre lexicographique croissant.

◦ **La procédure proposée**

• **Entrée** m le nombre de machine.

n le nombre de job.

n_i le nombre d'opérations dans chaque job i.

d_{ij} Les durées opératoires de chaque tâche. $\forall i = \overline{1,n}\,; \forall j = \overline{1,n_i}$.

M_{ij} Les affectations des opérations aux machines.

• **Sortie** L'ordre du passage des opérations sur chaque machine et le calcul des dates de début de chaque opération de chaque job.

Début

——-

| Pour k de 1 à m faire

| 1. Calculer la somme des durées $\sum_{j=1}^{n_i} d_{ij}, \quad i = \overline{1,n}$ des jobs distincts.

| 2. Déterminer l'ordre d'exécution des opérations suivant la règle de Rank (définie précédemment) sur chaque machine.

| 3. A (t=0), commencer l'exécution sur chaque machine toute en sachant que celle-ci essaye de traiter l'opération suivante (continuer l'exécution) tout en respectant la condition de précédence.

| 4. Calculer la date de début de chaque opération de chaque job, on obtiendra à la fin d'exécution les dates d'achèvement de chaque job.

| 5. A ce niveau nous aurons une première solution réalisable $\tilde{x}^{(0)}$. Nous devons vérifier si celle-ci est un équilibre social ou pas.

| 6.Répéter

| ◦ pour chaque joueur $i \in I$

| Si $\exists \;\; i \quad \in I \; x_i \in X_i, \quad f_i(x_i, \tilde{x}_{-i}^{(j-1)}) < f_i(\tilde{x}_i^{(j-1)}, \tilde{x}_{-i}^{(j-1)})$

| et $x_i \in C_i(\tilde{x}_{-i}^{(j-1)})$ alors $\tilde{x}^{(j-1)}$ est un équilibre social.

| Sinon $\tilde{x}^{(j)} = (x_i, \tilde{x}_{-i}^{(j-1)})$

| j = j+1.

Fin

——

4.3.1 Application numérique

Considérons un problème job shop 3×3 qui consiste à l'exécution de trois jobs sur trois machines. Pour chaque job i on associe trois opérations numérotées par des entiers, les durées opératoires et les machines utilisées pour exécuter les opérations sont fournies dans le tableau suivant :

TABLE 4.1 – Les durées opératoires et les affectations machines des opérations

Jobs	Opérations	N°	M_1	M_2	M_3
J_1	O_{11}	1	6	X	X
	O_{12}	2	X	12	X
	O_{13}	3	X	9	X
J_2	O_{21}	4	X	16	X
	O_{22}	5	14	X	X
	O_{23}	6	5	X	X
J_3	O_{31}	7	X	X	22
	O_{32}	8	X	X	18
	O_{33}	9	X	X	9

L'affectation des opérations aux machines est donnée dans la figure suivante

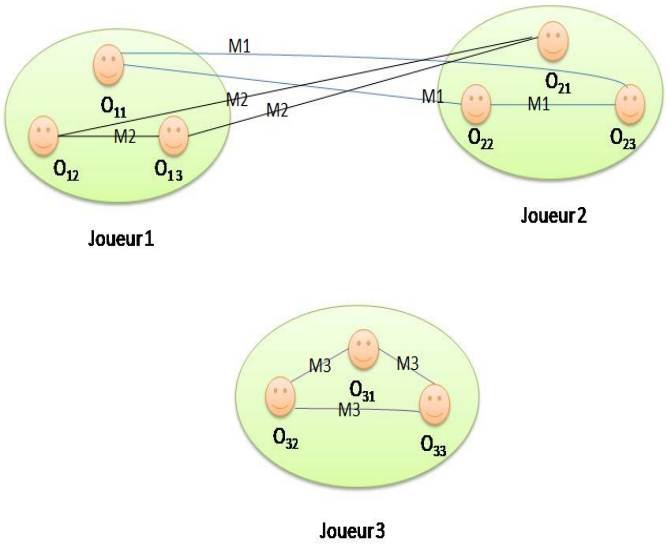


FIGURE 4.1 – Schéma représentant les affectations des opérations aux machines

Deux opérations liées par une arrête partagent une même ressource.

En implémentant la procédure proposée sous l'environnement MATLAB nous avons eu les résultats synthétisés dans le tableau ci-dessous. Ce tableau spécifie les dates de début et de fin de chaque opération.

TABLE 4.2 – Les dates de début et de fin de chaque opération

machines	opérations	Date de début	Date de fin
M_1	1	0	6
	5	16	28
	6	28	37
M_2	4	0	16
	2	16	30
	3	30	35
M_3	7	0	22
	8	22	40
	9	40	49

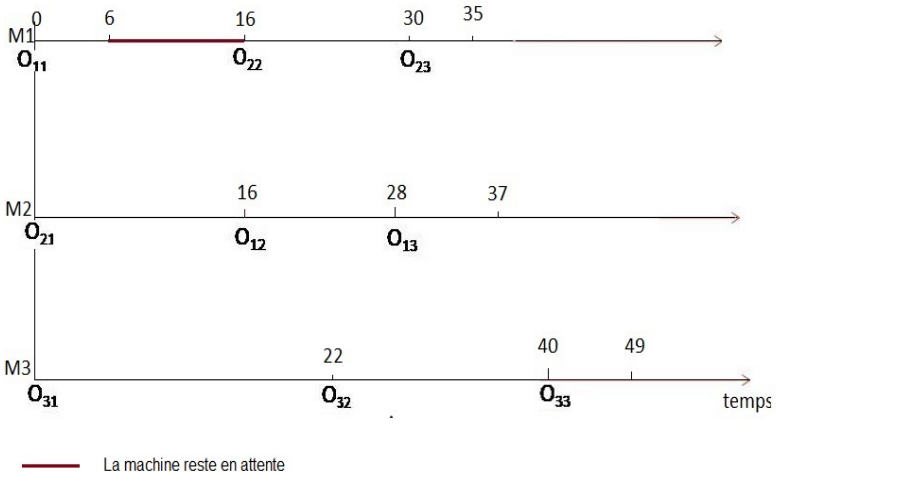


FIGURE 4.2 – Les dates de début de chaque opération

Le *Makespan* de cette solution est égal à 49 unités de temps.

◦ **La recherche d'un équilibre de Nash**

Un concept d'équilibre implique un mécanisme particulier de coordination des stratégies individuelles. Dans l'équilibre de Nash, chaque joueur cherche à améliorer son propre gain unilatéralement.

Nous savons qu'un équilibre est constitué d'une combinaison de choix stratégiques c'est-à-dire qu'aucun joueur ou groupe de joueurs n'aura d'incitation à modifier ses actions s'il suppose que ses rivaux ne modifieront pas non plus les leurs.

Dans le jeu (4.1), on tombe sur la situation dans laquelle chaque joueur arrête seul ses choix stratégiques sans consulter les autres joueurs (i.e un jeu non coopératif).
La solution du jeu (4.1) est donnée comme suit :
$x = (x_1, x_2, x_3)$ avec $x_1 = (x_{11}, x_{12}, x_{13})$ est la stratégie du joueur 1 (job1).
$x_2 = (x_{21}, x_{22}, x_{23})$ est la stratégie du joueur 2 (job2).
$x_3 = (x_{31}, x_{32}, x_{33})$ est la stratégie du joueur 3 (job3).

L'application numérique de la procédure donné ci-dessus nous donne les résultats suivants :
$x_1 = (0, 16, 28)$, $x_2 = (0, 16, 30)$, $x_3 = (0, 22, 40)$ et la date d'achèvement pour chaque job est donné par $f_1 = 37$, $f_2 = 35$ et $f_3 = 49$.

A partir de cette solution, on ne peut trouver aucun joueur qui pourrait améliorer son gain en modifiant au moins une date de début d'une des opérations qu'il gère sachant que ses adversaires maintiennent leurs choix et sans violer aucune contrainte. Par conséquent le seul équilibre social du jeu (4.1)
Donc le seul équilibre de Nash du jeu (4.1) est la solution réalisable $x = (x_1, x_2, x_3)$, où $x_1 = (0, 16, 28)$, $x_2 = (0, 16, 30)$, $x_3 = (0, 22, 40)$
Dans [26] les auteurs ont proposés une résolution du problème job shop en utilisant les algorithmes génétiques, avec les mêmes données prises précédemment nous regroupons leurs résultats et les nôtres dans le tableau suivant :

TABLE 4.3 – La présentation de nos résultats et ceux fournis par les algorithmes génétiques

jobs	date de fin obtenu par les A.G	date de fin obtenu par la procedure proposée
J_1	49	37
J_2	34	35
J_3	49	49

4.3.2 Comparaison des résultats

Si on procède à une comparaison de la solution fourni par la procédure proposée et celle obtenu dans [26] en appliquant un algorithme génétique, nous constatons qu'en terme de plus grande date d'achèvement des travaux ; les deux méthodes fournissent le même résultats.
D'un point de vu individuel, l'analyse révèle que le joueur 1 correspondant au job 1 a une date d'achèvement nettement meilleure avec notre procédure, contrairement au joueur 2 dont le gain est moins bon d'une unité de temps, et les résultats sont identiques pour le $3^{\hat{e}me}$ joueur. Il faut comme même noter que le modèle considéré par $Merhoumetal$ dans [26] n'exige pas de contraintes de game opératoire pour certaines opérations d'un même job, ce qui influt forcement sur la qualité de la solution.

4.4 Conclusion

Dans ce chapitre nous avons présenté un modèle de jeu non coopératif pour le problème job shop, ensuite nous avons proposé une procédure pour la recherche d'un équilibre social que nous avons implémenté sous l'environnement MATLAB à la fin on a fait une comparaison entre nos résultats et les résultats fourni par les algorithmes génétiques.

Conclusion Générale

Le problème d'ordonnancement d'atelier de type job shop fait parti des problèmes les plus étudiés. En terme de complexité, il figure dans la classe des problèmes NP-difficile, d'où l'utilisation des méthodes exactes pour leur résolution est inadéquate, ce qui justifie le recours aux méthodes approchées.

Dans le présent travail nous nous sommes intéressés à la représentation des problèmes d'ordonnancement par la théorie des jeux, en établissant une synthèse de quelques travaux ayant déjà abordé cette problématique. Nous avons également présenté un jeu sous contraintes modélisant le problème du job shop. Notons que l'équilibre social de ce méta-jeu correspond à la meilleure solution pour ce problème d'ordonnancement. Dans la littérature, très peu de travaux se sont intéressés aux méta-jeux et nous n'avons pas pu trouvé d'algorithme pour le calcul de l'équilibre social. Par conséquent, nous avons proposé un algorithme qui s'appuie sur la définition de cet équilibre pour le calculer tout en prenant en compte les contraintes de notre problème d'origine (contraintes de séquencement et de gammes opératoires). Cet algorithme a été implémenté sous l'environnement Matlab puis testé sur une instance de Job shop à 3 job (3 joueurs) et les résultats ont été comparé à ceux fourni par un algorithme génétique pour le même jeu de données.

En guise de perspective, il serait intéressant de :

Modifier l'algorithme en recherchant une issue réalisable du méta jeu par l'application d'une méta-heuristique, puis raffiner cette solution jusqu'à l'atteinte d'un équilibre social correspondant à la solution du problème de jobshop. Prise en compte de l'indisponibilité des machines dans le modèle de jeu, telle que les pannes qui peuvent survenir d'une manière aléatoire.

Intégrer la programmation par contraintes dans le processus de recherche de l'équilibre social du jeu d'ordonnancement proposé.

Bibliographie

[1] A.Babayen et D.He, *Solving the n-job 3 stage flexible flow shop sheduling problem using an agent-based approach*. Int. j. prod. res., vol. 42, no. 4, pages 777–799, 2004.

[2] A.Derbala, *Cours sur l'ordonnancement dans les ateliers : Formulation des problèmes d'ordonnancement*. Université de Blida, 2006.

[3] A.Gorine, *Ordonnancement des systèmes flexibles avec contrainte de blocage*. Thèse de doctorat, Université de Paul-verlaine de metz, 2011.

[4] A.Kouider et B.Bouzouia, *Approche multi-agents basée sur le récuit simulé pour le problème d'ordonnancement distribué*. 4th international conférence of computer integrated manufacturing CIP 2007.

[5] A.Ouali et K.Touahri, *L'ordonnancement sous l'angle de la théorie des jeux*. Memoire de master, Université de Béjaia, 2013.

[6] B.Elise, *Modélisation des interactions entre agents rationnels : les jeux booléens*. Thèse de doctorat, Université Paul Sabatier de Toulouse3, 2007.

[7] C.Andrea, L.T.Van et M.Guillaume, *Cours sur l'optimisation par colonies de fourmis*. 2006.

[8] C.Briande, J.C.Billaut, *Cooperative project scheduling with controllable processing times : a game theory frameworke*. Université de Toulouse, Universite Francois Rabelais Tours, 2011.

[9] CJ.Wang et YG.XI, *Non coopérative game on sheduling : the single machine case*. Université de shanghai, 2005.

[10] CJ.Wang, XI.YG, *Modeling and analysis of single machine scheduling based on noncooperative game theory*. Acta Automatica Sinica, Vol 31, N°4, 2005.

[11] D.Ginet Ramirez Rios, Claudia Marcela, *Game théoric approaches to parallel machine scheduling*. Undergraduate Project as a requisite for graduation, Industrial Engineering Department Barranquilla, Colombia, 2007.

[12] E.T.O.Opiyo, E.Ayienga, K.Getao, W.Okello-Odongo, B.Manderick, A.Nowe, *Game theoretic multi-agent systems scheduler for parallel machines*. International Journal of Computing and ICT Research, Special Issue Vol. 1, No1, pp. 21-27, 2008.

[13] F.Pascual, *Optimisation dans les réseaux : de l'approximation polynomiale à la théorie des jeux*.Thèse de doctorat, Université Evry Val D'essonne, 2006.

[14] E.T.O.Opiyo, E.Ayienga, K.Getao, W.Okello-Odongo, B.Manderick, A.Nowe, *Game theoretic multi-agent systems scheduler for parallel machines*. International Journal of Computing and ICT Research, Special Issue Vol. 1, No1, pp. 21-27, 2008.

[15] J.Cohen, D.Barth, O.Bournez, O.Boussaton, *Apprentissage d'équilibre de Nash pour l'ordonnancement de taches*. Séminaire sur le projet INRIA CASSIS du laboratoire LORIA, CNRS, Versailles, France, 2008.

[16] J.Carlier, E.Pinson, *A Branch and bound method for solving the job shop*, 1986.

[17] J.Cohen et O.Bournez, *Theorie des graphes, et l'approximation pour le routage, la coloration et l'apprentissage d'équilibres*. PRISM, CNRS, Versailles, France, 2008.

[18] Inès Alaya, Christine Solnon, Khaled Ghédira, *Optimisation par colonies de fourmis pour le problème du sac à dos multidimensionnel*. 1^re soumission à Technique et science informatiques, 2005.

[19] G.Vigeral, *Cours de Théorie des Jeux*. 2012.

[20] Gotha, *Les problèmes d'ordonnancement*. Revue française d'automatique, d'informatique et de recherche opérationnelle, tome 27, pages 77-150, 1993.

[21] G.Zhou, P.Jiang et G.Q-Huang, *A game-theory approach for job sheduling in networked manufacturing*. Int, J.Adv-Manuf-Technol, pages 972-985, 2009.

[22] G.Christoulao, E.Kootsoopias et A.Nanavati, *Coordination mecanisms dans proceeding of the 31st international colloquim on automata, languages, and paogramming (ICALP)*, volume 3142 de LNOS, pages 345-357, Spring, 2004.

[23] H.Boukef Ben Othman, *Sur l'ordonnancement d'atelier job-shop flexibles et flow-shop en industries pharmaceutiques*. Thèse de doctorat, Ecole centrale de Lille et l'école Nationale d'ingénieurs de Tunis, 2009.

[24] H.Habiba, *Planification et ordonnancement en temps réel d'un job-shop en utilisant l'intélligence artificielle*. Mémoire de Magister, Université de Tlemcen, 2012.

[25] K.Leyton-Broon, Y.Shoham, *Essentials of game théory :A consiste, multidixiplinary introduction.* Morgan and Claypool publishers, 2008.

[26] K.Merhoum et M.Djeghaba, *Algorithme génétique pour le problème d'ordonnancement de type job-shop.* 4th International Conference on Computer Integrated Manufacturing CIP , 2007.

[27] K.Mebarek, *Utilisation des stratégies métaheuristiques pour l'ordonnancement des ateliers de type job-shop.* Mémoire de Magister, Université de Batna, 2008.

[28] L.Wei, N.Xiu, *Computing Nash equilibria based on heuristic search methods.*Journal of Brijing jiaotong University, 31(3), pages 58-62, 2007.

[29] M.S.Radjef, *cours sur la théorie des jeux et l'optimisation multicritère.* 1ère anée post graduation, Université A/Mira. Béjaia, 2010.

[30] M.Rouha et Yaich.Mebrka, *Resolution du problème d'éffécacité énérgitique par la théorie des jeux.* Mémoire de Master, Université A/Mira. Béjaia, 2013.

[31] N.Khimoum, *Résolution numérique d'un jeu bi-matriciel mulicritère.* Mémoire de magister, Université De Béjaia, 2006.

[32] S.M.Douiri, S.Elbernoussi et H.Lakhbab, *Cours des méthodes de résolution exactes heuristiques et métaheuristiques.* Université Mohammed V, Faculté des Sciences de Rabat

[33] S.Khalouli, *Méta-heuristiques à base de modèles : Applications à l'ordonnancement d'atelier flow shop hybride monocritère et multicritère.* Thèse de doctorat, Université de Reims Champagne-Ardenne, 2010.

[34] S.Konieczny, *Introduction à la théorie des jeux.* Université d'Artois, pages 15, 2003-2004.

[35] S.OURARI, *De l'ordonnancement déterministes à l'ordonnancement distribué sous incertitudes.* Thèse de doctorat, Université de Toulouse3 -paul sabatier, 2011.

[36] T.O.Elisha et al, *Game Theoretic Multi-Agent Systems Scheduler for Parallel Machines.* Université de Nairobi.

[37] T.Chaari, *Un algorithme génétique pour l'ordonnancement robuste : application au problème du flow shop hybride.* Thèse de doctorat, Université de Valenciennes et du Hainaut-Cambrésis, 2010.

Résumé

Ce mémoire s'intéresse aux problèmes d'ordonnancement du point de vue théorie des jeux. Après avoir fait une synthèse des travaux ayant abordés cette problématique, nous nous somme intéressé au problème de job shop en le modélisant sous forme d'un jeu non coopératif avec contraintes (méta-jeux).

La résolution de ce jeu revient à retrouver son équilibre social. Ce dernier correspond à la meilleure solution pour le problème du job shop. Notons que dans la littérature, très peu de travaux se sont intéressés aux méta-jeux et nous n'avons pas pu trouver d'algorithme les résolvant. Par conséquent, nous avons proposé un algorithme fondé sur sa définition tout en respectant les contraintes de notre problème d'origine et nous l'avons testé sur une instance à 3 jobs et comparé les résultats obtenus à ceux fourni par la résolution en utilisant un algorithme génétique.

Mots clés : Ordonnancement, Équilibre de Nash, Job shop, Équilibre social, Théorie des jeux, Méta-jeux, Heuristiques, Makespan.

Abstract

This thesis focuses on scheduling problems in a point of view of game theory. After synthesis of works that discussed this problematic, we have been interested to job shop problem by modeling it in a form of a non cooperative game with constraints (meta-game).

The resolution of this game comes to find its social equilibrium. The latter correspond to the best solution for the job shop problem. In the literature, very little works have been interseted to meta game and we could not find an algorithm that resolve it. Consequently, we proposed an algorithm based on its definition by respecting constraints of our problem and we have tested it on three jobs and compared the obtained results to those provided by the resolution using a genetic algorithm.

Key words : Scheduling, nash equilibrium, Job shop, social equilibrium, Game theory , Meta-game, Heuristic, Makespan.