Ordonnancement stochastique avec impatience

Alexandre Salch

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathématiques-Informatique**

Arrêté ministérial : 7 août 2006

Présentée par

Alexandre SALCH

Thèse dirigée par **Jean-Philippe GAYON**
et co-encadrée par **Pierre LEMAIRE**

préparée au sein de **G-SCOP (Grenoble Science pour la Conception et l'Optimisation de la Production)**
et de **MSTII (Mathématiques, Sciences et Technologies de l'Information, Informatique)**

Ordonnancement stochastique avec impatience

Thèse soutenue publiquement le **29 novembre 2013**,
devant le jury composé de :

Mr Bruno GAUJAL
Directeur de recherche, INRIA Montbonnot, Examinateur

Mr Jean-Philippe GAYON
Maître de conférence, Institut polytechnique de Grenoble, Directeur de thèse

Mr Alain JEAN-MARIE
Directeur de recherche, INRIA Sophia Antipolis - Méditerranée, Rapporteur

Mr Pierre LEMAIRE
Maître de conférence, Institut polytechnique de Grenoble, Co-Encadrant de thèse

Mr Philippe NAIN
Directeur de recherche, INRIA Sophia Antipolis - Méditerranée, Président

Mr Christophe RAPINE
Professeur, Université de Lorraine, Rapporteur

Table des matières

Chapitre 1

Introduction

Le terme de *recherche opérationnelle* est apparu pour la première fois au milieu du siècle dernier, lorsqu'une équipe a été fondée dans le but de résoudre des problèmes d'optimisation pendant la seconde guerre mondiale. L'adjectif *opérationnel* caractérise les premiers travaux de recherches de cette discipline, qui consistaient à résoudre des problèmes appliqués, tels que la répartition de radars de surveillance ou encore la planification et la gestion des convois de ravitaillement. Par la suite, cette discipline, ainsi que ses champs d'applications, se sont élargis avec l'utilisation généralisée de l'informatique, des systèmes d'informations et l'explosion des capacités de calcul des ordinateurs. De nos jours, les systèmes de production de biens ou les plateformes de services tendent à devenir de plus en plus complexes, pour s'adapter aux besoins de chaque client, mais aussi pour réduire leurs coûts de fonctionnement. La recherche opérationnelle voit dans ces situations des problèmes d'optimisation et d'aide à la décision qui viennent enrichir ses domaines de recherche. Cette thèse s'inscrit quant à elle à l'intersection de l'ordonnancement et du contrôle optimal de files d'attente.

Dans ce manuscrit, nous allons nous intéresser plus particulièrement à des problèmes d'ordonnancement dans lesquels intervient un phénomène bien connu de toutes celles et ceux qui ont eu à attendre dans une file d'attente : l'impatience. L'impatience pourra dans certains cas être vue dans le sens conventionnel du terme, comme une personne dans une file d'attente qui considère avoir attendu trop longtemps, ou encore comme une date de livraison à laquelle un bien aurait dû être produit. Nous verrons que si dans certains cas ce phénomène a été abondamment étudié, il n'en reste pas moins difficile à aborder lorsque certains paramètres du système ont des variations aléatoires. Dans le problème étudié, des tâches deviennent disponibles au cours du temps. Elles patientent dans une file d'attente pour un service effectué par une machine unique. Si au bout d'un certain temps la tâche n'a pas été servie, elle s'impatiente et un coût est imputé au système (voir figure 1). Selon les cas, la tâche peut quitter définitivement le système suite à cette impatience. Notre objectif est alors de décider quelle sera la prochaine tâche à traiter afin de minimiser les coûts d'impatience.

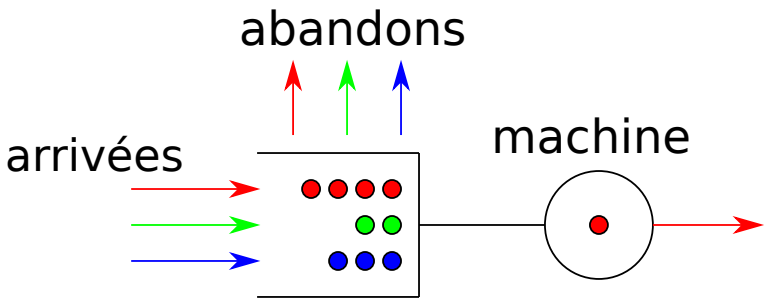


FIGURE 1 – Problème considéré

Ce document présente nos résultats sur ce sujet, comme par exemple des règles de service optimales. Avant cela, nous allons présenter les modèles et techniques sur lesquels nous nous sommes appuyés, issus de la littérature sur l'ordonnancement, mais aussi de celle sur la théorie des files d'attente. Par conséquent, même si les références nécessaires seront données au cours du document, la compréhension de ce dernier sera plus facile si le lecteur est familier avec au moins une de ces deux disciplines. Nous renvoyons d'ores et déjà le lecteur à des ouvrages détaillants les principaux modèles et résultats en ordonnancement (Pinedo 2008, Brucker 2007), ainsi qu'en théorie des files d'attente (Cooper 1972, Gross et Harris 1998).

1.1 Ordonnancements avec impatience

1.1.1 Contexte

Dans les problèmes d'ordonnancement, on doit exécuter le plus efficacement possible des tâches en ayant à notre disposition des ressources limitées et en respectant certaines contraintes. Si nous considérons l'exemple d'une chaîne d'assemblage de véhicules, les ressources disponibles peuvent être de deux types : les machines qui effectuent des tâches automatisées, ou les opérateurs chargés de monter des pièces à la main. Dans cet exemple, les tâches correspondent aux différentes opérations nécessaires pour produire un véhicule. Ensuite, ce système doit respecter certaines contraintes. Ces contraintes peuvent être de différents types, par exemple les opérateurs doivent respecter une durée légale de travail, le volant doit être monté après le tableau de bord... Enfin, se pose la question de l'efficacité de l'ordonnancement, quels sont le ou les critères qui vont au mieux décrire les performances de l'ordonnancement. Pour le problème d'assemblage de véhicules, doit-on chercher à maximiser le nombre de véhicules total qui peut être produit au cours d'une journée, ou alors chercher à minimiser le coût de la main d'œuvre nécessaire pour assembler ces véhicules, ou encore une combinaison de différents critères ? Il convient donc de proposer un critère d'optimisation correspondant au mieux à la nature du problème.

Le contrôle optimal de files d'attente peut aussi être vu comme un problème d'ordonnancement. En effet, comme en ordonnancement, les files d'attente se créent naturellement

lorsque des clients, ou des tâches, demandent l'accès à la même ressource dans un intervalle de temps réduit. Là encore, ces ressources peuvent avoir des caractéristiques différentes. Certains chargés de clientèle répondent aux besoins des clients plus au moins rapidement, d'autres encore sont spécialisés dans le traitement d'un certain type de demande. Par exemple, dans les centres d'appels, les demandes des clients sont filtrées pour être envoyées soit vers un service technique, soit vers un service commercial.

Le phénomène souvent associé aux files d'attente et qui fait le sujet de cette thèse est l'impatience. En effet, lorsqu'un client considère qu'il a passé trop de temps à attendre pour le service demandé, on dit qu'il s'impatiente. La manifestation de cette réaction varie d'un individu à l'autre mais nous les classerons par la suite dans deux catégories distinctes.

Impatience sans abandon : Dans ce cas, le client trouve qu'il a patienté trop longtemps par rapport au service qu'il a demandé mais il reste toutefois dans la file d'attente. Cette information peut par exemple être obtenue par le biais d'un questionnaire de satisfaction. Ce cas de figure se produit surtout pour des services ou des biens qui font face à une pénurie ou encore à un monopole. Par exemple lorsqu'un client achète une voiture neuve, il n'est pas inhabituel qu'il ait à attendre plusieurs mois avant d'obtenir le modèle qu'il avait demandé. Cette situation décrit le phénomène de pénurie. En effet, les véhicules étant pour la plupart fabriqués à la demande du client pour respecter exactement le panel d'options qu'il a choisies, cela entraîne un long délai de production. Dans ce cas, le client va éventuellement négocier un rabais si les délais de livraison ne sont pas respectés. Il s'impatiente, mais va rarement annuler sa commande pour se tourner vers un concurrent ; il ne va donc pas abandonner. La situation de monopole se produit par exemple à un péage autoroutier lors des heures de pointe. Les véhicules forment des files d'attente à chaque portique de péage, les automobilistes peuvent s'impatienter mais ils n'ont pas d'autre choix que de passer par ce péage. Là encore, ils peuvent s'impatienter mais n'abandonneront pas.

Impatience avec abandon : Dans ce cas, les clients qui s'impatientent quittent immédiatement le système. Cela se produit lorsque la ressource demandée est accessible à un faible coût pour le client ou lorsque le nombre de demandes pour cette ressource varie fortement. Dans ce cas, le client peut avoir intérêt à abandonner sa demande pour la reformuler plus tard. C'est ce qui se produit souvent dans les centres d'appels téléphoniques. Les clients qui s'impatientent raccrochent, car ils n'ont pas été servis assez vite.

1.1.2 Importance des abandons

Comme l'explique Zeltyn (2004), l'objectif des managers de ces systèmes est de trouver le meilleur équilibre entre la qualité de service pour ses clients et les coûts de fonctionnement de la structure. En effet, ne pas prendre en compte l'abandon dans les files d'attente peut conduire à un sur-effectif, car il y a plus de clients qui appellent qu'il n'y a de clients qui

joignent effectivement les chargés de clientèle, précisément à cause des abandons. Dans sa thèse, Zeltyn cite des études montrant l'importance de la bonne gestion des effectifs dans les centres d'appels, tant c'est un poste de dépense important dans ces systèmes.

Pire encore, si l'on souhaite minimiser le temps de réponse moyen sans tenir compte de ceux qui abandonnent, on choisira une politique de service du type LIFO (*Last In - First Out* : dernier arrivé - premier servi), comme le montrent Kallmes et al. (1989) et Zhao et al. (1991, 1993). Par conséquent, si le centre d'appel est en surcharge, on peut séparer les clients en deux catégories : ceux qui ont instantanément accès aux ressources et ceux qui attendent jusqu'à quitter le système. Ce choix de service s'avère être à l'opposé de ce que l'on attend généralement d'un système équitable d'accès à des ressources, où les clients passent en moyenne la même durée à attendre pour un service. Cependant, il existe des systèmes pour lesquels la politique LIFO conduit effectivement à une meilleure qualité de service pour les clients, comme par exemple l'accès aux serveurs web. En effet, dans le contexte de la navigation sur internet, l'utilisateur s'attend à ce que son navigateur lui affiche instantanément sa requête. S'il patiente en général plus de quelques secondes, il va tenter d'effectuer à nouveau sa requête, ce qui fait que le serveur doit soit traiter immédiatement les demandes soit les rejeter complètement. Cela correspond bien à une politique de type LIFO, comme le montrent Dalal et Jordan (2001).

Les références citées ci-dessus ainsi que celles que l'on peut trouver dans la littérature sont principalement classées en deux catégories ; l'évaluation de performance ou le contrôle optimal. Dans la première catégorie, on considère un système dont la politique de service est donnée et on cherche à exprimer divers indicateurs de performance pour ces systèmes. Par exemple, toujours avec les centres d'appels comme application, Zeltyn et Mandelbaum (2005) considèrent un système pour lequel la dynamique ainsi que la politique de service est donnée. Les auteurs étudient différentes configurations de ce système et dérivent des mesures de performance pour chacune d'entre elles. Dans cette partie de la littérature, on trouve aussi des articles considérant plusieurs classes de clients (Zeltyn et al. 2009). Là encore, les politiques de services sont données, comme par exemple les règles de priorités entre les différentes classes de clients. Les auteurs se concentrent sur des méthodes pour calculer des indicateurs de performance, en l'occurrence le temps d'attente et le temps total passé dans le système. Dans la partie de la littérature qui se concentre sur le contrôle optimal de ce type de systèmes, on trouve des articles qui ne font pas de distinctions entre les différents clients, comme Zhao et al. (1993) que nous avons présentés précédemment ou plus récemment Ward et Kumar (2008) ou Benjaafar et al. (2010). Cependant, l'hypothèse selon laquelle les clients sont tous identiques reste restrictive. En effet, dans de nombreux systèmes les clients sont différenciés, que ce soit au niveau de la demande qu'ils émettent (service après vente, service commercial...) ou encore grâce à un contrat qui leur permet un accès facilité à une certaine assistance. Par conséquent, ne considérer qu'un seul type de clients limite le champ d'application des résultats proposés. C'est pourquoi, comme Lillo (2001), Argon et al. (2008), nous proposons dans ce manuscrit des règles de contrôle

optimales pour des systèmes faisant face à l'arrivée de différents types de clients.

1.1.3 L'aléatoire dans ces systèmes

Dans nos travaux, nous étudions l'impatience et ses conséquences sur les systèmes de files d'attente ou de production. Il vient alors la question de comment modéliser l'impatience. L'impatience d'un client dans une file d'attente peut venir de la taille de la file qu'il voit jusqu'au guichet et du temps qu'il estime devoir passer dans la file. Bien entendu tout ceci est de l'ordre de la perception et est fondé sur des estimations personnelles. Par conséquent, l'impatience sera modélisée comme une variable aléatoire.

Plus généralement, nous considérerons dans de nombreux cas que toutes les durées qui entrent en jeux dans le système vont être des variables aléatoires. Nous avons déjà expliqué le cas de l'impatience, mais l'on peut faire de même avec les durées de service, par exemple. Dans une chaîne d'assemblage, si l'assemblage est effectué par une machine, cette durée peut varier du fait de l'usure des différentes parties de la machine. De même, si la tâche est effectuée par un opérateur, il va avoir besoin d'un certain temps pour l'effectuer, les différents temps peuvent certes être très proches les uns des autres, mais vont varier selon la fatigue de l'opérateur ou encore son aisance. Dans ces cas particuliers, considérer des durées déterministes est une bonne approximation du problème. Par conséquent toutes les approches de résolution déterministes sont parfaitement justifiées. De plus, lorsque des dates d'échéances sont considérées, elles modélisent des dates qui ont été fixées de manière contractuelle et sont donc exclusivement déterministes.

Dans ce document, nous supposons dans la plupart du temps que les durées ont une variabilité justifiant l'utilisation de variables aléatoires. Dans le reste du manuscrit, il sera toujours précisé explicitement lorsqu'une durée sera aléatoire ou déterministe. Nous nous efforcerons aussi de faire le parallèle avec les problèmes déterministes correspondants. Pour que ces comparaisons soient plus naturelles, nous adopterons les notations suivantes que nous respecterons tout au long du document :

- X : une variable aléatoire, les variables aléatoires seront notées en majuscules ;
- $\mathbb{P}$: la fonction de probabilité, on notera $\mathbb{P}(X < x)$ la probabilité que la variable aléatoire X soit plus petite que x ;
- $E(X)$: l'espérance de la variable aléatoire X ;
- f_X : la fonction de densité de la variable aléatoire X, les fonctions de densité seront écrites en minuscules ;
- F_X : la fonction de répartition de la variable aléatoire X, les fonctions de répartition seront notées en majuscules ;
- $X \sim F_X$: la variable aléatoire X suit la fonction de répartition F_X, on adoptera la notation spécifique $X \sim \exp(\lambda)$ pour une variable aléatoire exponentielle de taux λ.

Lorsque nous considérerons une variable déterministe, nous la noterons en minuscules. Une date d'échéance déterministe sera par exemple notée d_j. Lorsque nous supposons qu'une variable est une variable aléatoire, nous la noterons avec la même lettre en majus-

cule, par conséquent une date d'échéance étant une variable aléatoire sera notée D_j. Une seule exception sera faite pour les durées d'exécution p_j dont les variables aléatoires seront notées X_j.

1.2 Modélisation

1.2.1 Définitions et notations

Tout au long de ce manuscrit, nous utiliserons les notations ainsi que le vocabulaire usuel de l'ordonnancement dont les termes suivants :

ordonnancement : Un ordonnancement est une affectation de tâches à des ressources dans le temps. Lorsqu'une seule machine est à disposition, on note $\{1, 2, \ldots, n\}$ l'ordonnancement, dans cet ordre, des tâches 1 à n sur cette machine.

tâche : Une tâche représente une action à effectuer par le système (client...), elles seront numérotées de 1 à n et sont décrites par les paramètres suivants :

durée d'exécution : La durée d'exécution est le temps nécessaire à l'exécution de la tâche j. On la note p_j lorsque cette durée est déterministe et X_j lorsqu'elle est définie par une variable aléatoire. Si cette variable aléatoire est exponentielle son taux sera noté μ_j.

date de disponibilité : La date de disponibilité est la date à laquelle la tâche arrive dans le système. Elle est notée r_j lorsqu'elle est déterministe et R_j lorsqu'elle suit une variable aléatoire. Il n'est pas possible de commencer l'exécution de la tâche j avant r_j.

date d'échéance : La date d'échéance est la date à laquelle l'exécution d'une tâche doit être terminée. En revanche, rien n'empêche la tâche d'être exécutée après sa date d'échéance. La date d'échéance sera notée d_j lorsqu'elle est déterministe et D_j lorsqu'elle suit une variable aléatoire. Si cette variable est exponentielle son taux sera noté γ_j.

poids : Le poids d'une tâche, noté w_j, représente son importance par rapport aux autres tâches à exécuter. Lorsque le système que l'on considère comporte des coûts d'abandons, nous les noterons effectivement w_j. Cependant, pour certains modèles, nous aurons besoin d'introduire des coûts de mise en attente en plus des coûts d'abandon. Pour de tels modèles, nous noterons les coûts de mise en attente c_j, afin de les différencier des coûts d'abandon.

La position d'une tâche dans un ordonnancement peut être décrite par des variables additionnelles. On notera respectivement S_j et C_j les dates de début et de fin d'exécution d'une tâche j. Dans le cas déterministe, si la tâche j n'est pas interrompue, on a $C_j = S_j + p_j$. On note aussi U_j la fonction indicatrice qui est égale à 1 si la tâche j est en retard et 0 sinon. Nous verrons dans la Section 1.4 que la définition

de U_j dépend du type d'impatience considéré. Nous ne faisons pas de différences en notant ces variables lorsqu'elles sont déterministes ou stochastiques.

machine : Une machine représente l'unique ressource disponible ici (opérateur, serveur...), une machine est indispensable pour exécuter une tâche, mais une machine n'exécute qu'une seule tâche à la fois, elles seront numérotées de 1 à m.

préemption : Si on autorise la préemption, on autorise l'interruption d'une tâche, on peut la reprendre plus tard en ayant gardé en mémoire le temps déjà passé en exécution. Pendant cet intervalle de temps, on est libre d'exécuter n'importe quelle autre tâche présente dans le système.

Nous utiliserons la notation de Graham et al. (1979) pour décrire le problème d'ordonnancement que nous étudions. Cette notation comporte trois champs de la forme

$$\alpha \mid \beta \mid \gamma$$

avec α décrivant les machines ou les ressources disponibles, β les caractéristiques des tâches que les ressources doivent traiter et γ la fonction objectif. La composition de ces champs est potentiellement très variée, dans ce document, nous utiliserons les notations suivantes :

Champ α : machines

- 1 : une seule machine est disponible ;
- Pm : m machines parallèles et identiques sont disponibles ;
- Qm : m machines parallèles de vitesses différentes sont disponibles.

Champ β : contraintes

- $pmtn$: s'il apparaît dans le champ, la préemption est autorisée, sinon on considère que l'ordonnancement est sans préemption ;
- r_j ou R_j : s'il apparaît dans le champ, les tâches ont des dates de disponibilité non nulles, sinon on considère par défaut que les tâches sont toutes présentes à l'instant 0 ;
- p_j ou X_j : précise les contraintes sur les durées d'exécution. Par exemple, si elles sont toutes égales, on notera $p_j = p$, si elles suivent des variables aléatoires exponentielles, on notera $X_j \sim \exp(\mu_j)$. Si rien n'est précisé, on suppose que les durées d'exécution sont déterministes et quelconques ;
- d_j ou D_j : précise les contraintes sur les dates d'échéance, si elles suivent des variables aléatoires exponentielles i.i.d. (indépendantes et identiquement distribuées), on notera $D_j \sim \exp(\gamma)$, sans rappeler l'indice j dans la description de la loi de probabilité. Si aucune valeur n'est précisée, la présence ou non de dates d'échéances est impliquée par la fonction objectif. Si c'est le cas, on suppose que les dates d'échéance sont déterministes et quelconques ;
- w_j : précise les contraintes sur le poids des tâches. Si le poids n'est pas présent dans la fonction objectif, on considère que toutes les tâches ont le même poids égal à 1.

Champ γ : fonction objectif

- $\sum w_j C_j$ ou $E(\sum w_j C_j)$: la minimisation de la somme pondérée des dates de fin (ou son espérance), cette fonction objectif peut modéliser un coût de stockage ou de mise en attente pour les différentes tâches ;
- $\sum w_j U_j$ ou $E(\sum w_j U_j)$: cette fonction objectif correspond à la minimisation du nombre pondéré de tâches en retard (ou son espérance) ;

Par exemple, $1 \mid\mid \sum U_j$ décrit un problème d'ordonnancement déterministe à une machine, avec pour fonction objectif la minimisation du nombre de tâches en retard. Ici, les poids w_j sont égaux à 1, les tâches sont toutes disponibles à l'instant 0, les durées d'exécution et les dates d'échéance sont définies implicitement, elles sont donc quelconques. $1 \mid R_j \sim F_{R_j}, X_j \sim \exp(\mu_j), D_j \sim F_D \mid E(\sum w_j U_j)$ décrit un problème d'ordonnancement stochastique à une machine, avec pour fonction objectif la minimisation de l'espérance du nombre pondéré de tâches en retard. Les tâches sont disponibles suivant une variable aléatoire quelconque, les durées d'exécution suivent des variables aléatoires exponentielles et les dates d'échéance suivent des variables aléatoires quelconques mais i.i.d.

Ces problèmes d'ordonnancement sont très proches des problèmes de contrôle de files d'attente. La principale différence entre les deux systèmes est qu'en ordonnancement, on a n tâches à exécuter et en théorie des files d'attente, on considère que l'on a n classes de tâches à exécuter, dont les tâches deviennent disponibles suivant un processus d'inter-arrivées sur un horizon infini. En théorie des files d'attente, on utilise la notation de Kendall (1953) pour décrire le problème considéré. Cette notation est de la forme

$$A/S/c/K/N/D$$

avec

- A : le processus d'arrivée, si les arrivées suivent un processus de Poisson, on notera M (c.f. annexe A.1.5 pour des détails) et $M(n)$ si le taux de la loi de Poisson dépend du nombre de tâches présentes dans le système ; si les arrivées sont périodiques, on notera D cette période ;
- S : le processus de service, si les durées d'exécution suivent des variables aléatoires quelconques, on notera G ;
- c : le nombre de machines ;
- K : la capacité du système, lorsqu'elle n'est pas précisée, le système est de capacité infinie ;
- N : le nombre de tâches à exécuter, lorsqu'il n'est pas précisé, on suppose ce nombre tend vers l'infini ;
- D : la politique de service, comme par exemple FIFO.

On adapte cette notation en ajoutant « $+I$ » au champ c. I, si précisé, décrit les dates d'échéances des tâches. Si on considère un problème à une machine où les dates d'échéances sont des variables aléatoires exponentielles, on notera $1 + M$.

tâche	X_j		r_j
1	2	avec probabilité 1/2	0
	10	avec probabilité 1/2	
2	2	avec probabilité 1/2	1
	5	avec probabilité 1/2	

TABLE 1 – Exemple d'instance d'ordonnancement stochastique

Par exemple, le problème d'ordonnancement $1 \mid X_j \sim \exp(\mu_j) \mid E(\sum w_j C_j)$ sera noté $D/M/1/n$. En prenant la même période et le même instant initial pour toutes les n tâches, on obtient bien un problème où toutes les tâches sont toutes présentes à l'instant 0. De la même manière, le problème $1 \mid R_j \sim F_{R_j}, X_j \sim \exp(\mu_j), D_j \sim \exp(\gamma_j) \mid E(\sum w_j U_j)$ sera noté $G/M/1 + M/n$.

1.2.2 Modélisation des problèmes étudiés

Dans le chapitre 3 nous étudions entre autres le problème $1 \mid X_j \sim F_{X_j}, D_j \sim F_{D_j} \mid E(\sum w_j U_j)$ que l'on peut noter $D/G/1 + G/n$ en utilisant la notation de Kendall. Dans les problèmes que l'on considère, on fait certaines hypothèses sur les lois de probabilités que suivent X_j et D_j, nous permettant de donner des ordonnancements optimaux pour ces problèmes.

Dans le chapitre 4, nous étudions plus particulièrement le problème $1 \mid R_j \sim F_{R_j}, X_j \sim \exp(\mu_j), D_j \sim \exp(\gamma_j) \mid E(\sum w_j U_j)$ que l'on peut noter $G/M/1+M$ en utilisant la notation de Kendall (1953).

Nous venons de préciser quelles sont les notations qui vont être utilisées tout au long du document pour décrire un problème d'ordonnancement. Ces notations sont cependant insuffisantes pour décrire précisément le problème considéré. Nous préciserons dans quelle classe de politiques nous cherchons la politique optimale, ce qui est le thème de la section 1.3, ainsi que le type d'impatience considéré, ce que nous verrons dans la section 1.4.

1.3 Classes de politiques

La description d'un problème d'ordonnancement stochastique demande à être plus précise que celle d'un ordonnancement déterministe. En effet, dans un contexte stochastique, de nouvelles informations sont continuellement disponibles, comme la réalisation d'une variable aléatoire. Selon la classe de politique choisie, on a le droit ou non d'utiliser ces informations. C'est ce que nous précisons dans les prochaines sous-parties.

Exemple. Nous illustrerons les différents cas par l'ordonnancement stochastique à deux tâches suivant : $1 \mid r_j, X_j \mid E(\sum C_j)$, les durées d'exécution et les dates de disponibilité des deux tâches sont données dans le tableau 1.

1.3.1 Politiques statiques

Définition 1 (Pinedo 2008)**.** Dans la classe des politiques statiques, on décide de la liste de priorité suivant laquelle les tâches vont être exécutées à l'instant $t = 0$. Cette liste ne sera pas changée au cours de l'exécution et on ne prend en compte aucune information après l'instant initial.

Par conséquent, dans la classe des politiques statiques, si toutes les tâches sont présentes à l'instant initial, autoriser la préemption est inutile. En effet, comme toutes les tâches sont disponibles à l'instant initial et que l'ordre d'exécution ne change pas au cours du temps, une tâche n'est jamais préemptée. En revanche, si certaines tâches deviennent disponibles au cours de l'exécution, il est alors légitime de considérer la préemption. Dans ce cas, on construit toujours une liste d'exécution à l'instant initial qui ne sera pas modifiée, mais dans ce cas, on exécute à chaque instant la tâche disponible qui se trouve le plus en tête de liste.

Exemple (suite)**.** Prenons l'instance donnée dans le tableau 1. Il y a 4 réalisations équiprobables des durées d'exécutions des tâches 1 et 2, elles seront notées de S_1 à S_4. Dans la classe des politiques statiques, il y a 2 ordonnancements possibles, correspondant aux deux listes $\{1, 2\}$ et $\{2, 1\}$. Dans la classe des politiques statiques sans préemption, ces deux listes conduisent au même ordonnancement réalisé. En effet, dans les deux cas, on commence à exécuter la tâche présente qui se trouve en tête de liste, à savoir la tâche 1. Ce n'est qu'une fois que cette tâche est finie que l'on peut passer à la tâche 2. Dans ce cas, les quatre réalisations possibles de l'ordonnancement optimal sont données dans la figure 2 et l'objectif a pour valeur $15,5 = \frac{1}{4}(2+4) + \frac{1}{4}(10+12) + \frac{1}{4}(2+7) + \frac{1}{4}(10+15)$.

En revanche, si l'on autorise la préemption, dans l'ordonnancement $\{2, 1\}$ on exécutera la tâche 1 jusqu'à l'arrivée de la tâche 2 comme le montre la figure 3. Cet ordonnancement, de valeur 14, est au moins aussi bon que l'ordonnancement statique sans préemption optimal car on a une liberté supplémentaire.

En particulier, dans la classe des politiques statiques, il n'est pas autorisé d'utiliser des informations après $t = 0$, comme par exemple la réalisation d'une variable aléatoire. On ne peut donc pas réagir face à la fin d'une exécution ou à la réalisation d'une date d'échéance aléatoire. Cela implique que dans un problème d'ordonnancement avec impatience, on peut très bien être amené à exécuter des tâches qui sont en retard.

Prenons le problème déterministe d'ordonnancement avec dates d'échéances $1 \mid\mid \sum U_j$, l'ordonnancent optimal proposé par Moore (1968) sépare les tâches en deux ensembles. Le premier ensemble contient toutes les tâches qui terminent à temps, elles sont exécutées en premier. Le deuxième ensemble contient les tâches qui terminent en retard, elles sont exécutées à la fin de l'ordonnancement (nous donnons cet algorithme page 32). Si elle était adaptée à un contexte stochastique, cette politique n'entrerait pas dans la classe des politiques statiques. En effet, si les dates d'échéance étaient des variables aléatoires, nous

aurions besoin de savoir quand elles se réalisent afin de pourvoir déterminer si une tâche est en retard ou si elle termine à temps. Le fait de pouvoir éjecter une tâche à la fin de l'ordonnancement si elle dépasse sa date d'échéance, est donc une caractéristique des politiques dynamiques.

1.3.2 Politiques dynamiques

Définition 2 (Pinedo 2008)**.** Dans la classe des politiques dynamiques, on peut décider à n'importe quel instant quelle sera la **prochaine** tâche à être exécutée, en prenant en compte toutes les informations disponibles.

Dans la classe des politiques dynamiques, on peut aussi choisir d'attendre, alors que des tâches sont déjà disponibles. Dans ce cas, on parle de temps mort sur la machine. Même si dans la plupart des cas traités dans ce document, les ordonnancements avec temps morts sont sous-optimaux, nous ne nous restreindrons pas aux ordonnancements sans temps mort. Dans la classe des politiques dynamiques, lorsque la préemption est autorisée, on peut décider à chaque instant de quelle est la tâche qui est exécutée, ou d'attendre. Cependant, lorsque la préemption n'est pas autorisée, on doit attendre la fin de l'exécution d'une tâche avant d'en commencer une nouvelle (ou d'attendre).

Exemple (suite)**.** Le cas d'un ordonnancement avec temps mort est montré dans la figure 4. Pour notre exemple, il est plus avantageux d'attendre que la tâche 2 devienne disponible, plutôt que de commencer l'exécution de la tâche 1 alors qu'elle est déjà présente dans le système. La valeur de cet ordonnancement est alors de 15.

Enfin, dans la classe des politiques dynamiques avec préemption, la priorité accordée à chacune des tâches peut varier au cours du temps et l'exécution peut passer d'une tâche à l'autre sans contraintes. Dans l'exemple qu'illustre la figure 6, on commence par exécuter la tâche 1, car elle est la seule présente dans le système. Cependant, lorsque la tâche 2 devient disponible, on continue d'exécuter la tâche 1 pendant une unité de temps pour tester sa durée d'exécution. En effet, la tâche 1 peut soit être courte (d'une durée 2) et dans ce cas à l'instant 2 la tâche 1 est terminée, soit être plus longue et dans ce cas on bascule sur l'exécution de la tâche 2. On revient sur la tâche 1 après la fin de l'exécution de la tâche 2, si nécessaire. Cet ordonnancement conduit à la valeur optimale pour cette instance, à savoir 13,25. Dans la figure 5, on considère le même type d'ordonnancement, mais dans ce cas, on teste la durée d'exécution de la tâche 2 en premier. Cet ordonnancement a pour valeur 15.

Cet exemple illustre bien les différentes inclusions des ensembles de politiques de ces différentes classes. Notons respectivement $\{\pi_{stat}\}$, $\{\pi_{stat,pmtn}\}$, l'ensemble des politiques statiques sans préemption et l'ensemble des politiques statiques avec préemption, ainsi que $\{\pi_{dyn}\}$, $\{\pi_{dyn,pmtn}\}$, l'ensemble des politiques dynamiques sans préemption et l'ensemble des politiques dynamiques avec préemption. Ces différents ensembles respectent les inclusions données dans la figure 7.

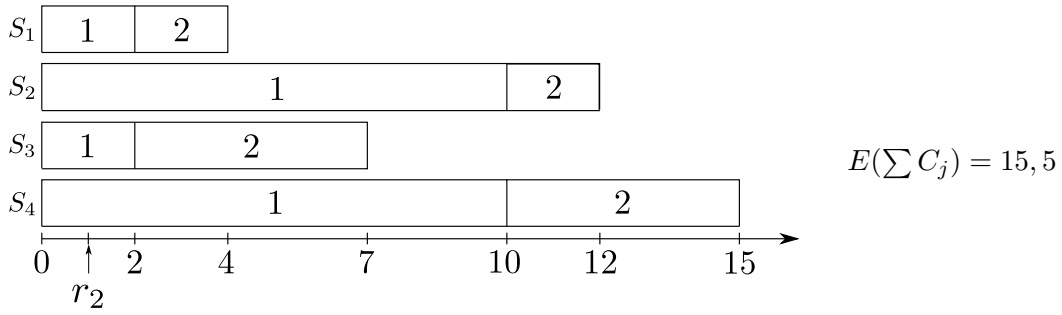


FIGURE 2 – Ordre de priorité $\{1, 2\}$ dans la classe des politiques statiques ou dynamiques, avec ou sans préemption, ou ordre de priorité $\{2, 1\}$ dans la classe des politiques statiques ou dynamiques sans préemption

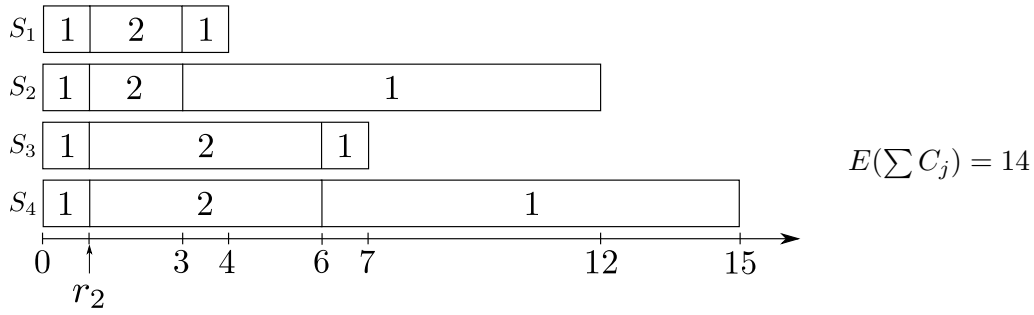


FIGURE 3 – Ordre de priorité $\{2, 1\}$ dans la classe des politiques statiques ou dynamiques avec préemption

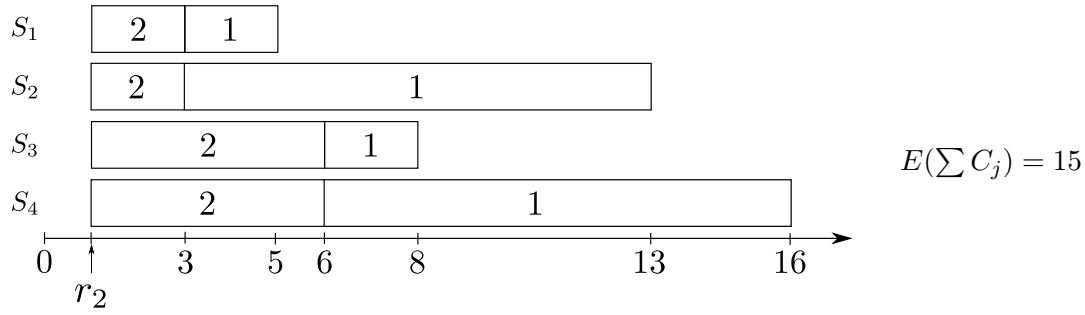


FIGURE 4 – Ordre d'exécution $\{2, 1\}$ dans la classe des politiques dynamiques sans préemption

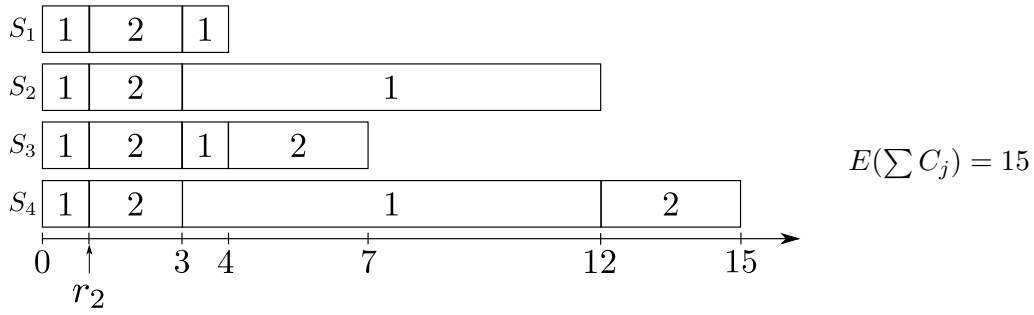


FIGURE 5 – Ordre de priorité à 2 pendant une durée de 2, puis priorité à 1, dans la classe des politiques dynamiques avec préemption

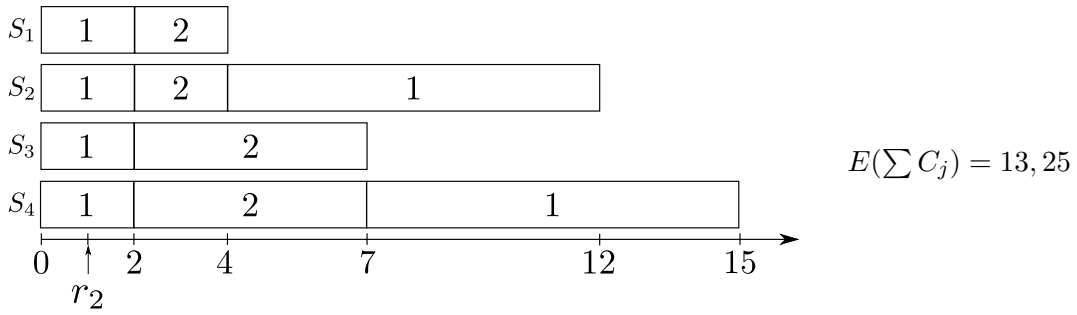


FIGURE 6 – Ordre de priorité à 1 pendant une durée de 2, puis priorité à 2, dans la classe des politiques dynamiques avec préemption

$$\begin{array}{ccc} \{\pi_{stat}\} & \subseteq & \{\pi_{dyn}\} \\ \cap| & & \cap| \\ \{\pi_{stat,pmtn}\} & \subseteq & \{\pi_{dyn,pmtn}\} \end{array}$$

FIGURE 7 – Inclusions des ensembles de politiques

Par la suite, nous étudions un problème d'ordonnancement stochastique avec abandon dans la classe des politiques statiques sans préemption (chapitre 3), ainsi que dans la classe des politiques dynamiques avec préemption (chapitre 4).

Dans le cas où le système est sans mémoire, son état se réduit au nombre de tâches de chaque classe présentes dans le système. On notera alors $x = (x_1, x_2, \ldots, x_n)$ l'état du système, avec x_i le nombre de tâches de la classe i présentes dans le système, pour tout $i \in \{1, 2, \ldots, n\}$. Par conséquent, les politiques d'ordonnancement ne dépendent elles aussi que du nombre de tâches de chaque classe dans le système. Nous distinguons alors deux types de politiques, les politiques strictes et les politiques à seuil.

Politiques strictes

Soient n classes de tâches indexées de 1 à n et considérons une politique stricte donnant la priorité aux tâches appartenant à la classe de plus petit indice. Une telle politique n'exécute une tâche d'une classe i que si aucune tâche appartenant à une classe d'indice inférieur n'est disponible. Une représentation graphique d'une politique stricte pour deux classes de tâches est donnée dans la figure 8.

Politiques à seuil

Nous ne considérerons des politiques à seuil que lorsque le système comporte deux classes de tâches. Dans ce cas, l'état du système est noté $x = (x_1, x_2)$. On dit qu'une politique est à seuil s'il existe une fonction t de $\mathbb{N}$ dans $\mathbb{N}$, telle que si $x_1 \leq t(x_2)$ alors la politique donne la priorité aux tâches appartenant à une certaine classe (e.g. la classe 2) et si $x_1 > t(x_2)$ alors la politique donne la priorité aux tâches de l'autre classe (e.g. la classe 1). Une représentation graphique d'une telle politique est donnée dans la figure 9.

1.4 Types d'impatience

Dans nos travaux, nous prenons soin de préciser le type d'impatience que nous considérons (nous en distinguons deux types) et de faire la différence entre l'*impatience* et l'*abandon*.

1.4.1 Impatience jusqu'à la fin du service

L'impatience jusqu'à la fin du service, sera abrégée par la suite en *IES* pour *impatience to the end of service*.

Définition 3. On dit d'un problème qu'il traite de l'impatience jusqu'à la fin du service, si une tâche est considérée en retard lorsque son exécution **termine** après sa date d'échéance.

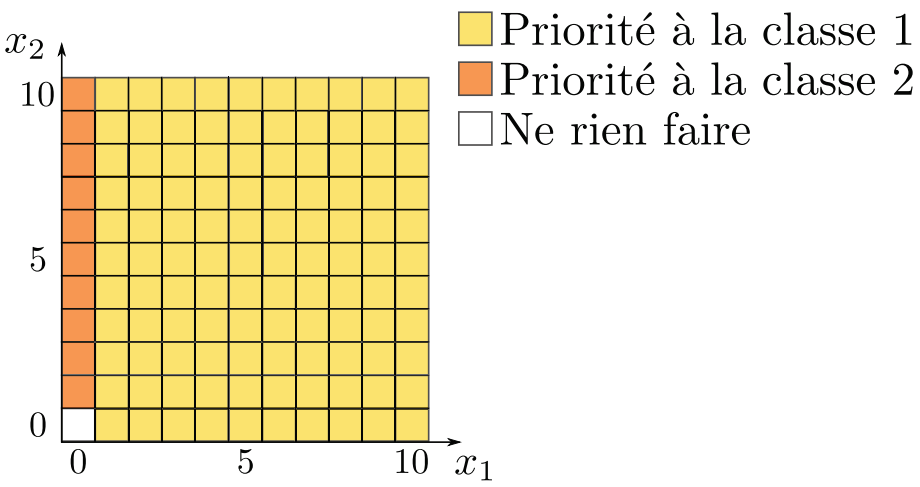


FIGURE 8 – Représentation d'une politique stricte

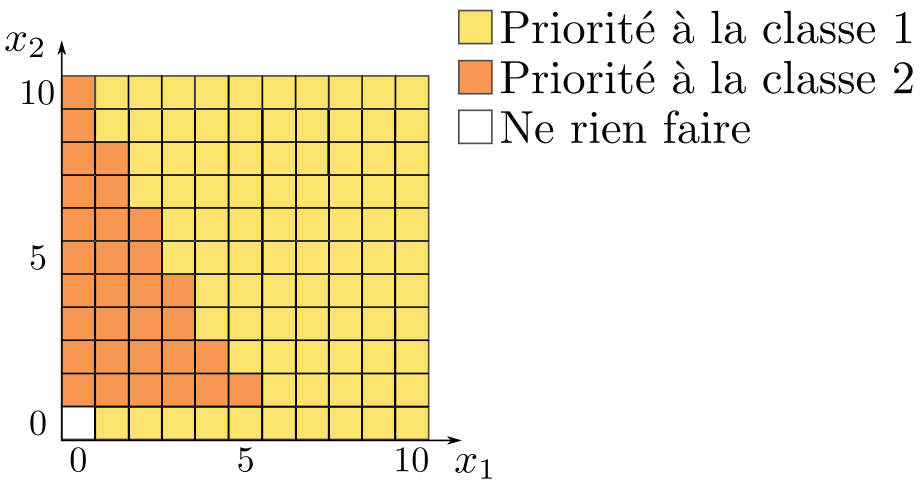


FIGURE 9 – Représentation d'une politique à seuil

Avec les notations définies plus tôt, nous avons

$$U_j = \begin{cases} 1 & \text{si } C_j > d_j \\ 0 & \text{sinon.} \end{cases}$$

Cette définition de l'impatience est celle qui est traditionnellement utilisée en ordonnancement avec dates d'échéances. Cette définition est utilisée pour modéliser des systèmes de production de biens, dans lesquels le client demande un produit pour une certaine date. Il faut alors que la production soit terminée avant la date stipulée dans le contrat.

1.4.2 Impatience jusqu'au début du service

L'impatience jusqu'au début du service sera abrégée par la suite en *IBS* pour *impatience to the beginning of service.*

Définition 4. On dit d'un problème qu'il traite de l'impatience jusqu'au début du service, si une tâche est considérée en retard lorsque son exécution **commence** après sa date d'échéance. Avec les notations définies plus tôt, nous avons

$$\tilde{U}_j = \begin{cases} 1 & \text{si } S_j > d_j \\ 0 & \text{sinon.} \end{cases}$$

Cette définition de l'impatience est traditionnellement utilisée dans les systèmes de files d'attente où elle traduit effectivement le comportement qu'aurait un client qui patiente pour obtenir un service. Si le service de ce client n'a pas commencé avant un certain temps, il va s'impatienter et éventuellement quitter la file d'attente. En revanche, il est raisonnable de supposer qu'une fois que le client est en train d'être servi, il ne s'impatientera plus, même si la durée nécessaire à son service est importante.

Précisons le comportement de ce type d'impatience face à la préemption. S'il est clair que la préemption n'a pas d'incidence sur la définition d'IES, il n'en est pas de même pour l'IBS. En effet, considérons un problème d'ordonnancement avec IBS dans lequel on autorise la préemption. Soit un ordonnancement où l'on exécute chacune des tâches pendant une durée ϵ assez faible, de telle sorte que toutes les tâches aient commencées avant leur date d'échéance. On termine ensuite les tâches dans un ordre quelconque. Dans ce cas, toutes les tâches ont commencé avant leur date d'échéance, on pourrait donc penser que cet ordonnancement est optimal. Dans notre cas, une tâche qui est en train d'être préemptée et qui dépasse sa date d'échéance, est considérée comme étant en retard.

1.4.3 Équivalence dans le cas déterministe

Considérons les deux problème 1) $1 \mid r_j \mid \sum w_j U_j$ (avec IES) et 2) $1 \mid r_j \mid \sum w_j \tilde{U}_j$ (avec IBS). Soit une instance quelconque pour le problème 1 et notons d_j^{IES} les dates d'échéances

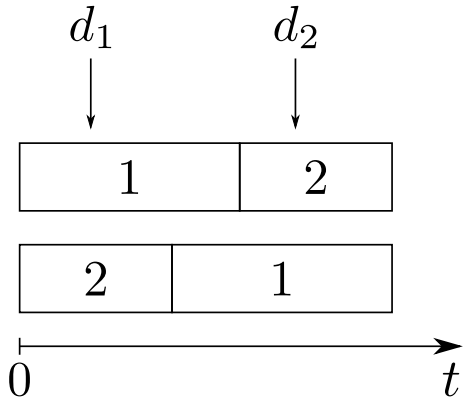


FIGURE 10 – Exemple d'instance dont l'ordonnancement optimal avec IES diffère de l'ordonnancement optimal avec IBS

pour cette instance. À partir de cette instance, nous construisons une instance pour le problème 2 dont tous les paramètres de toutes les tâches sont identiques, à l'exception des dates d'échéance $d_j^{IBS} = d_j^{IES} - p_j$. Dans ce cas, si l'on dispose d'un algorithme résolvant le problème 2 à l'optimal, on peut résoudre l'instance du problème 1 à l'optimal en appliquant cet algorithme sur l'instance du problème 2 que l'on vient de construire. La réciproque peut être démontrée de la même manière. Ces deux problèmes sont donc équivalents au sens de la complexité.

Notons que cela n'implique pas que les ordonnancements optimaux soient les mêmes pour les deux types d'impatience, comme le montre l'exemple donné dans la figure 10. Dans l'exemple donné, l'ordonnancement optimal avec IES est $\{2, 1\}$ et l'objectif vaut w_1 (l'objectif pour l'ordonnancement $\{1, 2\}$ vaut $w_1 + w_2$), alors que l'ordonnancement optimal avec IBS est $\{1, 2\}$ et l'objectif vaut 0 (l'objectif pour l'ordonnancement $\{2, 1\}$ vaut w_1).

En général, cette équivalence n'est plus vraie dans un contexte stochastique, d'une part car il n'y a pas de notion de complexité dans le cas stochastique et d'autre part car les résultats donnés ne sont valables que pour certaines lois de probabilité. Supposons que l'on connaît un ordonnancement optimal pour un problème avec IBS lorsque les dates d'échéance sont exponentielles. Pour pouvoir résoudre le problème avec IES à l'aide de cet ordonnancement, il faut que les dates d'échéances $D'_j \to D_j - X_j$ suivent aussi des lois exponentielles. Mais par exemple, la somme de deux variables aléatoires exponentielles (ou la différence) n'est pas une variable aléatoire exponentielle. Par conséquent, l'équivalence entre IES et IBS n'est que rarement valable dans un contexte stochastique.

1.4.4 Impatience et abandon

Dans cette thèse nous faisons aussi la distinction entre impatience et abandon. Une tâche devient impatiente, si elle a passé plus d'un certain temps dans le système. Dans ce cas, un coût peut être imputé au système. Suite à une impatience, la tâche peut abandonner,

	Impatience	Abandon
Statique	chapitre 3	explosion combinatoire
Dynamique	impatience $\Leftrightarrow$ abandon	chapitre 4

TABLE 2 – Impatience et classes de politiques

elle quitte alors le système et ne peut plus être exécutée. Il peut donc y avoir deux situations possibles :

impatience sans abandon : C'est le comportement que nous étudions dans le chapitre 3, les tâches s'impatientent, un coût est imputé au système, mais elles seront quand même exécutées.

impatience avec abandon : C'est le comportement que nous étudions dans le chapitre 4, les tâches s'impatientent et quittent le système simultanément, un coût est imputé au système mais les tâches ne seront pas exécutées.

Par la suite, lorsque l'on parlera d'un problème avec impatience, on sous-entendra un problème avec impatience mais sans abandon. De même, lorsque l'on parlera d'un problème avec abandon, on sous-entendra qu'il est dû à l'impatience de la tâche en question.

L'impatience et l'abandon des tâches dépendent aussi de la classe de politiques dans laquelle on cherche une politique optimale. Dans les chapitres suivants, nous étudierons les ordonnancements avec impatience dans la classe des politiques statiques, ainsi que les ordonnancements avec abandon dans la classe des politiques dynamiques. Dans la classe des politiques dynamiques, il n'est pas nécessaire de faire la distinction entre impatience ou abandon. En effet, si l'on considère un problème avec impatience, les tâches qui ont dépassées leurs dates d'échéances vont être exécutées à la fin de l'ordonnancement, après toutes les tâches qui terminent à temps. Ce comportement est donc comparable au phénomène d'abandon, à la différence près que les tâches sont quand même exécutées. Dans la classe des politiques statiques avec abandons, il faut considérer toutes les situations possibles lorsque l'on évalue la liste d'exécution des tâches pour trouver l'optimum. A savoir, pour chaque tâche, est-ce que celle-ci va abandonner ou non. Cela entraîne une explosion combinatoire du problème que nous n'étudions pas dans cette thèse.

1.5 Organisation du document

La suite de ce document est organisée comme suit.

Dans le chapitre 2, nous effectuons une revue de littérature sur l'impatience en ordonnancement, ainsi qu'en théorie des files d'attente.

Dans le chapitre 3 nous étudions les problèmes
– $1 \mid X_j \sim F_{X_j}, D_j \sim F_{D_j} \mid E(\sum w_j U_j)$ et

– $1 \mid X_j \sim F_{X_j}, D_j \sim F_{D_j} \mid E(\sum w_j \tilde{U}_j)$

dans la classe des politiques statiques sans préemption. Les tâches sont disponibles à l'instant initial, les durées d'exécution et les dates d'échéance suivent différentes distributions. La fonction objectif est de minimiser l'espérance du nombre pondéré de tâches en retard avec IES ou avec IBS.

Dans le chapitre 4, nous étudions plus particulièrement les problèmes

– $1 \mid R_j \sim F_{R_j}, X_j \sim \exp(\mu_j), D_j \sim \exp(\gamma_j) \mid E(\sum w_j U_j)$ et

– $1 \mid R_j \sim F_{R_j}, X_j \sim \exp(\mu_j), D_j \sim \exp(\gamma_j) \mid E(\sum w_j \tilde{U}_j)$

dans la classe des politiques dynamiques avec préemption. Les tâches deviennent disponibles de manière aléatoire, les durées d'exécution et les dates d'échéance suivent des variables aléatoires exponentielles.

Dans le chapitre 5, nous présentons des résultats ainsi que des perspectives de recherche concernant des problèmes à plusieurs machines. Nous cherchons des ordonnancements optimaux appartenant à la classe des politiques statiques ou à la classe des politiques dynamiques sans préemption.

Enfin, nous concluons cette thèse avec le chapitre 6 en rappelant nos principales contributions ainsi que des perspectives de recherche.

Chapitre 2

Revue de littérature

Dans ce chapitre, nous présentons les principaux résultats de deux problèmes d'ordonnancement dans les cas déterministes et stochastiques. Nous choisissons de nous concentrer sur ces problèmes car cette thèse s'intéresse à des variantes ou des extensions de ces problèmes. De plus, les techniques de preuve que nous utilisons sont adaptées de ces résultats.

Dans le premier problème, il n'y a pas d'abandon, mais un coût à payer par unité de temps passé par chaque tâche dans le système. Dans ce problème, on minimise par conséquent le temps d'attente total des tâches. Cela se traduit par la fonction objectif $\sum w_j C_j$. Le deuxième problème est, quant à lui, sujet à des abandons et un coût est payé à chaque fois qu'une tâche quitte le système suite à un abandon. Dans ce problème, on minimise donc les coûts d'abandon, ce qui se traduit par la fonction objectif $\sum w_j U_j$, ou $\sum w_j \tilde{U}_j$.

Nous donnerons aussi le détail de certaines preuves classiques issues de la littérature. En particulier, nous détaillerons des preuves utilisant des arguments d'échange et des preuves par induction, car ces techniques seront utilisées dans les chapitre 3 et 4. Nous verrons aussi une argumentation permettant de montrer qu'un ordonnancement optimal dans la classe des politiques statiques, l'est aussi dans la classe des politiques dynamiques. Cette technique sera plus particulièrement utilisée dans le chapitre 4. Enfin, nous présenterons un exemple de preuve utilisant des processus de décision de Markov (MDP), permettant d'exhiber la structure d'un problème stochastique. Les MDP prennent en effet une part importante dans les études numériques que nous effectuons dans les chapitres 4 et 5.

2.1 Minimisation des coûts de mise en attente

Dans cette section, nous présentons les problèmes d'ordonnancement ayant comme fonction objectif $\sum w_j C_j$. Dans les cas déterministes et stochastiques, nous préciserons quels problèmes sont faciles à résoudre et quels problèmes sont difficiles. Lorsque les problèmes sont faciles à résoudre, la règle d'ordonnancement WSPT (*Weighted Shortest Processing Time*), ou une variante de cette règle, est optimale. Dans un ordonnancement qui respecte

cette règle, les tâches sont ordonnées dans l'ordre décroissant des w_j/p_j. Cette règle a été proposée pour la première fois par Smith (1956) et a montré son efficacité pour résoudre bon nombre de problèmes d'ordonnancement.

2.1.1 Cas déterministe

Avant de présenter les principaux résultats de la littérature sur ce problème, nous commençons par donner la preuve classique de l'optimalité de WSPT pour le problème d'ordonnancement $1 \mid\mid \sum w_j C_j$ (telle que la donne Pinedo 2008 par exemple). Cet exemple simple nous permet de nous familiariser avec les preuves par argument d'échange, qui sont au cœur des preuves des chapitres 3 et 4.

Théorème 1. *La règle WSPT est optimale pour le problème* $1 \mid\mid \sum w_j C_j$.

Démonstration. Notons tout d'abord que tout ordonnancement avec un temps mort est sous-optimal pour ce problème. Nous ne considérerons donc pas de tels ordonnancements. Nous allons prouver ce résultat pour argument d'échange. Soit les ordonnancements $S = \{1, 2, \ldots, i, j, \ldots, n\}$ et $S' = \{1, 2, \ldots, j, i, \ldots, n\}$ qui ne diffèrent que par l'échange des deux tâches : i et j. Comme il n'y a pas de temps morts dans l'ordonnancement, $S_i = S'_j$ et $C_j = C'_i$ (cf. figure 11). On peut alors calculer $C(S)$ et $C(S')$, les coûts de ces deux ordonnancements :

$$C(S) = \sum_{\substack{k=1 \\ i \neq k \neq j}}^{n} w_k C_k + w_i(s_i + p_i) + w_j(s_i + p_i + p_j),$$

$$C(S') = \sum_{\substack{k=1 \\ i \neq k \neq j}}^{n} w_k C_k + w_j(s_i + p_j) + w_i(s_i + p_j + p_i).$$

Il vient $C(S') - C(S) = w_i p_j - w_j p_i$. Par conséquent, l'ordonnancement S est meilleur que l'ordonnancement S' si et seulement si $w_i/p_i \geq w_j/p_j$. En partant d'un ordonnancement optimal S, et en appliquant successivement de tels échanges, on obtient un ordonnancement optimal S^* respectant WSPT. □

La preuve que WSPT est optimale pour le problème $1 \mid\mid \sum w_j C_j$ a été donnée pour la première fois par Smith (1956). La complexité de cet algorithme est $O(n \log n)$, car il consiste en un tri des tâches. Nous allons maintenant présenter les principaux résultats sur les problèmes concernant la minimisation des coûts de mise en attente dans le cas déterministe.

Lorsque les tâches ont des dates de disponibilité non nulles ($r_j \neq 0$), Schrage (1968) prouve que la règle SRPT (*shortest remaining processing time*) est optimale pour résoudre le problème $1 \mid r_j, pmtn \mid \sum C_j$. Comme son nom l'indique, cette politique exécute à chaque instant la tâche qui est la plus proche de la fin de son exécution.

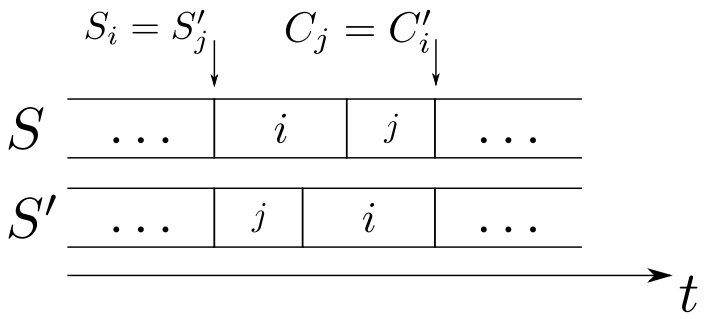


FIGURE 11 – Ordonnancements S et S'

Dès que l'on n'autorise plus la préemption, le problème devient difficile à résoudre. Comme le prouvent Lenstra et al. (1977), ce problème devient alors $\mathcal{NP}$-difficile.

On pourrait penser que dans le cadre plus général du problème $1 \mid r_j, pmtn \mid \sum w_j C_j$, la politique WSRPT (*Weighted Shortest Remaining Processing Time*) donnerait de bon résultats. Cependant, il y été prouvé par Labetoulle et al. (1984), que ce problème est $\mathcal{NP}$-difficile au sens fort, il n'y a donc aucune chance, sauf si $\mathcal{P} = \mathcal{NP}$, que pour n'importe quelle instance, un tri donné a priori conduise à une solution optimale.

Les problèmes que nous aborderons dans la suite de ce document conjuguent arrivées et poids des tâches différents. On constate que, dans le cas déterministe, des tels problèmes sont difficiles (mais deviennent faciles s'il n'y a pas d'arrivées, ou si les poids sont égaux).

2.1.2 Cas stochastique

Dans cette section, nous allons voir que les résultats sur ce problème dépendent fortement de la classe de politiques dans laquelle on cherche un ordonnancement optimal et des distributions que suivent les variables aléatoires. Dans le cas stochastique, la règle WSPT présentée plus tôt devient WSEPT (*Weighted Shortest Expected Processing Time*) où les tâches sont ordonnées suivant les $w_j/E(X_j)$ décroissants. Cette preuve, telle que la donne Pinedo (2008), se déroule en deux parties. Dans la première partie, on prouve que WSEPT est optimale dans la classe des politiques statiques sans préemption. Ce n'est pas sur cette partie de la preuve que nous allons nous concentrer, car elle reprend les mêmes arguments que celle du théorème 1. En revanche, dans la deuxième partie de la preuve, on pose des conditions garantissant que l'ordonnancement optimal dans la classe des politiques statiques sans préemption, l'est aussi dans la classe des politiques dynamiques sans préemption. Cette partie de la preuve utilise une récurrence que nous allons adapter dans le chapitre 4.

Théorème 2. *La règle WSEPT conduit à un ordonnancement optimal pour le problème* $1 \mid X_j \sim F_{X_j} \mid E(\sum w_j C_j)$ *dans les classes de politiques statiques et dynamiques sans préemption.*

Démonstration. On prouve ce résultat dans la classe des politiques statiques sans préemption en utilisant un argument d'échange, de la même manière que dans le cas déterministe. En effet, nous avons $E(S_i) = E(S'_j)$ et $E(C_j) = E(C'_i)$, par conséquent on peut exprimer l'espérance du coût des deux ordonnancements

$$E(C(S)) = \sum_{\substack{k=1 \\ i \neq k \neq j}}^{n} w_k E(C_k) + w_i(E(s_i) + E(p_i)) + w_j(E(s_i) + E(p_i) + E(p_j)),$$

$$E(C(S')) = \sum_{\substack{k=1 \\ i \neq k \neq j}}^{n} w_k E(C_k) + w_j(E(s_i) + E(p_j)) + w_i(E(s_i) + E(p_j) + E(p_i)).$$

Le reste de la preuve découle des mêmes arguments que ceux du théorème 1. La règle optimale w_j/p_j devient donc $w_j/E(X_j)$.

Dans la classe des politiques dynamiques sans préemption, il faut montrer que la décision optimale ne change à aucune fin d'exécution. Le résultat est prouvé par récurrence sur le nombre de tâches qu'il reste à exécuter. S'il reste deux tâches à exécuter, on suit l'ordonnancement optimal statique. En effet, avec deux tâches, l'ordonnancement optimal dans la classe des politiques statiques sans préemption, l'est aussi dans la classe des politiques dynamiques sans préemption. S'il reste n tâches à exécuter, il suffit de décider quelle sera la première tâche à exécuter. En effet, une fois que l'on a choisi la première tâche, les $n-1$ tâches restantes seront exécutées suivant WSEPT d'après l'hypothèse de récurrence. Raisonnons par l'absurde : la première tâche à être exécutée n'est pas celle de plus grand $w_j/E(X_j)$. Dans ce cas, en échangeant les deux premières tâches (d'après l'hypothèse de récurrence la deuxième est celle de plus grand $w_j/E(X_j)$), on diminue le coût de ces deux tâches et donc celui de l'ordonnancement tout entier. Par conséquent, il est optimal d'exécuter les tâches suivant WSEPT dans la classe des politiques dynamiques sans préemption. □

L'exemple qui a servi d'illustration à la section 1.3 (page 15) montre que, pour des variables aléatoires quelconques, la règle WSEPT n'est pas optimale dans la classe des politiques dynamiques avec préemption. Cependant, on peut montrer que WSEPT reste optimale dans la classe des politiques dynamiques avec préemption, sous réserve que les durées d'exécution satisfassent des conditions supplémentaires. On appelle *taux de défaillance* (*hazard rate* ou *completion rate* en anglais, voir Ross 2006), noté $c(t)$, la probabilité qu'une variable aléatoire se réalise entre les instants t et $t + dt$, alors qu'elle ne s'est pas réalisée plus tôt, lorsque dt tend vers zero. Si X est une variable aléatoire continue, alors $c(t) = f_X(t)/(1 - F_X(t))$, et si X est discrète, alors $c(t) = \mathbb{P}(X = t)/\mathbb{P}(X \geq t)$. Si les durées d'exécution ont un taux de défaillance croissant (ICR : Increasing Completion Rate), alors WSEPT reste optimale. Intuitivement, un taux de défaillance croissant signifie que la tâche va se terminer d'autant plus vite que l'on avance dans son exécution. Par conséquent, une fois que l'on a pris la décision de servir une tâche, cette décision se trouve

être renforcée au cours de son exécution. La loi exponentielle a quant à elle un taux de défaillance constant, qui est égal à son taux. C'est donc un cas particulier d'une variable aléatoire ICR. Par conséquent, pour le problème $1 \mid X_j \sim \exp(\mu_j) \mid E(\sum w_j C_j)$, la règle WSEPT donne un ordonnancement optimal dans la classe des politiques dynamiques avec préemption. Les tâches sont alors exécutées suivant les $w_j \mu_j$ décroissants. Cette règle est plus connue sous la forme $c\mu$ dans la littérature sur les files d'attente.

C'est dans la littérature sur les files d'attente, que l'on voit pour la première fois apparaître la politique d'ordonnancement WSEPT, ou $c\mu$. Cobham (1954) utilise en premier cette politique pour résoudre le problème $1 \mid r_j, X_j \sim \exp(\mu_j) \mid E(\sum w_j C_j)$ dans la classe des politiques dynamiques sans préemption. Dans la communauté travaillant sur des problèmes d'ordonnancement, on trouve surtout des résultats sur des problèmes qui sont des extensions de $1 \mid\mid E(\sum w_j C_j)$. Pour une machine, on peut par exemple citer Pinedo et Rammouz (1988) qui étudient le problème $1 \mid R_j, X_j \mid E(\sum w_j C_j)$ dans lequel les tâches peuvent avoir des contraintes de précédence et la machine peut être sujette à des pannes. Schrage et Miller (1966) étudient quant à eux le problème $M/G/1$ à une seule classe de tâches, lorsque les durées d'exécution sont connues à l'arrivée des tâches et la préemption est autorisée. Ils montrent alors que la règle d'ordonnancement SRPT est optimale pour minimiser le temps moyen passé dans le système. Harrison (1975a,b) étudie une extension de ce problème. Il considère n classes de tâches, les durées d'exécution sont quelconques et la préemption n'est pas autorisée. L'objectif est de minimiser la somme actualisée de coûts de mise en attente, plus des gains pour chaque service. Il donne l'expression de ces coûts, ainsi que d'autres mesures de performances, et prouve que WSEPT est optimale.

Concernant les problèmes à plusieurs machines, les hypothèses retenues sont généralement de considérer des durées d'exécution exponentielles. Rothkopf (1966) propose un programme dynamique pour le problème $Pm \mid X_j \mid E(\sum w_j C_j)$. Pour le cas particulier où les durées d'exécution sont exponentielles et lorsque le système traite deux classes de tâches homogènes, $Pm \mid X_j \sim \exp(\mu_j) \mid E(\sum w_j C_j)$, Chang et al. (1991) montrent que la politique optimale est à seuil. Les auteurs exhibent aussi une garantie de performance de l'ordonnancement WSEPT pour ce problème : $C(\text{WSEPT})/C^* \leq 1,71$. Weiss (1992) étudie quant à lui le problème dans lequel les tâches sont exécutées par lots, $Pm \mid X_j, batch \mid E(\sum w_j C_j)$. Il montre que sous certaines conditions, la règle WSEPT est encore optimale, et que dans le cas général, WSEPT est optimale « la plupart du temps ».

Dans la littérature, des résultats existent montrant que la règle WSEPT est optimale pour le contrôle d'une partie d'un système plus complexe (Buyukkoc et al. 1985, Nain et al. 1989). Ceci montre que cette règle est efficace pour résoudre à l'optimal une grande variété de problèmes d'ordonnancement et de contrôle de files d'attente. Dans le cas où la politique de contrôle optimale est plus complexe, WSEPT est aussi souvent utilisée pour donner des solutions approchées.

2.2 Minimisation des coûts d'impatience

Dans cette section, nous présentons les problèmes d'ordonnancement ayant comme fonction objectif $\sum w_j U_j$, dans les cas déterministes et stochastiques. Nous présenterons ici essentiellement les problèmes avec IES. On omettra de préciser la présence de dates d'échéance dans la description du problème, sauf si elles respectent des contraintes particulières.

2.2.1 Cas déterministe

Dans le section 1.3.1 (page 16), nous avons utilisé l'algorithme de Moore (1968) pour mettre en évidence les différences entre politiques statiques et dynamiques. En effet, cet algorithme construit un ordonnancement initial et l'améliore à chaque étape, en identifiant quelles tâches sont en retard, jusqu'à obtenir un ordonnancement optimal. Le fait de pouvoir identifier quelles tâches sont en retard est une spécificité des politiques dynamiques. En effet, dans la classe des politiques statiques, nous n'avons accès qu'à la probabilité qu'une tâche soit en retard. Nous présentons donc l'algorithme de Moore (1968) avant de détailler les résultats de la littérature sur les problèmes déterministes minimisant les coûts d'impatience.

Théorème 3 (Moore 1968)**.** *L'algorithme de Moore donné ci-dessous minimise le nombre de tâches en retard dans le problème* $1 \mid\mid \sum U_j$.

1. $S = \emptyset$, $k = 1$.
2. *Trier les tâches dans l'ordre croissant des dates d'échéance :* $d_1 \leq d_2 \leq \cdots \leq d_n$.
3. *Si* $k \leq n$, *passer à l'étape 4. Sinon passer à l'étape 8.*
4. *Ajouter la tâche* k *à la fin de* S.
5. *Si* k *est en retard, retirer de* S *la tâche dont la durée d'exécution est la plus grande.*
6. $k \leftarrow k + 1$.
7. *Passer à l'étape 3.*
8. *Ajouter les tâches qui ont été retirées à la fin de* S.
9. *Retourner* S.

Le problème $1 \mid\mid \sum w_j U_j$, où l'on cherche à minimiser le nombre pondéré de tâches en retard, est $\mathcal{NP}$-difficile. En effet, le cas particulier $1 \mid d_j = d \mid \sum w_j U_j$ est un problème de sac-à-dos : d correspond à la taille du sac-à-dos, les durées d'exécution correspondent à la taille des objets et les poids correspondent à l'utilité des objets.

Compte tenu de la complexité de ce problème et des nombreuses variantes qui ont été étudiées, nous avons choisi de présenter les principaux résultats de ce problème par catégories. Ces catégories dépendent des techniques de résolution utilisées. Nous présentons des résultats existants sur des cas particuliers polynomiaux de ce problème, des algorithmes

d'approximation, des programmes dynamiques, des solutions exactes basées sur des programmes linéaires et enfin des heuristiques.

Problèmes polynomiaux. Nous présentons ici des cas particuliers du problème d'ordonnancement avec dates d'échéance. Les résultats sont donnés par complexité croissante des algorithmes proposés. On voit que le problème devient difficile lorsque les tâches ont des poids différents, ou lorsque les tâches ont des durées de disponibilité non nulles.

- Moore (1968) propose le premier algorithme pour résoudre un problème avec dates d'échéance lorsque les poids des tâches sont égaux : $1 \mid\mid \sum U_j$. L'algorithme qu'il propose termine en $O(n \log n)$ et est donné dans le théorème 3.
- Lawler (1994) propose un algorithme efficace pour résoudre ce problème lorsque les tâches ont des poids différents. Pour obtenir un algorithme polynomial, il suppose qu'il existe un ordre liant les poids et les durées d'exécution : si $p_i \leq p_j$ alors $w_i \leq w_j$. Pour le problème $1 \mid p_i \leq p_j \Rightarrow w_i \geq w_j \mid \sum w_j U_j$, l'algorithme qu'il propose termine en $O(n \log n)$.
- Kise et al. (1978) utilisent le même type de contraintes supplémentaires, liant cette fois les dates de disponibilité et les dates d'échéance pour proposer un algorithme résolvant $1 \mid r_i \leq r_j \Rightarrow d_i \leq d_j \mid \sum U_j$ en $O(n^2)$. Récemment, Li et al. (2010) ont montré que la preuve de Kise et al. était incorrecte et Fan et al. (2012) ont corrigé cette preuve. Par conséquent l'algorithme de Kise et al. reste correct.
- Carlier (1981) propose un algorithme pour le cas particulier où les durées d'exécution et les poids sont égaux, $1 \mid p_j = p, r_j \mid \sum U_j$, en $O(n^3 \log n)$.
- Baptiste (1999a) étudie une variante du problème de Carlier (1981), les durées d'exécution ne sont plus égales, mais la préemption est autorisée : $1 \mid pmtn, r_j \mid \sum U_j$. L'algorithme proposé termine en $O(n^4)$ et nécessite une mémoire en $O(n^3)$.
- Baptiste (1999b) donne aussi des algorithmes pour les versions pondérées de ce problème, les durées d'exécution étant encore égales. Lorsque la préemption n'est pas autorisée, $1 \mid p_j = p, r_j \mid \sum w_j U_j$, l'algorithme proposé termine en $O(n^7)$ et lorsque la préemption est autorisée, $1 \mid p_j = p, pmtn, r_j \mid \sum w_j U_j$, il termine en $O(n^{10})$.

Algorithmes d'approximation. Dans cette partie, nous donnons des références proposant des FPTAS pour le problème de sac-à-dos et le problème d'ordonnancement avec dates d'échéance. Un FPTAS est un algorithme qui donne une solution S, telle que son coût $C(S)$ ait une erreur relative inférieure à ϵ par rapport à l'optimal : $C(S) \leq C(S^*)(1+\epsilon)$, en temps polynomial en $1/\epsilon$ et en la taille de l'instance.

- Ibarra et Kim (1978) donnent un algorithme d'approximation polynomial pour le problème de sac-à-dos ($1 \mid d_j = d \mid \sum w_j U_j$) en $O(n^2/\epsilon)$.
- Gens et Levner (1981) proposent un algorithme qui résout le problème $1 \mid\mid \sum w_j U_j$ avec une précision ϵ en temps $O(n^2 \log n + n^2/\epsilon)$ et sur un espace $O(n^2/\epsilon)$.

Programmes dynamiques. Dans cette partie, nous présentons quelques résultats utilisant des programmes dynamiques. Ils donnent donc une solution optimale, mais ne sont pas polynomiaux.

- Lawler et Moore (1969) donnent un programme dynamique pour résoudre le problème $1 \mid\mid \sum w_j U_j$ en temps $O(n \sum p_j)$.
- Lawler (1990) considère une version plus générale de ce problème, avec des dates de disponibilité et dans laquelle la préemption est autorisée : $1 \mid pmtn, r_j \mid \sum w_j U_j$. Il donne un programme dynamique en temps $O(nk^2W^2)$ et en espace $O(k^2W)$, où k est le nombre de dates de disponibilité différentes et W est la somme des poids (tous entiers).

Solutions exactes basées sur des programmes linéaires. Les références données ci-dessous utilisent toutes des relaxations de programmes linéaires qui permettent d'accélérer des techniques de résolution exacte, tel que le *Branch and Bound*. Pour le problème $1 \mid\mid \sum w_j U_j$, sans dates de disponibilité, les auteurs sont capables de résoudre des instances allant jusqu'à 1000 tâches, alors que pour le problème avec des dates de disponibilité, $1 \mid r_j \mid \sum w_j U_j$, la taille des instances que les auteurs arrivent à résoudre diminue jusqu'à 100 tâches.

- Potts et Van Wassenhove (1988) traitent le problème $1 \mid\mid \sum w_j U_j$, ils utilisent une relaxation linaire pour obtenir une borne inférieure au problème, ce qui leur permet d'éliminer des tâches en retard. Cette borne est alors utilisée dans un algorithme de Branch and Bound. Les auteurs arrivent à résoudre des instances contenant jusqu'à 1000 tâches.
- Dauzère-Pérès et Sevaux (2003b) étudient le problème $1 \mid r_j \mid \sum w_j U_j$, ils utilisent une relaxation lagrangienne d'un programme linéaire en variables mixtes. Les auteurs arrivent à résoudre des instances contenant jusqu'à 100 tâches.
- Péridy et al. (2003) utilisent aussi une relaxation lagrangienne dans un algorithme de Branch and Bound, en optimisant une partie de l'ordonnancement dans le problème $1 \mid r_j \mid \sum w_j U_j$. Leur solution est elle aussi efficace jusqu'à 100 tâches.
- M'Hallah et Bulfin (2007) traitent aussi le problème $1 \mid r_j \mid \sum w_j U_j$, avec les mêmes techniques que celles présentées précédemment. Selon les auteurs, la méthode qu'ils proposent est la plus efficace en terme de durées d'exécution et de taille d'instances traitées. La taille des instances traitées est toujours de l'ordre de 100 tâches.
- Sadykov (2008) étudie le problème $1 \mid r_j \mid \sum w_j U_j$, en partant de l'algorithme de Carlier (1981), il propose des coupes d'infaisabilité pour ce problème.

Heuristiques et méta-heuristiques. Les heuristiques et méta-heuristiques fournissent des solutions sans garanties de performance, mais ont l'avantage de s'exécuter rapidement et de ne pas être limitées par la taille des instances traitées.

- Dauzère-Pérès (1995) propose une heuristique qu'il utilise comme borne inférieure dans une méthode exacte pour la résolution du problème $1 \mid r_j \mid \sum U_j$.
- Dauzère-Pérès et Sevaux (2003a) donnent un algorithme génétique pour le problème $1 \mid r_j \mid \sum w_j U_j$.

2.2.2 Cas stochastique en ordonnancement

Avant de passer en revue la littérature sur les problèmes d'ordonnancement minimisant le nombre pondéré de tâches en retard, nous allons présenter une preuve, détaillée par exemple par Pinedo (2008), utilisant un argument d'échange, ainsi qu'une récurrence. Cette preuve reprend le même plan que celle du théorème 2. Cependant, pour mener la preuve jusqu'au bout, il est nécessaire qu'il y ait une date d'échéance commune à toute les tâches. Dans le chapitre 4, nous allons reprendre et modifier cette preuve, ce qui nous permet de donner des ordonnancements optimaux pour des problèmes plus généraux.

Théorème 4. *La règle WSEPT est optimale pour minimiser l'espérance pondérée du nombre de tâches en retard pour le problème* $1 \mid X_j \sim \exp(\mu_j), d_j = d \mid E(\sum w_j U_j)$, *avec IES, dans les classes de politiques statiques et dynamiques, avec ou sans préemption.*

Démonstration. Pour ce problème en particulier, il n'est pas nécessaire de distinguer l'impatience et l'abandon car toutes les tâches ont la même date d'échéance. En effet, passé la date d, toutes les tâches sont en retard. Le détail des calculs est donné dans l'annexe B.2.

Considérons deux tâches 1 et 2 et montrons que WSEPT est optimale pour minimiser la fonction objectif. On compare le coût des deux ordonnancements statiques possibles : $\{1,2\}$, dont le coût est noté $C(1,2)$, et $\{2,1\}$, dont le coût est noté $C(2,1)$. Les coûts de ces deux ordonnancements sont

$$C(1,2) = (w_1+w_2)e^{-\mu_1 d} - \frac{w_2\mu_1}{\mu_1-\mu_2}\left(e^{-\mu_1 d} - e^{-\mu_2 d}\right) \qquad \text{et}$$
$$C(2,1) = (w_1+w_2)e^{-\mu_2 d} - \frac{w_1\mu_2}{\mu_1-\mu_2}\left(e^{-\mu_1 d} - e^{-\mu_2 d}\right).$$

Lorsque l'on fait la différence entre ces deux coûts on trouve

$$C(2,1) - C(1,2) = (w_2\mu_2 - w_1\mu_1)\underbrace{\frac{e^{-\mu_1 d} - e^{-\mu_2 d}}{\mu_1-\mu_2}}_{\leq 0}.$$

Par conséquent, il est optimal d'exécuter en premier la tâche 1 si et seulement si $w_1\mu_1 \geq w_2\mu_2$.

En procédant par récurrence sur le nombre de tâches restantes, on montre que WSEPT est optimale pour exécuter n tâches dans la classe des politiques statiques sans préemption. Supposons que WSEPT soit optimale pour résoudre un problème à n tâches et que l'on ait un problème à $n+1$ tâches à traiter. Une fois que l'on a décidé quelle sera la première tâche à être exécutée, les n suivantes seront exécutées suivant l'ordre WSEPT. Or, d'après l'argument d'échange décrit ci-dessus, la première tâche à être exécutée doit être celle de plus petit $w_j\mu_j$ afin de minimiser l'espérance des coûts des deux premières tâches. Par conséquent WSEPT est optimale quel que soit le nombre de tâches à exécuter.

De plus, le règle optimale étant indépendante de la date d'échéance d, il reste optimal d'exécuter les tâches selon la règle WSEPT à chaque fin de service. WSEPT est donc

optimale dans la classe des politiques dynamiques sans préemption.

Enfin, les durées d'exécution étant sans mémoire, s'il a été décidé à l'instant t d'exécuter la tâche j, il n'est pas nécessaire de révoquer cette décision jusqu'à la fin de l'exécution de cette tâche. Par conséquent, WSEPT est optimale dans la classe des politiques dynamiques sans préemption. □

Maintenant que nous avons présenté ce résultat classique, sur lequel nous allons nous appuyer par la suite, nous allons détailler les problèmes d'ordonnancement étudiés dans la littérature qui sont particulièrement proches de ceux qui nous intéressent dans cette thèse. Nous étendons les résultats de Pinedo (1983) et de Boxma et Forst (1986) dans le chapitre 3 pour des problèmes plus généraux ainsi que pour des problèmes avec IBS. Le papier de Argon et al. (2008) est plus proche du problème que nous traitons dans le chapitre 4.

Problèmes proches. Nous allons décrire les problèmes suivants plus en détails, car ce sont les plus proches des modèles que nous étudions dans cette thèse. En effet, ces articles traitent de la minimisation du nombre pondéré de tâches en retard, lorsque les durées d'exécution et les dates d'échéance sont aléatoires.

- Pinedo (1983) étudie deux problèmes d'ordonnancement stochastique sans préemption, lorsque les durées d'exécution sont exponentielles. Pour les deux problèmes, Pinedo prouve ses résultats par un argument d'échange. Concernant le problème $1 \mid X_j \sim \exp(\mu_j), D_j \sim F_D \mid E(\sum w_j U_j)$, il prouve que l'ordonnancement WSEPT est optimal dans la classe des politiques statiques. Dans le cas où les tâches ont toutes une date d'échéance commune, $1 \mid X_j \sim \exp(\mu_j), D_j = D \sim F_D \mid E(\sum w_j U_j)$, il montre que WSEPT est optimale dans la classe des politiques dynamiques. Par rapport au problème statique, l'hypothèse supplémentaire d'une échéance commune permet de garantir qu'un ordonnancement statique reste optimal ; sans cela, WSEPT peut conduire à exécuter des tâches en retard.
- Boxma et Forst (1986) utilisent aussi une preuve par argument d'échange. Ils donnent des ordonnancements statiques sans préemption optimaux pour plusieurs cas particuliers. Ils donnent une autre démonstration pour le problème traité par Pinedo (1983), $1 \mid X_j \sim \exp(\mu_j), D_j \sim F_D \mid E(\sum w_j U_j)$, et étudient des variantes de ce problème, comme par exemple $1 \mid X_j \sim F_X, D_j \sim \exp(\gamma) \mid E(\sum w_j U_j)$.
- Argon et al. (2008) considèrent un problème d'ordonnancement dynamique sans préemption à une machine avec IBS, dans lequel toutes les n tâches sont disponibles à $t = 0$, leur objectif est de minimiser l'espérance du nombre de tâches en retard : $1 \mid X_j \sim \exp(\mu_j), D_j \sim \exp(\gamma_j) \mid \sum \tilde{U}_j$. Comme le système est markovien (toutes les variables aléatoires sont exponentielles), ils déduisent certains résultats de l'écriture de la fonction objectif sous la forme d'un programme dynamique stochastique. En particulier, pour deux classes de tâches, les auteurs prouvent que si $\mu_1 \geq \mu_2$ et $\gamma_1 \geq \gamma_2$, alors il est optimal de servir en priorité les tâches de la classe 1. Ils montrent aussi que si n tend vers l'infini, la règle SEPT est optimale. Dans le cas où

les tâches ne respectent pas les deux conditions citées ci-dessus, sans perte de généralité : $\mu_1 \leq \mu_2$ et $\gamma_1 \geq \gamma_2$, les auteurs montrent qu'avec les conditions supplémentaires $\mu_1 < \mu_2 \leq \gamma_2$ et $\mu_1 \leq \gamma_1$, la politique optimale est une politique à seuil et les tâches de la classe 2 sont prioritaires après ce seuil. Sous certaines conditions particulières, les auteurs donnent une expression de la position de ce seuil.

Le reste des résultats de la littérature en ordonnancement que nous avons jugé pertinent de présenter est séparé en deux parties. La première considère des problèmes d'ordonnancement avec des dates d'échéance déterministes et la deuxième considère des fonctions objectif définies à l'aide d'intervalles de confiance.

Dates d'échéance déterministes. Traditionnellement, les dates d'échéance modélisent des dates de livraison, ou de fin de production. Par conséquent, il existe des résultats dans lesquels les durées d'exécution sont aléatoires mais les dates d'échéance sont déterministes.

- Pinedo et Rammouz (1988) étudient les problèmes $1 \mid R_j, X_j \mid E(\sum w_j C_j)$ et $1 \mid R_j, X_j, d_j \mid E(\sum w_j U_j)$, dans lesquels les tâches peuvent avoir à respecter des contraintes de précédences et la machine peut être sujette à des pannes.
- Jang (2002) et Jang et Klein (2002) étudient le problème $1 \mid R_j, X_j, d_j \mid E(\sum U_j)$. Ils proposent une politique myope, c'est-à-dire un algorithme glouton qui minimise uniquement le coût induit par la prochaine tâche à exécuter. Pour des durées d'exécution suivant des lois normales, $X_j \sim \mathcal{N}(\mu_j, \sigma_j)$, la politique myope est optimale si 1) $\sigma_i \geq \sigma_j$ ou 2) $\sigma_i < \sigma_j$ et $\mu_j \sigma_i \geq \mu_i \sigma_j$. Dans la classe des politiques dynamiques, seule la tâche 1 doit satisfaire ces conditions pour induire l'optimalité ; dans la classe des politiques statiques ces conditions doivent être satisfaites par toutes les tâches.
- Seo et al. (2005) étudient le problème $1 \mid X_j \sim \mathcal{N}, d_j = d \mid E(\sum w_j U_j)$, où les durées d'exécution sont indépendantes, identiquement distribuées et suivent une loi normale. Ils utilisent un programme non linéaire en nombres entiers et des relaxations pour prouver qu'une politique statique est optimale.

Notons que dans les deux dernières références, les durées d'exécution sont des variables aléatoires suivant des lois normales. Or, les lois normales ont comme support $\mathbb{R}$, ce qui est incompatible avec des durées qui devraient avoir un support positif. Ce point n'est pas discuté dans ces articles.

Ordonnancements avec intervalles de confiance. Dans les deux articles cités ci-dessous, la fonction objectif est la minimisation du nombre pondéré de tâches en retard, avec un intervalle de confiance sur les retards. La valeur de l'ordonnancement S est augmentée de w_j, si la tâche j est en retard avec une probabilité supérieure à α (α faisant partie de l'instance).

- van den Akker et Hoogeveen (2008) étudient, entre autres, le problème $1 \mid X_j \mid \sum U_j$. Grâce à l'utilisation des intervalles de confiance, ils peuvent réutiliser des ordonnancements optimaux issus de la littérature sur les problèmes déterministes, pour créer des ordonnancements statiques optimaux pour le problème stochastique.

– Trietsch et Baker (2008) étendent les résultats de van den Akker et Hoogeveen (2008) pour d'autres problèmes d'ordonnancement.

2.2.3 Cas stochastique en théorie des files d'attente

Nous avons choisi de séparer les résultats en files d'attente en plusieurs catégories. La première concerne les problèmes les plus proches de ceux que l'on considère. C'est-à-dire des systèmes à plusieurs classes de tâches, dans lesquels il faut décider de quelle sera la prochaine tâche à exécuter.

Plusieurs classes de tâches.

– Atar et al. (2010) donnent une politique de service optimale pour le problème de contrôle de files d'attente $M/M/m+M$ avec n classes de tâches. Un coût de mise en attente c_j est payé par tâche en attente et par unité de temps. Un coût w_j est payé pour chaque abandon. Ils prouvent tout d'abord que lorsque toutes les durées sont exponentielles, les coûts d'abandon sont équivalents aux coûts de mise en attente en posant pour tout j, $w_j = c_j/\gamma_j$. Ils supposent que la file d'attente est dans un régime appelé *limite fluide* et saturé (il y a toujours beaucoup de tâches dans le système). Dans ce régime, on suppose que l'on peut exécuter des fractions de tâches. De cette manière, le comportement du système correspond à celui de l'écoulement d'un fluide. Sous ces hypothèses, donner la priorité à la classe de plus grand $c_j\mu_j/\gamma_j$ minimise les coûts moyens de mise en attente. En considérant l'écriture avec des coûts d'abandon, la politique optimale consiste à donner la priorité à la classe de plus grand $w_j\mu_j$ qui est la règle WSEPT. Dans ce cas, la règle WSEPT est optimale.
– Benjaafar et Elhafsi (2011) étudient un problème de gestion de stock où deux classes de clients se partagent un unique produit. Les clients de la classe 2 ne sont pas sujets à l'impatience (ils génèrent uniquement des coûts de mise en attente), tandis que les clients de l'autre classe doivent être servis immédiatement avec des produits en stock (dans le cas contraire ils génèrent un coût d'abandon). L'objectif est de contrôler les rejets, les mises en attente, la production et l'allocation des stocks pour minimiser les coûts actualisés. Ils montrent entre autre que l'allocation des stocks ne suit pas une politique statique, mais dépend du nombre de clients de chaque classe en attente.
– Down et al. (2011) étudient une file d'attente $M/M/1+M$ avec 2 classes de tâches et des durées d'exécution égales ($\mu_1 = \mu_2 = \mu$). Ils considèrent deux modèles ; le premier comporte des coûts de mise en attente et des coûts d'abandon, que nous pouvons réécrire en n'utilisant que des coûts d'abandon (ce n'est valable que lorsque l'objectif est de minimiser les coûts moyens) ; le deuxième modèle comporte uniquement un revenu r_j pour chaque tâche exécutée et l'objectif est de maximiser les revenus moyens sur un horizon infini. Ce modèle est équivalent à un modèle avec coûts d'abandon. En effet, si l'on considère des arrivées avec un taux λ_j, qu'on les sépare en la fraction des tâches qui sont servies λ_j^S et la fraction des tâches qui abandonnent λ_j^A, on a $\lambda_j = \lambda_j^S + \lambda_j^A$. Et la fonction objectif s'écrit $\max\{\sum r_j\lambda_j^S\} = \max\{\sum r_j(\lambda_j -$

$\lambda_j^A)\} = \sum r_j\lambda_j - \min\{\sum r_j\lambda_j^A)$. Par conséquent, maximiser les revenus est équivalent à minimiser les coûts d'abandon. Les auteurs prouvent que si 1) $\gamma_1 \geq \gamma_2$ et $w_1 \geq w_2$, ou si 2) $w_1\gamma_1 \geq w_2\gamma_2$ et $\gamma_1 \leq \gamma_2$, alors il est optimal de servir en priorité les tâches de la classe 1.

– Blok et al. (2012) étudient une file d'attente $M(n)/M/1+M$ avec n classes de tâches. Le modèle qu'ils étudient est le même que celui de Down et al. (2011) à l'exception du nombre de classes de tâches et du processus d'arrivée. En effet, les taux d'arrivée ne sont pas constants mais diminuent lorsque le nombre de tâches en attente augmente. Ils montrent alors que si 1) $w_1\gamma_1\mu_1 \geq w_2\gamma_2\mu_2 \geq \cdots \geq w_n\gamma_n\mu_n$, 2) $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_n$ et 3) $w_1\gamma_1 \geq w_2\gamma_2 \geq \cdots \geq w_n\gamma_n$ il est optimal d'exécuter les tâches dans l'ordre croissant des indices.

Dans la suite, nous donnons les principaux résultats existants sur des mesures de performance de systèmes de files d'attente. Dans ce cas, on suppose que le comportement du système, ainsi que la politique de service, sont fixés et le but est de donner des expressions pour calculer différents indicateurs de performance. Dans cette catégorie, on fera la distinction entre les résultats concernant une classe de tâches et ceux concernant plusieurs classes de tâches. Enfin, nous présentons des résultats sur le contrôle optimal des files d'attente. Là encore, on distingue les résultats concernant une classe et ceux concernant deux classes de tâches. Lorsqu'il n'y a qu'une classe de tâches dans le système, les décisions à prendre peuvent être l'ordre d'exécution des tâches. En effet, la règle d'ordonnancement optimale n'est pas toujours celle du premier entré, premier sorti. Une autre décision que l'on peut être amené à prendre, est celle de l'admission d'une nouvelle tâche dans le système. Dans une catégorie à part, nous présenterons des résultats sur des systèmes plus complexes, comme des systèmes de production ou des centres d'appels téléphoniques, qui peuvent être mis en relation avec notre problème.

Mesures de performance à une classe de tâches.

– Movaghar (1998) donne des mesures de performance pour le problème de files d'attente $M(n)/M/1+G$, avec IBS lorsque la préemption est autorisée. Les tâches sont exécutées suivant la politique FIFO (*first in first out*). Les taux d'arrivée dans le système étudié diminuent avec le nombre de tâches dans le système.
– Movaghar (2006) donne des mesures de performance pour le problème de files d'attente $M(n)/M/m+G$, avec IES lorsque la préemption est autorisée. Là aussi, les tâches sont exécutées suivant la politique FIFO. Les différences avec la précédente référence sont d'une part le fait que ce système a plusieurs machines et d'autre part que l'impatience est du type IES.
– Artalejo et Pla (2009) étudient le système $M/M/m+M$ modélisant plus particulièrement un centre d'appel, où les tâches peuvent être sujettes à des rejets à l'entrée du système et par la suite, peuvent se représenter dans le système.
– Boxma et al. (2011) quant à eux, donnent des mesures de performance pour le problème $M/G/1+G$ et fournissent une revue de littérature conséquente sur les mesures

de performance dans les systèmes de files d'attente avec abandons.

Mesures de performance à plusieurs classes de tâches.

- Iravani et Balcioglu (2008) étudient trois configurations différentes d'une file d'attente $M/G/1+M$ et en donnent des mesures de performance. La première configuration comporte deux classes de tâches et le système donne une priorité stricte aux tâches de la classe 1. La préemption est autorisée. Dans la deuxième configuration, la préemption n'est plus autorisée mais seulement une des classes de tâches est sujette à l'impatience. La troisième concerne un système à plusieurs machines, dans lequel les tâches qui sont impatientes peuvent redevenir disponibles passé un certain temps. Ce dernier système peut par exemple modéliser un centre d'appel téléphonique pouvant rappeler les clients.
- Zeltyn et al. (2009) donnent des mesures de performance pour n classes de tâches sur des serveurs en parallèle de même taux. Les tâches sont traitées dans l'ordre des indices, avec une priorité stricte pour les tâches de classe d'indice le plus faible.

Choix de l'ordre d'exécution à une classe de tâches.

- Panwar et al. (1988) étudient une file d'attente $M/G/1+G$ sans préemption et avec IBS. Ils montrent que la politique de service EDD minimise la proportion de tâches qui quittent le système suite à un abandon.
- Kallmes et al. (1989) étudient une file d'attente $G/G/1$ avec abandons et IES. La fonction objectif est aussi la minimisation du pourcentage de tâches qui abandonnent. Lorsque les dates d'échéance sont des variables aléatoires i.i.d. avec une fonction de répartition concave, la politique LIFO (*last in first out*) est optimale dans la classe des politiques sans préemption. Le même résultat est montré pour la file $G/M/1$ dans la classe des politiques avec préemption.
- Zhao et al. (1991) étudient eux aussi une file $G/G/1$ avec abandons, IBS et sans préemption, avec la même fonction objectif que précédemment. Cependant, dans leur modèle, il est possible de rejeter une tâche à son arrivée avant qu'elle n'entre dans la file d'attente. Ils montrent aussi que la politique LIFO est optimale si les dates d'échéance sont i.i.d. avec une fonction de répartition concave.

Contrôle d'admission à une classe de tâches.

- Lillo (2001) étudie le cas d'une file d'attente $M/G/1$ sans préemption avec 2 classes de tâches. Les tâches de la classe 1 sont servies directement à leur arrivée ou bien elles quittent le système. Les tâches de la classe 2 ne sont pas sujettes à l'impatience, mais ont un coût de mise en attente. Les décisions que l'on peut prendre dans le système sont de choisir si l'on exécute une tâche de la classe 2 ou non et de décider de mettre en marche ou d'éteindre le serveur (entraînant un certain coût). L'objectif est de minimiser les coûts moyens.
- Movaghar (2005) optimise l'admission pour une file d'attente $M(n)/M/m+G$, où à chaque serveur est associée une file d'attente dédiée. L'objectif est encore de minimiser le pourcentage des tâches qui abandonnent. Avec IES comme avec IBS, l'auteur

montre que la politique consistant à joindre la file d'attente contenant le moins de tâches est optimale.

– Ward et Kumar (2008) étudient la file d'attente $G/G/1$ avec abandons en régime saturé (il y a toujours beaucoup de tâches dans la file d'attente). Ils utilisent une technique de diffusion pour trouver la politique d'admission qui minimise les coûts actualisés de rejet et d'abandon pour ce système.

Contrôle de systèmes de production à une classe de tâches.

– Zeltyn et Mandelbaum (2005, 2009) étudient le cas d'un centre d'appel téléphonique, où l'objectif est de minimiser les coûts générés par le personnel tout en respectant une certaine qualité de service. Ils modélisent ce problème par une file d'attente $M/M/n+G$ qu'ils étudient sous différents régimes de saturation.

– Benjaafar et al. (2010) étudient un problème de gestion des stocks. Des produits sont fabriqués par une machine et placés dans un stock. Lorsqu'un client place une demande pour un produit, celle-ci est soit satisfaite immédiatement s'il y a un produit en stock, soit la demande est mise en attente. Une demande peut être perdue si elle passe trop de temps en attente. L'objectif est de minimiser les coûts actualisés de mise en attente, de stockage et de rejet. Les auteurs prouvent que la politique optimale possède deux seuils, un seuil de stock maximum à ne pas dépasser (passé ce seuil, on ne produit plus) et un nombre de demandes en attente maximum (passé ce seuil, les nouvelles demandes sont rejetées). Lorsque le niveau de stock est nul et qu'il y a des demandes en attente, on peut voir ce problème comme un contrôle d'admission dans une file d'attente avec abandons.

– Economopoulos et al. (2011) étudient une extension du modèle de Benjaafar et al. (2010) où les clients peuvent d'eux-même décider de ne pas joindre la file si le nombre de demandes en attente est trop important. Les auteurs prouvent les mêmes résultats que Benjaafar et al. (2010).

2.2.4 Une preuve alternative pour la file $M/M/1+M$ avec deux classes de tâches

Dans cette section, nous présentons une modélisation d'un problème avec abandons sous forme de MDP (processus de décision de Markov). La modélisation et le résultat que nous présentons sont identiques à ceux donnés par Down et al. (2011). En revanche, la preuve que nous donnons en est une variante simplifiée. Les MDP sont utilisés pour modéliser des systèmes sans mémoire, toutes les lois considérées seront donc exponentielles. De plus, cette technique nous permet uniquement de traiter le cas où les durées d'exécution sont exponentielles et de même taux. En effet, la preuve n'est plus valide lorsque les taux d'exécution sont différents. Une fois un problème modélisé à l'aide d'un MDP, on peut donner une politique optimale pour ce problème, à une précision près, à l'aide d'un algorithme d'itération sur la valeur (e.g. Puterman 1994). Nous utiliserons de tels algorithmes, notamment pour les résultats numériques donnés dans les chapitres 4 et 5.

On considère un problème de files d'attente avec deux classes de tâches, les classes sont notées j, avec $j \in \{1, 2\}$. Les tâches deviennent disponibles suivant un processus de Poisson de taux λ_j et sont placées dans une file d'attente F_j. Les tâches présentes dans la file d'attente patientent pendant une durée exponentielle de taux γ_j avant la fin de leur service. Si elles ne sont pas exécutées avant cette date, elles quittent le système avec un coût w_j. Le système compte une seule machine qui exécute une tâche en une durée exponentielle de taux μ, les deux classes de tâches se partagent donc cette machine. À chaque instant, la variable de décision consiste à choisir à quelle file on affecte la machine, afin de minimiser les coûts actualisés sur un horizon infini. Contrairement aux résultats présentés précédemment, on ne cherche pas à minimiser l'espérance du coût d'abandon (celui-ci tendant vers l'infini), mais bien l'espérance des coûts actualisés. La préemption est autorisée.

Une tâche de la classe j est définie par :

- λ_j : son taux d'arrivée ;
- γ_j : son taux d'impatience ;
- w_j : son coût d'impatience.

Le taux de service μ est le même pour les deux classes de tâches.

L'état du système est décrit par le nombre de tâches présentes à chaque instant t et est noté $x(t) = (x_1(t), x_2(t))$, où $x_j(t)$ est le nombre de tâches de la classe j présentes dans le système à l'instant t. Comme toutes les durées sont exponentielles, le système que l'on considère est sans mémoire et on peut s'affranchir de la variable temporelle pour le décrire. L'état du système sera par la suite noté $x = (x_1, x_2)$ et on notera les vecteurs de base $e_1 = (1, 0)$ et $e_2 = (0, 2)$. L'espace d'état est donc $\mathbb{N} \times \mathbb{N}$. Pour pouvoir résoudre ce problème, on a besoin d'un espace d'état fini. On borne donc la taille des files d'attente par un entier M_j, pour que la taille de l'espace d'état devienne $M_1 \times M_2$. Si une tâche de la classe j entre dans le système alors que la file F_j est pleine, elle est rejetée. Ces bornes constituent une approximation raisonnable de ce problème, car nous pouvons les faire tendre vers l'infini et ainsi tendre vers le problème non borné. Tronquer l'espace d'état est une limitation expérimentale naturelle, il faudra cependant choisir judicieusement les bornes de cet espace d'état, pour que la troncature n'ai qu'un faible impact sur les résultats numériques de l'instance considérée. Pour chacune des classes de tâches, la probabilité qu'il y ait x_j tâches dans la file est bornée par

$$\pi_{x_j} = \frac{(\lambda_j/\gamma_j)^{x_j}}{x_j!} e^{-\lambda_j/\gamma_j}.$$

Cette expression est celle des probabilités stationnaires d'un processus de naissance et mort (voir annexe B.1). On peut donc a priori borner l'espace d'état, de telle sorte que la probabilité d'arriver aux bornes reste inférieure à un certain ϵ que l'on s'est fixé.

En utilisant le cadre d'étude proposé par Koole (2006), le coût de la politique optimale

vérifie

$$v^*(x) = \frac{1}{C} T_{IES} v^*(x),$$

avec le méta-opérateur T_{IES} défini par

$$T_{IES}v(x) = \mu T_{\text{prod}}v(x) + \sum_{j\in\{1,2\}} \left(\gamma_j T^j_{\text{imp}}v(x) + \lambda_j T^j_{\text{arr}}v(x)\right) \tag{2.1}$$

en notant $C = \alpha + \beta$, avec α le taux d'actualisation et $\beta = \mu + \sum_{j\in\{1,2\}} (\lambda_j + M_j\gamma_j)$. Comme il est toujours possible de redimensionner l'échelle de temps de telle sorte que $\alpha + \beta = 1$, l'équation d'optimalité vérifie $v^*(x) = T_{IES}v^*(x)$.

On définit alors les opérateurs

$$T_{\text{prod}}v(x) = \begin{cases} \min\{v(x-e_1); v(x-e_2); v(x)\} & \text{si} \quad x_1 > 0,\, x_2 > 0 \\ \min\{v(x-e_2); v(x)\} & \text{si} \quad x_1 = 0,\, x_2 > 0 \\ \min\{v(x-e_1); v(x)\} & \text{si} \quad x_1 > 0,\, x_2 = 0 \\ v(x) & \text{si} \quad x_1 = 0,\, x_2 = 0 \end{cases},$$

$$T^j_{\text{imp}}v(x) = \begin{cases} x_j(v(x-e_j) + w_j) + (M_j - x_j)v(x) & \text{si} \quad x_j > 0 \\ M_j v(x) & \text{si} \quad x_j = 0 \end{cases},$$

$$T^1_{\text{arr}}v(x) = \begin{cases} v(x+e_1) & \text{si} \quad x_1 < M_1 \\ v(x) + w_1 & \text{si} \quad x_1 = M_1 \end{cases} \text{ et}$$

$$T^2_{\text{arr}}v(x) = \begin{cases} v(x+e_2) & \text{si} \quad x_2 < M_2 \\ v(x) & \text{si} \quad x_2 = M_2 \end{cases}.$$

T_{prod} correspond à la décision de production, c'est la variable de décision. Dans le cas où x_1 et x_2 sont strictement positifs, on choisit la décision minimisant les coûts entre : exécuter une tâche de la classe 1 ($v(x-e_1)$) ; exécuter une tâche de la classe 2 ($v(x-e_2)$) ; ne rien faire ($v(x)$). T^i_{imp} correspond à l'impatience des tâches de la classe i, dans ce cas un coût w_i est imputé au système. S'il y a x_j tâches de la classe j, le coût engendré est de $x_j w_j$ car x_j tâches peuvent abandonner. De manière à garder un taux de transition constant sur tout l'espace d'état, nous ajoutons $M_j - x_j$ transitions fictives. T^i_{arr} correspond à l'arrivée d'une tâche de la classe i. Si une tâche de la classe i arrive dans le système alors qu'il y a déjà M_i tâches de cette classe, elle est rejetée avec un coût w_1 pour la classe 1 et sans coût supplémentaire pour la classe 2. Cette hypothèse doit être posée pour mener la preuve suivante jusqu'à son terme. Cette hypothèse n'influe pas sur l'étude théorique du problème car on peut faire tendre les bornes de l'espace d'état vers l'infini. En revanche, il n'est pas nécessaire d'introduire ce coût supplémentaire lors des études numériques.

En introduisant les variations suivant les deux indices : $\Delta_i v(x) = v(x + e_i) - v(x)$,

on peut réécrire les opérateurs de production et d'impatience sous la forme suivante (les opérateurs d'arrivées restent inchangés) :

$$T_{\text{prod}}v(x) = v(x) - \begin{cases} \max\{\Delta_1 v(x-e_1); \Delta_2 v(x-e_2); 0\} & \text{si} \quad x_1 > 0,\, x_2 > 0 \\ \max\{\Delta_2 v(x-e_2); 0\} & \text{si} \quad x_1 = 0,\, x_2 > 0 \\ \max\{\Delta_1 v(x-e_1); 0\} & \text{si} \quad x_1 > 0,\, x_2 = 0 \\ 0 & \text{si} \quad x_1 = 0,\, x_2 = 0 \end{cases}$$

$$T^j_{\text{imp}}v(x) = \begin{cases} x_j(w_j - \Delta_j v(x-e_j)) + M_j v(x) & \text{si} \quad x_j > 0 \\ M_j v(x) & \text{si} \quad x_j = 0 \end{cases}.$$

Pour de tels systèmes, Down et al. (2011) prouvent le résultat suivant.

Théorème 5 (Down et al. 2011)**.** *Soit le problème d'ordonnancement* $1 \mid R_j \sim \mathcal{P}(\lambda_j), X_j \sim \exp(\mu), D_j \sim \exp(\gamma_j) \mid E(\sum w_j U_j)$ *décrit selon un processus de décision de Markov (comme ci-dessus). Si* $w_1\gamma_1 \geq w_2\gamma_2$ *et* $\gamma_1 \leq \gamma_2$ *(ce qui entraîne* $w_1 \geq w_2$*), alors la fonction de valeur optimale* $v^*(x)$ *vérifie les propriétés suivantes :*

Propriété P1 *pour tout* x *et pour* $i \in \{1,2\}$ *:* $\Delta_i v(x) = v(x+e_i) - v(x) \geq 0$ *;*

Propriété P2 *pour tout* x *et pour* $i \in \{1,2\}$ *:* $\Delta_i v(x) = v(x+e_i) - v(x) \leq w_i$ *;*

Propriété P3 *pour tout* x *:* $\Delta_1 v(x-e_1) - \Delta_2 v(x-e_2) \geq 0$.

Nous donnons quant à nous une preuve simplifiée de la propagation de ces propriétés dans l'annexe B.3.

La propriété P1 signifie qu'il est sous-optimal de ne pas produire lorsqu'il y a des tâches en attente. La propriété P2 signifie qu'il n'est jamais désirable de perdre une tâche suite à un abandon. Enfin, la propriété P3 signifie qu'il est toujours préférable d'exécuter une tâche de la classe 1, plutôt qu'une tâche de la classe 2.

Le reste de la preuve se compose des arguments suivants. Soit l'ensemble $\mathcal{U}$ des fonctions de $\mathbb{N}^2$ dans $\mathbb{R}$, telles que si $v \in \mathcal{U}$, alors, pour tout x, v vérifie les propritétés P1, P2 et P3. L'opérateur T_{IES} propage ces propriétés, comme le montrent Down et al. (2011) ou comme nous le montrons dans l'annexe B.3. Comme toute séquence de fonctions de valeurs du type $v_{n+1} = T_{IES}v_n$ converge vers la fonction de valeur optimale (Lippman 1975), on en déduit que v^* respecte ces propriétés et qu'il est optimal de donner la priorité stricte aux tâches de la classe 1. Comme l'espace d'état est fini, on en déduit les mêmes résultats concernant la minimisation du coût moyen en faisant tendre α vers 0 (Ross 1970).

Corollaire 1. *Soit le problème d'ordonnancement* $1 \mid R_j \sim \mathcal{P}(\lambda_j), X_j \sim \exp(\mu), D_j \sim \exp(\gamma_j) \mid E(\sum w_j U_j)$*. Si* $w_1\gamma_1 \geq w_2\gamma_2$ *et* $\gamma_1 \leq \gamma_2$*, alors la politique dynamique avec préemption est de donner la priorité stricte aux tâches de la classe 1.*

Problème	IES	IBS
$X_j \sim \exp(\mu_j)$, $D_j \sim \exp(\gamma_j)$	section 3.5.1	section 3.5.1
$X_j \sim F_{X_j}$, $D_j \sim F_D$	Boxma et Forst (1986) et section 3.5.2	section 3.5.2
$X_j \sim F_X$, $D_j \sim \exp(\gamma_j)$ 2 classes de tâches	section 3.5.3	section 3.5.3
$X_j \sim F_{X_j}$, $D_j \sim \exp(\gamma)$	Boxma et Forst (1986)	section 3.5.4
$X_j \sim F_X$, $D_j \sim F_D$	Boxma et Forst (1986)	section 3.5.5

TABLE 3 – Problèmes statiques étudiés

2.3 Résumé des contributions

Dans le chapitre 3, nous étudions les problèmes d'ordonnancement stochastique $1 \mid X_j, D_j \mid E(\sum w_j U_j)$ et $1 \mid X_j, D_j \mid E(\sum w_j \tilde{U}_j)$ dans la classe des politiques statiques sans préemption.

Nous montrons que dans le cas déterministe, il est équivalent de considérer de l'impatience jusqu'au début ou jusqu'à la fin du service.

Dans le cas stochastique, nous donnons des politiques optimales dans des cas plus généraux que ceux qui ont été étudiés par Pinedo (1983) et Boxma et Forst (1986). En effet, les auteurs ne considèrent que des problèmes dans lesquels les durées d'exécutions ou les dates d'échéances sont i.i.d. Nous étudions quant à nous un problème dans lequel les durées d'exécution, ainsi que les dates d'échéance, sont des variables aléatoires exponentielles de taux différents. Pour ce problème, nous donnons des ordonnancements optimaux pour IES ainsi que pour IBS. De plus, nous étudions les problèmes posés par Pinedo (1983) et Boxma et Forst (1986) avec IBS. Le résumé de nos contributions pour ces problèmes est donné dans le tableau 3.

Dans le chapitre 4, nous étudions les problèmes d'ordonnancement stochastique $1 \mid R_j, X_j, D_j \mid E(\sum w_j U_j)$ et $1 \mid R_j, X_j, D_j \mid E(\sum w_j \tilde{U}_j)$ dans la classe des politiques dynamiques avec préemption. Nous donnons des conditions pour qu'une politique de priorité stricte soit optimale. Cette politique généralise les résultats donnés par Down et al. (2011). En effet, nous considérons n classes de tâches alors qu'ils considèrent 2 classes. Dans notre cas, les durées d'exécution ne sont pas identiquement distribuées. Nous proposons aussi une preuve plus simple pour retrouver leur résultat. La politique que nous proposons est aussi en accord avec celle proposée par Atar et al. (2010). Les auteurs montrent qu'à un changement de variable près, lorsque le système est markovien, il est équivalent de considérer des coûts de mise en attente et des coûts d'abandon. Nous étendons ce résultat dans le cas où seules les dates d'échéance sont des variables aléatoires exponentielles, les autres variables peuvent avoir des distributions quelconques. Nous étudions un système plus général que celui de Argon et al. (2008) ; dans notre cas, les tâches ont des dates de disponibilité non

	Argon et al.	Atar et al.	Down et al.	chapitre 4
Impatience	IBS	IES	IES	IES et IBS
Tâches	n tâches	n classes de tâches	2 classes de tâches	n classes de tâches
Poids	$w_j = w$	w_j	w_j	w_j
Durées d'exécution	$X_j \sim \exp(\mu_j)$	$X_j \sim \exp(\mu_j)$	$X_j \sim \exp(\mu)$	$X_j \sim \exp(\mu_j)$
Disponibilités	disponibles à $t = 0$	processus de Poisson	processus de Poisson	quelconque
Régime	quelconque	limite fluide	quelconque	quelconque
Résultats applicables à toutes les instances	non	oui	non	non

TABLE 4 – Principaux résultats sur le problème dynamique

nulles et les tâches n'ont pas toutes le même poids. Dans le tableau 4, nous donnons les principales différences des modèles et des résultats obtenus entre ces articles et cette thèse.

Enfin, dans le chapitre 5, nous explorons deux pistes de recherche concernant les problèmes d'ordonnancement avec impatience. Dans le premier cas, nous considérons un problème d'ordonnancement avec plusieurs machines dans la classe des politiques statiques sans préemption. Pour ce problème, nous adaptons au cas IBS des résultats classiques traitant du problème avec IES issus de la littérature. Nous discutons aussi du cas où seule une classe de tâche se présente dans le système. Ce dernier cas reste un problème ouvert en terme de complexité des ordonnancements employés. Dans le deuxième cas, nous considérons deux problèmes d'ordonnancement dans la classe des politiques dynamiques sans préemption. Dans le premier problème, il y a une seule classe de tâches et deux machines dont les durées d'exécution sont exponentielles mais de taux différents. Dans ce cas, on peut se rapprocher d'un problème sans impatience étudié par Hajek (1984) et Koole (1995). Ces auteurs ont décrit la structure de ce problème et en ont donné la politique de service optimale. Certains de leurs résultats peuvent être étendus au problème avec impatience, mais la description complète de ce système, notamment l'existence d'un seuil pour la mise en marche du second serveur, reste un problème ouvert. Et pour finir, nous discutons rapidement du problème à deux classes de tâches sur une machine. Pour ce problème, une nouvelle question se pose : quand est-il optimal de mettre la machine en attente, alors qu'il reste des tâches à exécuter dans le système ?

Chapitre 3

Optimal static priority rules for stochastic scheduling with impatience

A preliminary version of this chapter was presented in the *INCOM 2012* conference (Salch et al. 2012a) and a final version was published in *Operations Research Letters* (Salch et al. 2013).

3.1 Introduction

In many service systems (e.g. hotlines), customers make requests and have to wait until their requests are met. If the service is not provided quickly enough, customers can decide to quit (or renege) the system. This is what happens when customers are subject to impatience. In the context of scheduling, the impatience of a customer can be modelled as a due date, because the service has to be provided before the customer becomes impatient. This due date is a random variable because one can not know *a priori* how long a customer will be ready to wait for a certain service.

Systems which deal with impatience can be separated in two categories: problems with Impatience to the End of Service (IES), traditionally considered in the scheduling literature, or with Impatience to the Beginning of Service (IBS). In problems with IES, a job has to end before its due date to be on time. This case is more consistent with production systems where a demand is satisfied when the production of the ordered item ends.

In problems with IBS, a penalty cost is incurred only when a job begins after its due date. No penalty is incurred otherwise. In particular, if the due date occurs during the process, the process can last for a long period of time without any cost. This second case is consistent with systems providing services to the customer such as call centers. In these systems, a customer is satisfied when she/he reaches the hotline. After that, one can

consider that she/he will not hang up.

A difference must also be made between impatience and abandonment. A customer is impatient when she/he considers to have been waiting for too long (incurring a cost), but nevertheless remains in the queue; if the customer actually leaves the system, then it is an abandonment. Consequently, in systems with impatience, late jobs are processed but in systems with abandonment, they are not.

In this chapter we study the problem of scheduling jobs on a single machine in order to minimize the expected weighted number of late jobs with IBS or IES and no abandonment. The optimal policy is searched among the class of static list scheduling policies. In this class of policies, a schedule is built at time zero and it cannot be changed thereafter.

3.2 Literature review

Relatively many papers consider abandonment (or reneging) in queueing systems with a single class of jobs (e.g. Zeltyn 2004, Movaghar 1998, 2006, Ward and Kumar 2008, Benjaafar et al. 2010). Several papers investigate the scheduling of different classes of jobs in a system with abandonment in order to minimize the (weighted) number of late jobs. These papers consider strict priority rules. In a strict priority rule, jobs are ordered according to their priority and whenever a job of higher priority arises, the current job is preempted to the benefit of this new job. Atar et al. (2010) prove that a strict priority rule is asymptotically optimal in an overloaded system. Panwar et al. (1988) characterize an optimal policy when all durations are known at the arrival of jobs. Down et al. (2011) consider a problem with two classes of customers, Poisson arrivals and all durations being exponential. They provide sufficient conditions for a strict priority rule to be optimal. Jang (2002) and Jang and Klein (2002) propose a heuristic for the problem of scheduling n different jobs with stochastic processing times and deterministic due dates. Seo et al. (2005) give near optimal schedules when processing times are normal random variables and when jobs have a common due date.

Some other papers consider the scheduling of jobs with impatience costs but without abandonment. Argon et al. (2008) study a scheduling problem with IBS where all jobs are available at time zero, the objective being to minimize the expected number of late jobs in the class of dynamic policies. When only two classes of customers enter the system, the authors give conditions under which a strict priority rule is optimal. Pinedo (1983) studies a stochastic IES scheduling problem on a single machine with the objective to minimize the expected weighted number of late jobs in the class of static list scheduling policies. For the particular case where processing times follow independent exponential distributions and due dates follow general distribution function, he proves that processing the jobs in non-increasing order of the ratio of their weights times their mean processing times, the so-called $c\mu$ rule, is optimal. Boxma and Forst (1986) consider the same problem for some other probability distributions. In all the results mentioned above, either processing times

or due dates are identically distributed.

In this chapter, we extend the results of Pinedo (1983) and Boxma and Forst (1986). First, we consider situations where both processing times and due dates are not identically distributed. Second, we extend their results to IBS situations. This IBS counterpart has not been studied in the literature, to the best of our knowledge.

The remainder of this chapter is organized as follows. Section 3.3 formulates the problem and introduces notations. Section 3.4 shows a summary of our results and Section 3.5 gives optimal static priority rules and their proofs.

3.3 Problem description

We consider a scheduling problem where n jobs have to be processed on a single machine. All jobs are available at time zero. A job j has a processing time X_j with mean $1/\mu_j$, a due date D_j with mean $1/\gamma_j$ and a deterministic weight w_j. We assume that all random variables are independent. A random variable Y that has a cumulative density function (c.d.f.) F_Y, is noted $Y \sim F_Y$. The probability density function (p.d.f.) of Y is noted f_Y. Especially, $Y \sim \exp(\gamma)$ means that Y is exponentially distributed with mean $1/\gamma$. A family Y_j of independent and identically distributed (i.i.d.) random variables with c.d.f. F_D will be denoted by $D_j \sim F_D$, without specifying the index of the distribution.

In the IES case, a job j is late if the end of its execution, C_j, occurs after its due date, D_j (i.e. when $C_j > D_j$). The value of the objective function for a schedule S is $C(S) = E(\sum w_j U_j)$, where U_j is assigned the value 1 if $C_j > D_j$, and 0 otherwise.

The IBS problem is similar to the IES problem except that a job j is late if the starting time of its execution, S_j, occurs after its due date, D_j (i.e. when $S_j > D_j$). The value of the objective function for a schedule S is $\tilde{C}(S) = E(\sum w_j \tilde{U}_j)$, where $\tilde{U}_j$ is assigned the value 1 if $S_j > D_j$, and 0 otherwise.

For both IES and IBS problems, our objective is to sequence the jobs in order to minimize the expected weighted number of late jobs in the class of static list scheduling policies. Since we are looking for a static policy, we are not allowed to change the schedule after time zero. Hence if a job is already late, it has to be processed in the predefined order, possibly before a job still on time. One can remark that there always exists an optimal solution without idle time. Consequently, S_j coincides with C_{j-1}, the completion time of job $j-1$.

Note that the deterministic IBS problem can be polynomially reduced to the deterministic IES problem by changing the due date d_j of the instance of the IBS problem to $d'_j = d_j + p_j$; the reverse is also true. In a stochastic setting, however, the sum of two random variables from the same class of distribution does not necessarily remain in this class of distribution. Hence, when the theoretical results depend on the selected distributions, the IBS problem can not be reduced to the IES problem, and vice-versa (see Salch et al. 2011 for details).

♯	Problem	IES	IBS
1	$X_j \sim \exp(\mu_j)$, $D_j \sim \exp(\gamma_j)$	I : $\gamma_j \nearrow$ and $w_j\gamma_j\mu_j \searrow$	II : $\gamma_j \nearrow$, $w_j\gamma_j\mu_j \searrow$ and $w_j \searrow$
2	$X_j \sim F_{X_j}$, $D_j \sim F_D$	$X_j \nearrow_{st}$ and $w_j \searrow$ (*)	$X_j \nearrow_{st}$ and $w_j \searrow$
3	$X_j \sim F_X$, $D_j \sim \exp(\gamma_j)$, 2 classes of jobs	Threshold policy	Threshold policy
4	$X_j \sim F_{X_j}$, $D_j \sim \exp(\gamma)$	$\beta_j \searrow$ (Boxma and Forst 1986)	$\beta'_j \searrow$
5	$X_j \sim F_X$, $D_j \sim F_D$	$w_j \searrow$ (Boxma and Forst 1986)	$w_j \searrow$

(*) The special case $w_j = w$ was proved by Boxma and Forst (1986)

Table 5: Optimal static list scheduling policies

3.4 Summary of results and comments

Table 5 summarizes the results of the literature together with our contributions for the IES and the IBS problems. Our results are proved in the next section.

In Problem 1, neither the processing times nor the due dates are identically distributed. To the best of our knowledge, either processing times or due dates are always assumed identically distributed in the literature. For exponential distributions, we show that if jobs can be simultaneously sequenced by non-decreasing γ_j and by non-increasing $w_j\gamma_j\mu_j$, then such a sequence is optimal for the IES problem. For the IBS problem, the additional condition that the weights w_j are non-increasing is needed. None of these assumptions can be relaxed.

In Problem 2, if jobs can be simultaneously sequenced by non-decreasing stochastic order of their processing times and by non-increasing order of their weights, then such a sequence is optimal. Suppose that X and Y are two stochastic variables, X is said to be stochastically smaller than Y ($X \leq_{st} Y$) if and only if $\mathbb{P}(X > t) \leq \mathbb{P}(Y > t)$ for all t. However, such a sequence does not necessarily exist, because two c.d.f. cannot always be compared with regard to stochastic order. The same scheduling rule is also optimal for the problem with IES.

In Problem 3 there are only two homogeneous classes of jobs in the system (n_1 jobs of class 1, n_2 jobs of class 2). For the IBS problem, we show that the optimal policy is a threshold policy and we derive a closed form formula for the optimal threshold. Below this threshold, priority is given to one of the classes and above this threshold, priority is given to the other class. A threshold policy is also optimal for the problem with IBS.

In Problem 4, the optimal IES schedule is to process jobs in non-increasing order of

$\beta_j = w_j/(1/\mathcal{L}\{f_{X_j}\}(\gamma) - 1)$ and in non-increasing order of $\beta'_j = w_j/(1 - \mathcal{L}\{f_{X_j}\}(\gamma))$ for the IBS problem, where $\mathcal{L}\{f\}(\gamma) = \int_{t=0}^{+\infty} f(t)\exp(-\gamma t)dt$ is the Laplace transform of f in γ.

For Problem 5, IBS and IES can be reduced one to the other and the same schedule is optimal in both cases.

In the next section, we detail the proofs of the results for Problem 1 to Problem 5 with IBS and for Problem 1 with IES. We do not provide proofs of the additional results for the IES problems, since the techniques that are used are similar, see the results of Salch et al. (2011) for details.

3.5 Optimal static priority rules

First we provide a property which gives an analytic formula to compute the cost of a schedule for both kinds of impatience when processing times and due dates are exponentially distributed.

Property 1. *Consider a scheduling problem with n jobs. Job j has a weight w_j, an arbitrarily distributed processing time $X_j \sim F_{X_j}$ and an exponentially distributed due date with mean $1/\gamma_j$. Then the expected weighted number of late jobs for schedule $S = \{1, 2, \dots n\}$ is*

$$C(S) = \sum_{j=1}^{n} w_j \left(1 - \prod_{k=1}^{j} \mathcal{L}\{f_{X_k}\}(\gamma_j)\right) \text{ for IES problems and}$$
$$\tilde{C}(S) = \sum_{j=2}^{n} w_j \left(1 - \prod_{k=1}^{j-1} \mathcal{L}\{f_{X_k}\}(\gamma_j)\right) \text{ for IBS problems.}$$

Proof. Since there is no idle time on the machine and jobs are performed even if they are tardy, the value of the objective function of schedule S is

$$C(S) = \sum_{j=1}^{n} w_j \mathbb{P}(C_j > D_j) \text{ for IES problems and} \tag{3.1}$$
$$\tilde{C}(S) = \sum_{j=2}^{n} w_j \mathbb{P}(S_j > D_j) \text{ for IBS problems.} \tag{3.2}$$

From Equation (3.1),

$$\begin{aligned} C(S) &= \sum_{j=1}^{n} w_j \int_{t=0}^{+\infty} (1 - F_{C_j}(t)) f_{D_j}(t) dt \\ &= \sum_{j=1}^{n} w_j \gamma_j \int_{t=0}^{+\infty} (1 - F_{C_j}(t)) e^{-\gamma_j t} dt. \\ &= \sum_{j=1}^{n} w_j \gamma_j \mathcal{L}\{1 - F_{C_j}\}(\gamma_j) \\ &= \sum_{j=1}^{n} w_j \left(1 - \prod_{k=1}^{j} \mathcal{L}\{f_{X_k}\}(\gamma_j)\right) \end{aligned}$$

For these simplifications, we used the derivation property ($\mathcal{L}\{f\}(\gamma) = \gamma\mathcal{L}\{F\}(\gamma) - f(0)^-$) and the convolution property ($\mathcal{L}\{f_{X_i} * f_{X_j}\}(\gamma) = \mathcal{L}\{f_{X_i}\}(\gamma)\mathcal{L}\{f_{X_j}\}(\gamma)$) of the Laplace transforms (e.g. Kuhfittig 1978 or Appendix A.3).

Adapting this result for problems with IBS is trivial, up to some modifications in the indices of sums and products. This ends the proof of Property 1. □

From this property, we can derive the following corollary using $\mathcal{L}\{\mu \exp(-\mu t)\}(\gamma) = \mu/(\mu + \gamma)$, the Laplace transform of an exponential p.d.f.

Corollary 2. *Consider a scheduling problem with n jobs. Job j has a weight w_j, an exponentially distributed processing time with mean $1/\mu_j$ and an exponentially distributed due date with mean $1/\gamma_j$. Then the expected weighted number of late jobs for schedule $S = \{1, 2, \ldots n\}$ is*

$$C(S) = \sum_{j=1}^{n} w_j \left(1 - \prod_{k=1}^{j} \frac{\mu_k}{\mu_k + \gamma_j}\right) \text{ for IES problems and}$$

$$\tilde{C}(S) = \sum_{j=2}^{n} w_j \left(1 - \prod_{k=1}^{j-1} \frac{\mu_k}{\mu_k + \gamma_j}\right) \text{ for IBS problems.}$$

Preliminaries

In order to prove the theorems, we use a pairwise interchange argument between two adjacent jobs. The two schedules

$$\begin{aligned} S &: \{1, 2, \ldots s, i, u, s+3, \ldots n\} \quad \text{and} \\ S' &: \{1, 2, \ldots s, u, i, s+3, \ldots n\} \end{aligned}$$

differ only by the two jobs i and u which are swapped. These two jobs are in position $s+1$ and $s+2$, depending on the schedule under consideration. Since all jobs are processed, the

end of execution of job s and the beginning of execution of job $s+3$, occur at the same time in both schedules.

IES case

The starting time of job i in schedule S, denoted by $Z = \sum_{j=1}^{s} X_j$, is equal to the starting time of job u in schedule S'. Then, from Equation (3.1),

$$\begin{aligned} C(S) = &\sum_{j=1}^{s} w_j \mathbb{P}(C_j > D_j) + \sum_{j=s+3}^{n} w_j \mathbb{P}(C_j > D_j) \\ &+ w_i \mathbb{P}(Z + X_i > D_i) + w_u \mathbb{P}(Z + X_i + X_u > D_u). \end{aligned}$$

The same separation can be made on the cost of schedule S' and making the difference between these costs gives

$$\begin{aligned} &C(S') - C(S) \\ &= w_u \mathbb{P}(Z + X_u > D_u) + w_i \mathbb{P}(Z + X_u + X_i > D_i) \\ &\quad - w_i \mathbb{P}(Z + X_i > D_i) - w_u \mathbb{P}(Z + X_i + X_u > D_u). \end{aligned} \tag{3.3}$$

Schedule S performs better than schedule S' if and only if this difference is positive. However, Boxma and Forst (1986) state that this formula "is too general to allow useful comments", and that further assumptions are required. They choose to use independent and identically distributed processing times and/or due dates, to drastically simplify Equation (3.3). We do not assume i.i.d. variables, and thus require the two new lemmas below.

When considering exponential due dates, we can further simplify Equation (3.3).

Lemma 1. *When $D_j \sim \exp(\gamma_j)$ and $X_j \sim F_{X_j}$, we have*

$$\begin{aligned} &C(S') - C(S) \\ &= w_i \mathbb{P}(Z + X_i \leq D_i) \mathbb{P}(X_u > D_i) \\ &\quad - w_u \mathbb{P}(Z + X_u \leq D_u) \mathbb{P}(X_i > D_u) \\ &= w_i \mathcal{L}\{f_Z\}(\gamma_i) \mathcal{L}\{f_{X_i}\}(\gamma_i) \left[1 - \mathcal{L}\{f_{X_u}\}(\gamma_i)\right] \\ &\quad - w_u \mathcal{L}\{f_Z\}(\gamma_u) \mathcal{L}\{f_{X_u}\}(\gamma_u) \left[1 - \mathcal{L}\{f_{X_i}\}(\gamma_u)\right] \end{aligned}$$

Proof. We have

$$\begin{aligned} &\mathbb{P}(Z + X_u + X_i > D_i) \\ &= \mathbb{P}(Z + X_u + X_i > D_i \mid Z + X_i > D_i) \mathbb{P}(Z + X_i > D_i) \\ &\quad + \mathbb{P}(Z + X_u + X_i > D_i \mid Z + X_i \leq D_i) \mathbb{P}(Z + X_i \leq D_i) \end{aligned}$$

The memoryless property of D_i implies that

$$\mathbb{P}(Z + X_u + X_i > D_i \mid Z + X_i \leq D_i) = \mathbb{P}(X_u > D_i).$$

Consequently,

$$\begin{aligned}\mathbb{P}(Z + X_u + X_i > D_i) =& \mathbb{P}(Z + X_i > D_i) \\ &+ \mathbb{P}(X_u > D_i)\mathbb{P}(Z + X_i \leq D_i).\end{aligned}$$

Applying this formula to $\mathbb{P}(Z + X_u + X_i > D_i)$ and $\mathbb{P}(Z + X_i + X_u > D_u)$ in Equation (3.3) explains the first equality of Lemma 1. If D is exponentially distributed with mean $1/\gamma$ then $\mathbb{P}(Z \leq D) = \mathcal{L}\{f_Z\}(\gamma)$, see Appendix A.3.4, which results in the second equality of the lemma using the convolution property of the Laplace transform. This ends the proof. □

IBS case

From Equation (3.2), the difference between the costs of schedules S and S' is

$$\begin{aligned}\tilde{C}(S') - \tilde{C}(S) =& w_u\mathbb{P}(Z > D_u) + w_i\mathbb{P}(Z + X_u > D_i) \\ &- w_i\mathbb{P}(Z > D_i) - w_u\mathbb{P}(Z + X_i > D_u).\end{aligned} \tag{3.4}$$

Again, considering exponential due dates leads to a simpler formula:

Lemma 2. *When $D_j \sim \exp(\gamma_j)$ and $X_j \sim F_{X_j}$, we have*

$$\begin{aligned}\tilde{C}(S') - \tilde{C}(S) =& w_i\mathbb{P}(Z \leq D_i)\mathbb{P}(X_u > D_i) \\ &- w_u\mathbb{P}(Z \leq D_u)\mathbb{P}(X_i > D_u) \\ =& w_i\mathcal{L}\{f_Z\}(\gamma_i)\left[1 - \mathcal{L}\{f_{X_u}\}(\gamma_i)\right] \\ &- w_u\mathcal{L}\{f_Z\}(\gamma_u)\left[1 - \mathcal{L}\{f_{X_i}\}(\gamma_u)\right].\end{aligned}$$

Proof. Similar to that of Lemma 1. □

3.5.1 Problem 1: $X_j \sim \exp(\mu_j)$, $D_j \sim \exp(\gamma_j)$

Theorem 6. *Assume Job j has a weight w_j, an exponentially distributed processing time with mean $1/\mu_j$ and an exponentially distributed due date with mean $1/\gamma_j$.*

If jobs can be ordered such that

$$I: \begin{cases} i) & \gamma_1 \leq \gamma_2 \leq \cdots \leq \gamma_n \text{ and} \\ ii) & w_1\gamma_1\mu_1 \geq w_2\gamma_2\mu_2 \geq \cdots \geq w_n\gamma_n\mu_n \end{cases}$$

then scheduling jobs in non-decreasing order of the indices is optimal for problems with IES.

If jobs can be ordered such that

$$II: \begin{cases} i) & \gamma_1 \leq \gamma_2 \leq \cdots \leq \gamma_n, \\ ii) & w_1\gamma_1\mu_1 \geq w_2\gamma_2\mu_2 \geq \cdots \geq w_n\gamma_n\mu_n \text{ and} \\ iii) & w_1 \geq w_2 \geq \cdots \geq w_n, \end{cases}$$

then scheduling jobs in non-decreasing order of the indices is optimal for problems with IBS.

Proof.
IES case

From Lemma 1, when all distributions are exponential, we have $C(S') - C(S) \geq 0$ if and only if

$$\frac{w_u\gamma_u}{\mu_i+\gamma_u}\frac{\mu_u}{\mu_u+\gamma_u}\mathcal{L}\{f_Z\}(\gamma_u) \leq \frac{w_i\gamma_i}{\mu_u+\gamma_i}\frac{\mu_i}{\mu_i+\gamma_i}\mathcal{L}\{f_Z\}(\gamma_i). \tag{3.5}$$

As the Laplace transform is a decreasing function, $\gamma_i \leq \gamma_u$ implies that $\mathcal{L}\{f_Z\}(\gamma_i) \geq \mathcal{L}\{f_Z\}(\gamma_u)$.

Moreover, $i)$ also implies that $(\mu_i+\gamma_u)(\mu_u+\gamma_u) \geq (\mu_u+\gamma_i)(\mu_i+\gamma_i)$ and then

$$\frac{w_u\gamma_u}{\mu_i+\gamma_u}\frac{\mu_u}{\mu_u+\gamma_u} \leq \frac{w_i\gamma_i}{\mu_u+\gamma_i}\frac{\mu_i}{\mu_i+\gamma_i} \text{ because of } ii).$$

Hence, with the set of assumptions I of Theorem 6, Inequality (3.5) holds for every pair of jobs and therefore it is optimal to schedule jobs in non-decreasing order of their indices. This ends the proof of Theorem 6 for IES.

IBS case

From Lemma 2, when all distributions are exponential, we have $\tilde{C}(S') - \tilde{C}(S) \geq 0$ iff

$$\frac{w_u\gamma_u}{\mu_i+\gamma_u}\mathcal{L}\{f_Z\}(\gamma_u) \leq \frac{w_i\gamma_i}{\mu_u+\gamma_i}\mathcal{L}\{f_Z\}(\gamma_i). \tag{3.6}$$

As before, with the set of assumptions II of Theorem 6, Inequality (3.6) holds for every pair of jobs and therefore it is optimal to schedule jobs in non-decreasing order of their indices. This ends the proof of Theorem 6 for IBS. □

Compared to the IES case, the IBS case requires the extra condition that the weights are non-increasing. To understand why, consider an instance with two jobs and equal due date distributions ($\gamma_1 = \gamma_2 = \gamma$). When γ is very large, the two jobs have very little chance to be completed on time. In the IBS problem, priority should be given to the job with

Relaxed condition	Counter-examples for IES			
		w_i	μ_i	γ_i
i) $\gamma_j \nearrow$	Job 1	1	1	2
	Job 2	1	1	1
ii) $w_j\gamma_j\mu_j \searrow$	Job 1	1	1	1
	Job 2	1	2	2

Relaxed condition	Counter-examples for IBS			
		w_i	μ_i	γ_i
i) $\gamma_j \nearrow$	Job 1	1.5	0.5	2.5
	Job 2	1	0.5	3
	Job 3	0.5	0.5	0.5
ii) $w_j\gamma_j\mu_j \searrow$	Job 1	2	1	1
	Job 2	1	2	3
iii) $w_j \searrow$	Job 1	1	3	3
	Job 2	2	1	4

Table 6: Counter-examples when one condition is relaxed in Theorem 6

the largest weight, in order to minimize the short term cost (no cost is paid for the job scheduled in first). This is not the case in the IES problem.

None of the assumptions of Theorem 6 can be relaxed. Table 6 provides counter-examples when one condition is relaxed. For example, the second row of Table 6 indicates that if condition i) is relaxed, then the instance described in the second column is a counter-example. Corollary 2 has been used to compute the costs.

3.5.2 Problem 2: $X_j \sim F_{X_j}, D_j \sim F_D$

Theorem 7. *Assume that the processing times are independent ($X_j \sim F_{X_j}$) and that the due dates are i.i.d. ($D_j \sim F_D$). If jobs can be simultaneously sequenced 1) by non-decreasing stochastic order of their processing times and 2) by non-increasing order of their weights, then such a sequence is optimal for the IBS problem.*

Proof. Given that the due dates are i.i.d. ($D_j \sim F_D$) and using cumulative and density probability functions, Equation (3.4) can be simplified to

$$\begin{aligned}&\tilde{C}(S') - \tilde{C}(S)\\ &= \int_{t=0}^{+\infty} \big[(w_i - w_u)F_Z(t) - w_i F_{Z+X_u}(t)\\ &\qquad\qquad + w_u F_{Z+X_i}(t)\big] f_D(t)dt.\end{aligned}$$

If $X_i \leq_{st} X_u$, then $Z + X_i \leq_{st} Z + X_u$ (c.f. Appendix A.2.2) and

$$\begin{aligned}&\tilde{C}(S') - \tilde{C}(S)\\ &\geq \int_{t=0}^{+\infty} \big[(w_i - w_u)(F_Z(t) - F_{Z+X_u}(t))\big] f_D(t)dt.\end{aligned}$$

Given that all possible values for X_u are non-negative, we have the following inequality: $F_{Z+X_u}(t) = \mathbb{P}(Z + X_u \leq t) \leq \mathbb{P}(X_u \leq t) = F_{X_u}(t)$ for all $t \geq 0$. Then, all the terms in the integral are positive. This ends the proof of Theorem 7. □

3.5.3 Problem 3: $X_j \sim F_X, D_j \sim \exp(\gamma_j)$, 2 classes of jobs

Theorem 8. *Let the processing times be i.i.d. and follow a general distribution function F_X. Assume that there are n_1 jobs from class 1 (with weight w_1 and impatience rate γ_1) and n_2 jobs from class 2 (with weight w_2 and impatience rate γ_2). The total number of jobs is $n = n_1 + n_2$. Let $\alpha_j = \mathcal{L}\{f_X\}(\gamma_j)$ for $j \in \{1, 2\}$ and assume without loss of generality that $\alpha_1 \geq \alpha_2$. Then, it is optimal to give priority to class 1 in any position s such that*

$$s \geq t_{IBS} = \ln\left(\frac{w_2(1-\alpha_2)}{w_1(1-\alpha_1)}\right)\left(\ln\left(\frac{\alpha_1}{\alpha_2}\right)\right)^{-1}.$$

Proof. Starting again from Lemma 2 with i.i.d. processing times

$$\begin{aligned}&\tilde{C}(S') - \tilde{C}(S)\\ &= w_i \left[\mathcal{L}\{f_X\}(\gamma_i)\right]^s \left(1 - \mathcal{L}\{f_X\}(\gamma_i)\right)\\ &\quad - w_u \left[\mathcal{L}\{f_X\}(\gamma_u)\right]^s \left(1 - \mathcal{L}\{f_X\}(\gamma_u)\right).\end{aligned} \tag{3.7}$$

This inequality describes that it is preferable to process job i before job u when looking for which job to process in position $s + 1$. If it is always preferable to process job i before

job u, i.e. the inequality is valid for all s, then i should be processed before u in an optimal schedule.

When there are only two classes of jobs in the system, the quantity $(\tilde{C}(S') - \tilde{C}(S))$ is non-negative in Equation (3.7) if $w_1\alpha_1^s(1-\alpha_1) \geq w_2\alpha_2^s(1-\alpha_2)$.

When $\alpha_1 = \alpha_2$, $(\tilde{C}(S') - \tilde{C}(S))$ is non-negative if $w_1 \geq w_2$, independently of position s. It is therefore optimal to always give priority to class 1 when $w_1 \geq w_2$ and $\alpha_1 = \alpha_2$.

When $\alpha_1 > \alpha_2$, it is optimal to give priority to class 1 when

$$s \geq t_{IBS} = \ln\left(\frac{w_2(1-\alpha_2)}{w_1(1-\alpha_1)}\right)\left(\ln\left(\frac{\alpha_1}{\alpha_2}\right)\right)^{-1}.$$

Hence it is optimal to give priority to class 1 in position s when $s \geq t_{IBS}$. If $t_{IBS} \leq 0$, priority is always given to class 1. This occurs for example when w_1 is very large. If $t_{IBS} > n$, it means that priority is always given to class 2. This occurs for example when w_2 is very large. □

The same proof can be used to prove the optimality of a threshold policy for the problem with IES. In this case it is optimal to give priority to class 1 in position s when $s \geq t_{IES} = t_{IBS} - 1$. (See Salch et al. 2011 for a proof.)

3.5.4 Problem 4: $X_j \sim F_{X_j}, D_j \sim \exp(\gamma)$

Theorem 9. *Let the processing times be independent random variables ($X_j \sim F_j$) and the due dates be i.i.d., exponentially distributed with mean $1/\gamma$. An optimal IBS schedule is to process jobs in non-increasing order of $\beta'_j = w_j/(1 - \mathcal{L}\{f_{X_j}\}(\gamma))$.*

Proof. Starting from Lemma 2 with i.i.d. and exponential due dates, $\tilde{C}(S') - \tilde{C}(S) \geq 0$ iff $w_i[1 - \mathcal{L}\{f_{X_u}\}(\gamma) \geq w_u[1 - \mathcal{L}\{f_{X_i}\}(\gamma)]$, which ends the proof of the theorem. □

3.5.5 Problem 5: $X_j \sim F_X$, $D_j \sim F_D$

Theorem 10. *Let the processing times be i.i.d. random variables ($X_j \sim F_X$) and the due dates be also i.i.d. random variables ($D_j \sim F_D$). An optimal IBS static list scheduling policy is to process the jobs in non-increasing order of their weights.*

Proof. The optimal IES schedule is to process the jobs in non-increasing order of their weights (Boxma and Forst 1986). Here one can modify the due dates from the IBS problem to $D'_j = D_j + X_j$ resulting in an instance of an IES problem. The due dates are still i.i.d. and since the IES policy does not depend on the probability distributions of the due dates, the same optimal scheduling rule holds for the IBS problem. □

Chapitre 4

Dynamic priority rules for stochastic scheduling with abandonments

A preliminary version of this chapter was presented in the *ILS 2012* conference (Salch et al. 2012b).

4.1 Introduction

In several service or production systems, customers or tasks may become impatient and leave the system, even if they have not been served. This behavior is called an abandonment, which generate a cost of insatisfaction or a loss of benefit for the system. Typical examples are call centers, where the quality of service is clearly increasing with the ability to answer a phone call quickly. The performance of such systems can be measured by associating a cost to each abandonment of a customer, e.g. a cost for each hang up, depending on which class the customer belongs to. The objective of the system manager is then to sequence demands so that the overall cost is minimized. For additional information on consequences of abandonments, one can read Artalejo and Pla (2009) who study the case of telecommunication systems.

In this chapter we address the problem of scheduling jobs with stochastic processing times, stochastic release dates and stochastic patience times, in order to minimize the expected abandonment and holding costs.

In problems with abandonments, we distinguish two types of impatience : impatience to the end of service (IES) and impatience to the beginning of service (IBS). In IES situations, even if a job is already in service, it still may abandon. The job is served on time if the service ended before it became impatient; this is typically the situation of production systems, when manufacturing must be achieved for a product to be available. In IBS situations, when a job is in service, it cannot become impatient anymore. A job may only abandon before its service has begun; this is the case of call centers, when customers are considered to be served on time if they have not abandoned before they reach an operator.

In this chapter, we consider both IES and IBS situations.

The remainder of the chapter is organized as follows. In Section 4.2 we present a literature review on problems with impatience and abandonments. In Section 4.3 we present the model of the problem under consideration and some notations. In Section 4.4 we provide conditions under which a strict priority rule is optimal for both IES and IBS problems. By strict priority we mean that classes of jobs can be sorted according to a priority index. The class of jobs of lowest index being the class of highest priority, and a job from class j is processed only if there are no jobs from any classes j' such that $j' < j$. We detail also some additional results and discuss the limits of the main rule. Section 4.5 investigates some extensions. In Section 4.6 we analyze the relevance of the rule and the performance of several heuristics through a numerical study on the problem. Finally, we present our concluding remarks in Section 4.7.

4.2 Literature review

In the literature dedicated to queueing systems with abandonments, authors mainly consider a single class of customers (e.g. Mandelbaum and Zeltyn 2009, Movaghar 1998, 2006)). In particular, Boxma et al. (2011) derive new results for the $M/G/1+G$ queue, for both types of impatience (IES and IBS), and provide an extensive literature review on queueing with abandonments. However Iravani and Balcioglu (2008) derive some performance measures for a queueing system with several classes of customers.

The literature is much less extensive when considering the problem of scheduling several types of jobs with abandonments. Such a problem can be treated either as a scheduling problem with stochastic due dates or as a queueing control problem with abandonments. A stochastic due date can be interpreted as a customer getting impatient at some random time.

We first review the stochastic scheduling literature with abandonments. In the following papers, all jobs are available at time 0. Pinedo (1983) studies a stochastic scheduling problem with IES on a single machine with the objective to minimize the expected abandonment costs, in the class of static list scheduling policies. A list is built at time zero and never changed thereafter: if a job reaches its due date, it is processed anyway. This case describes impatience rather than a true abandonment. For the particular case where processing times follow independent exponential distributions and due dates follow general distribution function, he proves that serving customers in non-increasing order of the ratio of their abandonment cost times their mean processing time, the so-called "$c\mu$ rule", is optimal. Results from Pinedo are extended by Boxma and Forst (1986) for other probability distributions. Salch et al. (2013) generalize these results and study the IBS counterpart of these problems.

Argon et al. (2008) study a similar problem with IBS except that there are true abandonments: once the deadline of a job is reached, it cannot be processed afterwards. In

other words, when customers get impatient, they quit the system and cannot be served anymore. The objective is to minimize the expected number of abandonments in the class of dynamic policies (i.e. decisions can be taken at any time). With only two classes of customers, they give several sufficient conditions under which a strict priority rule is optimal. They also propose two heuristics based on their observations of the optimal policy and on a stochastic dynamic programming formulation of the problem.

In the literature on optimal control of queues, Panwar et al. (1988) characterize an optimal policy for a $M/G/1+G$ queue with IBS when all durations are known at the arrival of jobs. The objective is to minimize the fraction of late jobs. Down et al. (2011) consider a $M/M/1+M$ queueing problem with two classes of customers, IES and preemption. An abandonment cost is incurred when a customer abandons the system. Processing times are identically distributed for both classes of customers. They provide sufficient conditions for a strict priority rule to be optimal. Atar et al. (2010) studies a variant of this problem with n classes of customers. They also relax the assumption of identically distributed processing times. Let c_j, μ_j and γ_j be respectively the waiting cost per unit of time, service rate and impatience rate of a class j customer. They show that scheduling jobs with the highest index $c_j\mu_j/\gamma_j$ is asymptotically optimal, under overload conditions and many-server fluid scaling. They also prove that in a markovian system, considering a waiting cost per unit of time c_j or an abandonment cost w_j is equivalent by letting $c_j = w_j\gamma_j$. We extend this result for non-markovian systems, when only the patience time is assumed to be exponentially distributed.

This chapter is closely related to the results of Down et al. (2011) and Atar et al. (2010). We extend the results of Down et al. (2011) in several directions. First, we consider an arbitrary number of classes of customers instead of two classes. Then, the processing times are not necessarily identically distributed. Finally, the arrival processes can be arbitrarily distributed. Our results are consistent with the asymptotically optimal policy of Atar et al. (2010).

4.3 Problem description and notations

We consider a stochastic scheduling problem where n jobs have to be processed on a single machine. Job j has a stochastic processing time X_j, a stochastic release date R_j, a stochastic patience time D_j, a holding cost c_j and an abandonment cost w_j. We assume that in problems with IES, holding costs are paid while the job is in the system, and that in problems with IBS, holding costs are paid while the job waits for service (but are not paid during the service). We assume that patience time D_j is exponentially distributed with rate γ_j. Processing times and release dates are independent random variables and can be arbitrarily distributed.

Let C_j be the random time when Job j leaves the system (due to abandonment or end of execution) and S_j be the random time when Job j begins its service. Let U_j be equal

to 1 if $D_j < C_j$, and equal to 0 otherwise. U_j indicates if Job j is late in the sense of IES. Let $\tilde{U}_j$ be equal to 1 if $D_j < S_j$, and equal to 0 otherwise. $\tilde{U}_j$ indicates if Job j is late in the sense of IBS.

Our objective is to minimize the expected abandonment costs, either in the class of static list scheduling policies or in the class of dynamic policies with preemption. In the class of static list scheduling policies, a priority list is built at time 0 and jobs are processed in the predefined order: the list is not modified thereafter. As jobs leave the system when they become impatient, jobs can be present in this list but not in the system anymore. In such a case, the next job on the list which is still available is processed. In the class of dynamic policies, decisions can be made at any point in time to choose the next job to be processed.

The expected cost of a schedule S is then

$$C(S) = \sum_j c_j E(C_j) + \sum_j w_j E(U_j) \qquad \text{for IES and}$$
$$\tilde{C}(S) = \sum_j c_j E(S_j) + \sum_j w_j E(\tilde{U}_j) \qquad \text{for IBS.}$$

Atar et al. (2010) prove that holding costs and abandonment costs are equivalent and can be treated in a unified manner when all durations are exponentially distributed. In Appendix C.2, we show that this is also the case when only the patience times are exponentially distributed. The expected cost of a schedule S can be rewritten as

$$C(S) = \sum_j \left(w_j + \frac{c_j}{\gamma_j} \right) E(U_j) = \sum_j c'_j E(C_j) \qquad \text{for IES and}$$
$$\tilde{C}(S) = \sum_j \left(w_j + \frac{c_j}{\gamma_j} \right) E(\tilde{U}_j) = \sum_j c'_j E(S_j) \qquad \text{for IBS.}$$

An abandonment cost w_j is equivalent to a holding cost $c_j = w_j \gamma_j$. In the rest of the chapter we will consider, without loss of generality, only abandonment costs ($w_j > 0, c_j = 0, \forall j$). We will also assume that processing times are exponentially distributed (with rate μ_j for Job j).

4.4 Strict priority rules

In this section we assume that processing times are exponentially distributed with rate μ_j and that patience time are exponentially distributed as well with rate γ_j.

4.4.1 Two jobs

In this section, only static priority rules with abandonments are discussed and the two jobs under consideration are available at time $t = 0$: $R_1 = R_2 = 0$. All decisions are taken

at time zero. The following lemmas give sufficient conditions under which a strict priority rule is optimal and conditions under which a schedule empties the system the fastest.

Property 2. *The cost of the static schedule* $\{1,2\}$ *with abandonments and IES is*

$$C(\{1,2\}) = w_1\mathbb{P}(X_1 \geq D_1) + w_2[1 - \mathbb{P}(C_1 \leq D_2)\mathbb{P}(X_2 \leq D_2)]$$

Proof. For Job 1, we have $C_1 = \min(X_1, D_1)$ which leads to

$$\begin{aligned} C(\{1,2\}) &= w_1\mathbb{P}(X_1 \geq D_1) + w_2\mathbb{P}(C_1 + X_2 \geq D_2) \\ &= w_1\mathbb{P}(X_1 \geq D_1) \\ &\quad + w_2\mathbb{P}(C_1 + X_2 \geq D_2 \mid C_1 \geq D_2)\mathbb{P}(C_1 \geq D_2) \\ &\quad + w_2\mathbb{P}(C_1 + X_2 \geq D_2 \mid C_1 \leq D_2)\mathbb{P}(C_1 \leq D_2) \\ &= w_1\mathbb{P}(X_1 \geq D_1) + w_2\mathbb{P}(C_1 \geq D_2) \\ &\quad + w_2\mathbb{P}(C_1 + X_2 \geq D_2 \mid C_1 \leq D_2)\mathbb{P}(C_1 \leq D_2). \end{aligned}$$

Property 7 (page 131) is then used to simplify this expression to

$$\begin{aligned} C(\{1,2\}) &= w_1\mathbb{P}(X_1 \geq D_1) + w_2\mathbb{P}(C_1 \geq D_2) \\ &\quad + w_2\mathbb{P}(X_2 \geq D_2)\mathbb{P}(C_1 \leq D_2) \\ &= w_1\mathbb{P}(X_1 \geq D_1) \\ &\quad + w_2[1 - \mathbb{P}(C_1 \leq D_2)\mathbb{P}(X_2 \leq D_2)] \end{aligned}$$

Which ends the proof of Property 2. □

Lemma 3. *For* $n = 2$ *jobs, it is optimal to schedule Job* 1 *first in the class of static list scheduling policies with IES if and only if*

$$\frac{w_1\gamma_1\mu_1}{(\mu_2 + \gamma_1 + \gamma_2)(\mu_1 + \gamma_1)} \geq \frac{w_2\gamma_2\mu_2}{(\mu_1 + \gamma_1 + \gamma_2)(\mu_2 + \gamma_2)}.$$

Proof. We compare the costs of the two possible schedules: $C(\{1,2\})$, the expected abandonment costs when processing Job 1 first and then Job 2, and $C(\{2,1\})$, when processing Job 2 first and then Job 1. From Property 2, if all durations are exponentially distributed, it leads to the expression

$$\begin{aligned} C(\{1,2\}) &= \frac{w_1\gamma_1}{\mu_1 + \gamma_1} + w_2\left[1 - \frac{\mu_1 + \gamma_1}{\mu_1 + \gamma_1 + \gamma_2}\frac{\mu_2}{\mu_2 + \gamma_2}\right] \\ &= \frac{w_1\gamma_1}{\mu_1 + \gamma_1} + w_2\gamma_2\left[\frac{\mu_1 + \gamma_1 + \mu_2 + \gamma_2}{(\mu_1 + \gamma_1 + \gamma_2)(\mu_2 + \gamma_2)}\right] \\ &= \frac{w_1\gamma_1}{\mu_1 + \gamma_1} + \frac{w_2\gamma_2}{\mu_2 + \gamma_2} + \frac{w_2\gamma_2\mu_2}{(\mu_1 + \gamma_1 + \gamma_2)(\mu_2 + \gamma_2)}. \end{aligned}$$

Likewise,

$$C(\{2,1\}) = \frac{w_1\gamma_1}{\mu_1+\gamma_1} + \frac{w_2\gamma_2}{\mu_2+\gamma_2} + \frac{w_1\gamma_1\mu_1}{(\mu_2+\gamma_1+\gamma_2)(\mu_1+\gamma_1)}$$

and thus

$$\begin{aligned} C(\{2,1\}) - C(\{1,2\}) = &\frac{w_1\gamma_1\mu_1}{(\mu_2+\gamma_1+\gamma_2)(\mu_1+\gamma_1)} \\ &- \frac{w_2\gamma_2\mu_2}{(\mu_1+\gamma_1+\gamma_2)(\mu_2+\gamma_2)}. \end{aligned} \tag{4.1}$$

This ends the proof of Lemma 3. □

With a slight loss of generality, Lemma 3 leads to this simpler corollary.

Corollary 3. *For $n = 2$ jobs, if 1) $\mu_1 \geq \mu_2$, 2) $\gamma_1 \leq \gamma_2$ and 3) $w_1\gamma_1 \geq w_2\gamma_2$, then it is optimal to schedule Job 1 first in the class of static list scheduling policies with IES.*

Proof. Equation (4.1) is equivalent to

$$\begin{aligned} C(\{2,1\}) - C(\{1,2\}) = &\frac{w_1\gamma_1}{(\mu_2+\gamma_1+\gamma_2)\left(1+\frac{\gamma_1}{\mu_1}\right)} \\ &- \frac{w_2\gamma_2}{(\mu_1+\gamma_1+\gamma_2)\left(1+\frac{\gamma_2}{\mu_2}\right)}. \end{aligned}$$

Hence, given the assumptions of Corollary 3, $C(\{2,1\}) \geq C(\{1,2\})$. □

The same lemma can be derived for problems with IBS. As the proof has the same outline, it is omitted.

Lemma 4. *For $n = 2$ jobs, it is optimal to schedule Job 1 first in the class of static list scheduling policies with IBS if and only if*

$$\frac{w_1\gamma_1}{\mu_2+\gamma_1} \geq \frac{w_2\gamma_2}{\mu_1+\gamma_2}.$$

Proof. Same as for Lemma 3. □

The argument used to prove Lemma 3 is a pairwise interchange argument. When $n > 2$, a pairwise interchange can postpone the remaining jobs in the schedule, so that the contribution of the two first jobs to the objective function decreases, but the overall cost increases. An easy way to prevent the interchange from postponing the remaining jobs is to consider conditions which 1) improve the contribution of the two jobs from the interchange and 2) bring forward the remaining jobs.

Forcing the jobs from the interchange to end earlier is a sufficient condition to decrease the cost generated by the other jobs in the schedule. Indeed during the amount of time

which is saved by the interchange, the machine can process another job, which in turn can only quit the system earlier and so on.

Let us introduce $T_{\{1,2\}}$, respectively $T_{\{2,1\}}$, the time needed to process Job 1 then Job 2, respectively Job 2 then Job 1.

Property 3. *The expected time needed to clear the system for schedule* $\{1,2\}$ *in the class of static scheduling policies with abandonment and IES is*

$$E(T_{\{1,2\}}) = E(C_1) + E(\min(X_2, D_2))\mathbb{P}(C_1 \leq D_2).$$

Proof. When Job 1 is processed first, the expected time needed to process the two jobs is

$$E(T_{\{1,2\}}) = E(C_1) + E[\min(X_2, D_2 - C_1) \mid C_1 \leq D_2]\mathbb{P}(C_1 \leq D_2).$$

Property 8 (page 131) leads to

$$E(T_{\{1,2\}}) = E(C_1) + E(\min(X_2, D_2))\mathbb{P}(C_1 \leq D_2).$$

□

The following lemma gives conditions under which giving priority to Job 1 minimizes the mean time needed to process the two jobs.

Lemma 5. *For* $n = 2$ *jobs,* $E(T_{\{1,2\}}) \leq E(T_{\{2,1\}})$ *if and only if*

$$\frac{\gamma_2}{(\mu_2 + \gamma_2)(\mu_1 + \gamma_1 + \gamma_2)} \geq \frac{\gamma_1}{(\mu_1 + \gamma_1)(\mu_2 + \gamma_1 + \gamma_2)}.$$

Proof. From Property 3, if all durations are exponentially distributed,

$$E(T_{\{1,2\}}) = \frac{1}{\mu_1 + \gamma_1} + \frac{1}{\mu_2 + \gamma_2}\frac{\mu_1 + \gamma_1}{\mu_1 + \gamma_1 + \gamma_2}.$$

The term $1/(\mu_1 + \gamma_1)$, respectively $1/(\mu_2 + \gamma_2)$, represents the time Job 1, respectively Job 2, spends in the system if it is processed right away. In the previous equality, the mean processing time of Job 2 is only taken into account when it does not abandon during the processing of Job 1.

Likewise, we have

$$E(T_{\{2,1\}}) = \frac{1}{\mu_2 + \gamma_2} + \frac{1}{\mu_1 + \gamma_1}\frac{\mu_2 + \gamma_2}{\mu_2 + \gamma_1 + \gamma_2}.$$

Now let us write the difference between $E(T_{\{2,1\}})$ and $E(T_{\{1,2\}})$,

$$\begin{aligned} E(T_{\{2,1\}}) - E(T_{\{1,2\}}) &= \frac{1}{\mu_1+\gamma_1}\left(\frac{\mu_2+\gamma_2}{\mu_2+\gamma_1+\gamma_2}-1\right) \\ &\quad + \frac{1}{\mu_2+\gamma_2}\left(1-\frac{\mu_1+\gamma_1}{\mu_1+\gamma_1+\gamma_2}\right) \\ &= \frac{\gamma_2}{(\mu_2+\gamma_2)(\mu_1+\gamma_1+\gamma_2)} \\ &\quad - \frac{\gamma_1}{(\mu_1+\gamma_1)(\mu_2+\gamma_1+\gamma_2)}. \end{aligned} \tag{4.2}$$

This ends the proof of Lemma 5. □

Again, with a slight loss of generality, Lemma 5 leads to this simpler property.

Corollary 4. *For $n = 2$ jobs, if 1) $\mu_1 \geq \mu_2$ and 2) $\gamma_1 \leq \gamma_2$, then $E(T_{\{1,2\}}) \leq E(T_{\{2,1\}})$.*

Proof. From Equation (4.2):

$$\begin{aligned} E(T_{\{2,1\}}) - E(T_{\{1,2\}}) &= \gamma_2(\mu_1+\gamma_1)(\mu_2+\gamma_1+\gamma_2) \\ &\quad - \gamma_1(\mu_2+\gamma_2)(\mu_1+\gamma_1+\gamma_2) \\ &= \mu_1\mu_2\gamma_2 + \mu_1\gamma_2^2 - \mu_1\mu_2\gamma_1 - \mu_2\gamma_1^2 \\ &= (\gamma_2-\gamma_1)\mu_1\mu_2 + (\mu_1-\mu_2)\gamma_1\gamma_2 \\ &= \mu_1\mu_2\gamma_1\gamma_2\left(\frac{1}{\gamma_1}-\frac{1}{\gamma_2}+\frac{1}{\mu_2}-\frac{1}{\mu_1}\right). \end{aligned}$$

With assumptions 1) and 2), this expression is clearly non-negative. As a consequence, processing Job 1 and then Job 2 is faster than the opposite. □

The same lemma can be derived for problems with IBS. As the proof has the same outline, it is omitted.

Lemma 6. *For $n = 2$ jobs, $E(\tilde{T}_{\{1,2\}}) \leq E(\tilde{T}_{\{2,1\}})$ if and only if*

$$\frac{1}{\mu_2}\frac{\gamma_2}{\mu_1+\gamma_2} \geq \frac{1}{\mu_1}\frac{\gamma_1}{\mu_2+\gamma_1}.$$

Proof. Same as for Lemma 5 □

Now that these sufficient conditions for optimality are identified, the theorem giving a static list priority rule can be introduced. The outline of the proof is inspired by Pinedo (2008).

4.4.2 Arbitrary number of jobs

In this section, we consider dynamic priority rules with an arbitrary number of jobs. The n jobs are not necessarily available at time $t = 0$.

Theorem 11. *Let us consider a scheduling problem with n jobs and IES. A Job j has an exponentially distributed processing time with mean $1/\mu_j$, an exponentially distributed patience time with mean $1/\gamma_j$, an abandonment cost w_j and an arbitrarily distributed release date. If Jobs can be ordered such that Lemma 3 and Lemma 5 hold for every pair (i, j) of jobs, with $i < j$, then scheduling jobs in non-decreasing order of the indices minimizes the expected costs of abandonments in the class of dynamic policies with preemption.*

Proof. The proof is divided in two parts, both using induction. First we prove that the priority rule given in Theorem 11 is an optimal policy among static scheduling policies, without arrivals ($R_j = 0$ for all j in $\{1, 2, \ldots, n\}$). This result is then extended to dynamic scheduling policies with preemption and arrivals.

Static policy. If there are two jobs left in the system, the optimal policy is to process the job with smallest index first, as claims Lemma 3.

If there are three jobs left, we assume without loss of generality that they are Job 1, 2 and 3. One knows for sure that the two last jobs must be processed in non-increasing order of their index. Therefore, the three possible schedules are $\{1, 2, 3\}$, $\{2, 1, 3\}$ and $\{3, 1, 2\}$. Let us focus on schedule $\{3, 1, 2\}$. We know from Lemma 3 that $\{3, 1\}$ is more expensive than $\{1, 3\}$ and from Lemma 5 that $\{3, 1\}$ ends later than $\{1, 3\}$. Consequently, schedule $\{3, 1, 2\}$ can not be optimal, because schedule $\{1, 3, 2\}$ performs better. By following the same reasoning on Jobs 1 and 2, we deduce that schedule $\{1, 2, 3\}$ performs better than schedule $\{2, 1, 3\}$. Thus schedule $\{1, 2, 3\}$ is optimal.

Now suppose that there are n jobs left and that when there are $n - 1$ jobs left, it is optimal to process jobs with smallest index first. Because patience times are memoryless, the amount of time spent waiting for the first job to leave the system is not taken into account for jobs still in the system. Let us assume that the first job to be processed does not have the smallest index among the other jobs present in the system. Then interchanging this job with the second job, which is by construction the one of smallest index, reduces the overall cost of the schedule. This interchange reduces the cost generated by the first two jobs (Lemma 3) and it reduces the cost generated by the other $n - 2$ jobs, because they can begin to be processed sooner (Lemma 5). Consequently, it is optimal to process the job of smallest index first in the class of static scheduling policies, without arrivals of jobs.

Dynamic policy. When the optimal policy is searched among the class of dynamic policies and when preemption is authorized, at any point in time one can decide which job has to be processed. However when all durations are exponential, the system at time t is completely described by which jobs are present in the system at that time. Therefore when a decision is taken, there is no need to revoke it until the next event, that is to say an abandonment, the end of a process or an arrival. While considering arrivals, the question of idling the machine comes up. It may be optimal to idle the machine, even if there are jobs waiting to be processed, if one knows that there is a higher chance that a very important job will soon show up. But this reasoning does not apply here, because

processing times are exponentially distributed and preemption is allowed. Hence, a job can be processed and its process can be stopped (because a more important job shows up) without having to pay any cost for it, neither in terms or money nor of time. In addition, if the process of this job has ended before the arrival, there is one less job in the system which can abandon. Therefore, there always exists an optimal policy without forcing the machine to idle, that is why we only consider such policies in the remainder of this proof.

Suppose that there are two jobs left, then the optimal policy is again to process the job with smallest index first, as claims Lemma 3. Now suppose that there are n jobs left and that when there are $n-1$ jobs left, it is optimal to process the job with smallest index first. Because patience times and processing times are memoryless, neither the amount of time spent waiting for the first job to leave the system, nor spent by this job in process, changes the remaining of the schedule. Let us assume again that the first job to be processed has not the smallest index. The same way as in the static case, interchanging this job with the second job reduces the overall cost of the schedule, because of Lemma 3, and it reduces the cost generated by the other $n-2$ jobs, because of Lemma 5. Consequently, it is optimal to process the job of smallest index first in the class of dynamic scheduling policies with preemption but without arrivals of jobs.

This ends the proof of Theorem 11. □

Lemma 4 and Lemma 6 allows to derive the same theorem for IBS.

Theorem 12. *Let us consider a scheduling problem with n jobs and IBS. A Job j has an exponentially distributed processing time with mean $1/\mu_j$, an exponentially distributed patience time with mean $1/\gamma_j$, an abandonment cost w_j and an arbitrarily distributed release date. If Jobs can be ordered such that Lemma 4 and Lemma 6 hold for every pair (i,j) of jobs, with $i<j$, then scheduling jobs in non-decreasing order of the indices minimizes the expected costs of abandonments in the class of dynamic policies with preemption.*

Proof. Same as for Theorem 11. □

Using Corollary 3 and Corollary 4 instead of Lemma 3 and Lemma 5, allows to derive the following corollary, which extends the results presented by Down et al. (2011). Note that Corollary 3 and Corollary 4 are also true for problems with IBS, therefore the following corollary does not depend on the type of impatience that is considered.

Corollary 5. *Let us consider a scheduling problem with n jobs either with IES or with IBS. A Job j has an exponentially distributed processing time with mean $1/\mu_j$, an exponentially distributed patience time with mean $1/\gamma_j$, an abandonment cost w_j and an arbitrarily distributed release date. If Jobs can be ordered such that*

i) $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_n$,

ii) $\gamma_1 \leq \gamma_2 \leq \cdots \leq \gamma_n$ *and*

iii) $w_1\gamma_1 \geq w_2\gamma_2 \geq \cdots \geq w_n\gamma_n$,

then scheduling jobs in non-decreasing order of the indices minimizes the expected abandonment costs in the class of dynamic policies with preemption.

4.4.3 Limit cases

Lets investigate some limit cases, when the patience rate takes some particular values. First, one can notice that when both γ_1 and γ_2 are equal and tend to 0 or to infinity, Lemma 5 states that in order to clear the system as fast as possible, the job of greatest μ_j should be processed first. Indeed, the inequality of Lemma 5 reduces to $\mu_1 \geq \mu_2$. Now focus on the cost of two consecutive jobs. And in this case, the inequality of Lemma 3 reduces to

$$\frac{w_1\mu_1}{w_2\mu_2} \geq \frac{(\mu_1\mu_2 + \gamma\mu_1 + \gamma\mu_2 + 2\gamma^2) + \gamma\mu_1}{(\mu_1\mu_2 + \gamma\mu_1 + \gamma\mu_2 + 2\gamma^2) + \gamma\mu_2}.$$

Either when γ tends to 0 or to infinity, this expression tends to the so called "$c\mu$ rule", when considering abandonment costs instead of holding costs (see Theorem 20 page 133). Now if the abandonment rate of one job tends to zero, Lemma 5 states that this job should be processed first, in order to minimize the duration of both processes. Lets assume that $\gamma_1 \to 0$, then the right hand side of the inequality of Lemma 5 tends to zero. And the inequality of Lemma 3 reduces to $w_1\gamma_1(\mu_1 + \gamma_2) \geq w_2\gamma_2\mu_2$. Again, the "$c\mu$ rule" is a sufficient condition for this inequality to hold and thus is also optimal in this case. At the opposite, if the patience rate of one job tends to infinity, Lemma 5 states that this job should be processed last. Indeed, when $\gamma_2 \to +\infty$, the inequality of Lemma 5 reduces to $\mu_1 + \gamma_1 \geq \gamma_1$, which is always true. The inequality of Lemma 3 reduces to $w_1\gamma_1\mu_1/(\mu_1 + \gamma_1) \geq w_2\mu_2$. If $w_1\mu_1$ is sufficiently larger than $w_2\mu_2$, then it is optimal to process Job 1 first. This is in contradiction with the "$c\mu$ rule", because with this rule Job 2 should be processed first. The first two examples show that when patience rates are low enough, the "$c\mu$ rule" is still optimal because there is no need to consider abandonments. But the last example shows that when impatience takes an important part in the behavior of the system, an optimal scheduling rule should take this into account, which is not the case of the "$c\mu$ rule".

4.5 Extensions

4.5.1 Infinite horizon

The policies that we introduced in Section 4.4 minimize the expected abandonment costs when n jobs enter the system. In this section we conjecture that a policy that minimizes the expected abandonment costs for n different jobs, also minimizes the long run expected abandonment costs when the system faces an infinite arrival stream from n classes of jobs. Class j jobs having the same parameters that job j.

Now we consider that n classes of jobs enter the system. Within each class j, jobs

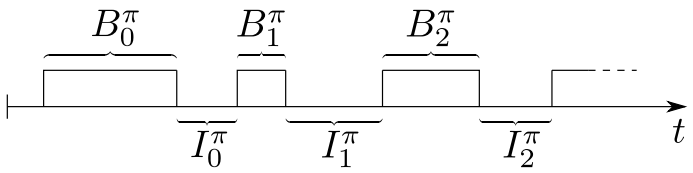


Figure 12: Busy and idle periods for a policy π

have independent and identically distributed processing times of rate μ_j, independent and identically distributed patience time of rate γ_j and have identical weight w_j. The time between two arrivals of jobs from class j follows a Poisson process with rate λ_j.

Conjecture. *A stationary policy that minimizes the expected abandonment costs for n jobs also minimizes the long run expected abandonment costs for an infinite arrival stream of n classes of jobs.*

Kofman and Lippman (1981) prove a similar result for an $M/M/1$ queue with two classes of customers and with a mechanism of promotion: they prove that a policy π minimizing the costs for a finite number of jobs, also minimizes the long run expected costs when the system faces an infinite arrival stream. They prove that the time horizon can be separated in a succession of busy periods $\{B_i^\pi\}_{i\in\mathbb{N}}$ and idle periods $\{I_i^\pi\}_{i\in\mathbb{N}}$, such as represented in Figure 12. By definition, in a busy period there is at least one job in the system, and because in their problem an idling policy is suboptimal, in a busy period the machine is also processing a job. Because there is no abandonment, the length of a busy period does not depend on the policy that is used. With their assumptions, there is a finite number of jobs entering the system in each busy period. It allows them to apply the optimal policy on each busy period and using the strong law of large numbers, to prove that this policy also minimizes the long run expected costs.

We conjecture that such a reasoning should hold for the problem we study as well. Using the same assumptions, the problem we study could also be separated in a succession of busy periods $\{B_i^\pi\}_{i\in\mathbb{N}}$ and idle periods $\{I_i^\pi\}_{i\in\mathbb{N}}$ with a finite number of arrivals in each period (due to abandonments, our system is even more stable than the one studied by Kofman and Lippman). However, in our case the length of a busy period depends on the policy that is used. As a consequence, policies cannot be compared in a similar manner. Given two policies π and π', we conjecture that the time horizon can be separated in a sequence of pseudo-busy periods $\{B_i^{\pi,\pi'}\}_{i\in\mathbb{N}}$ and pseudo-idle periods $\{I_i^{\pi,\pi'}\}_{i\in\mathbb{N}}$ as shown in Figure 13. If this is the case, policies can be pairwise compared on a pseudo-busy period and the conclusion would be that a policy minimizing the expected costs on a pseudo-busy period, also minimizes the long run expected costs when the system faces an infinite arrival stream. In our opinion the difficulty resides in proving that a finite number of jobs show up in the system during such a pseudo-busy period.

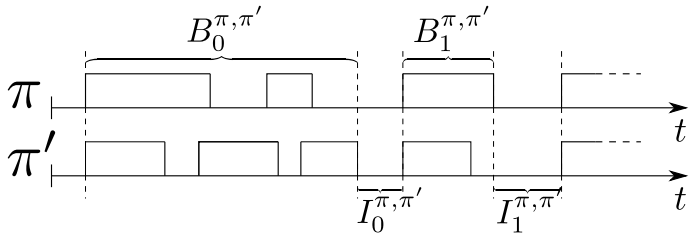


Figure 13: Busy and idle periods for two policies: π and π'

4.5.2 Non exponential processing times

In this section we assume that processing times have increasing completion rate (ICR) distributions. If the processing time of a job is an ICR random variable, that means that the job is all the more likely to end that it has spent time in process (Pinedo 2008).

Theorem 13. *Let us consider a scheduling problem with n jobs all of them available at time zero. A Job j has an arbitrarily distributed processing time that is ICR and an abandonment cost w_j. Patience times are i.i.d. and exponentially distributed with mean $1/\gamma$. If Jobs can be ordered such that*

$$\textit{i)} \quad \frac{w_1\mathcal{L}\{f_{X_1}\}(\gamma)}{1-\mathcal{L}\{f_{X_1}\}(2\gamma)} \geq \frac{w_2\mathcal{L}\{f_{X_2}\}(\gamma)}{1-\mathcal{L}\{f_{X_2}\}(2\gamma)} \geq \cdots \geq \frac{w_n\mathcal{L}\{f_{X_n}\}(\gamma)}{1-\mathcal{L}\{f_{X_n}\}(2\gamma)} \qquad \textit{and}$$

$$\textit{ii)} \quad \frac{1-\mathcal{L}\{f_{X_1}\}(2\gamma)}{1-\mathcal{L}\{f_{X_1}\}(\gamma)} \geq \frac{1-\mathcal{L}\{f_{X_2}\}(2\gamma)}{1-\mathcal{L}\{f_{X_2}\}(\gamma)} \geq \cdots \geq \frac{1-\mathcal{L}\{f_{X_n}\}(2\gamma)}{1-\mathcal{L}\{f_{X_n}\}(\gamma)},$$

then scheduling jobs in non-decreasing order of the indices minimizes the expected abandonment costs in the class of dynamic policies with IES.

Before presenting the proof of this theorem, note that it is consistent with the previous results. For 2 jobs, when the processing times are exponentially distributed, the two conditions reduce to

$$\textbf{i)} \quad \frac{w_1\mu_1}{(\mu_1+\gamma)(\mu_2+2\gamma)} \geq \frac{w_2\mu_2}{(\mu_2+\gamma)(\mu_1+2\gamma)} \qquad \text{and}$$

$$\textbf{ii)} \quad \frac{1}{(\mu_2+\gamma)(\mu_1+2\gamma)} \geq \frac{1}{(\mu_1+\gamma)(\mu_2+2\gamma)},$$

which are consistent with both Lemma 3 (page 63) and Lemma 5 (page 65).

Proof. Starting from Property 2 (page 63) we compare the costs of the two following

schedules

$$\begin{aligned}C(\{2,1\}) - C(\{1,2\}) &= w_2\mathbb{P}(X_2 \geq D_2) + w_1[1 - \mathbb{P}(C_2 \leq D_1)\mathbb{P}(X_1 \leq D_1)] \\ &\quad - w_1\mathbb{P}(X_1 \geq D_1) - w_2[1 - \mathbb{P}(C_1 \leq D_2)\mathbb{P}(X_2 \leq D_2)] \\ &= w_1\mathbb{P}(C_2 \geq D_1)\mathbb{P}(X_1 \leq D_1) \\ &\quad - w_2\mathbb{P}(C_1 \geq D_2)\mathbb{P}(X_2 \leq D_2).\end{aligned}$$

Using Laplace transforms and the relation proved in Appendix C.3 (page 135)

$$\begin{aligned}C(\{2,1\}) - C(\{1,2\}) \geq 0 \Leftrightarrow \frac{w_1\gamma_1}{\gamma_1+\gamma_2}\left[1 - \mathcal{L}\{f_{X_2}\}(\gamma_1+\gamma_2)\right]\mathcal{L}\{f_{X_1}\}(\gamma_1) \\ - \frac{w_2\gamma_2}{\gamma_1+\gamma_2}\left[1 - \mathcal{L}\{f_{X_1}\}(\gamma_1+\gamma_2)\right]\mathcal{L}\{f_{X_2}\}(\gamma_2) \geq 0 \\ \Leftrightarrow \frac{w_1\gamma_1\mathcal{L}\{f_{X_1}\}(\gamma_1)}{1-\mathcal{L}\{f_{X_1}\}(\gamma_1+\gamma_2)} \geq \frac{w_2\gamma_2\mathcal{L}\{f_{X_2}\}(\gamma_2)}{1-\mathcal{L}\{f_{X_2}\}(\gamma_1+\gamma_2)}.\end{aligned}$$

And if patience times are identically distributed with rate γ

$$C(\{2,1\}) - C(\{1,2\}) \geq 0 \Leftrightarrow \frac{w_1\mathcal{L}\{f_{X_1}\}(\gamma)}{1-\mathcal{L}\{f_{X_1}\}(2\gamma)} \geq \frac{w_2\mathcal{L}\{f_{X_2}\}(\gamma)}{1-\mathcal{L}\{f_{X_2}\}(2\gamma)}.$$

Starting from Property 3 (page 65) we compare the time needed by both schedules to clear the system (we take advantage of Property 9 page 132)

$$\begin{aligned}E(T_{\{2,1\}}) - E(T_{\{1,2\}}) &= E(\min(X_2, D_2)) + E(\min(X_1, D_1))\mathbb{P}(C_2 \leq D_1) \\ &\quad - E(\min(X_1, D_1)) - E(\min(X_2, D_2))\mathbb{P}(C_1 \leq D_2) \\ &= E(D_2)\mathbb{P}(X_2 \geq D_2)\mathbb{P}(C_1 \geq D_2) \\ &\quad - E(D_1)\mathbb{P}(X_1 \geq D_1)\mathbb{P}(C_2 \geq D_1) \\ &= \frac{1}{\gamma_2}(1 - \mathcal{L}\{f_{X_2}\}(\gamma_2))\frac{\gamma_2}{\gamma_1+\gamma_2}\left[1 - \mathcal{L}\{f_{X_1}\}(\gamma_1+\gamma_2)\right] \\ &\quad - \frac{1}{\gamma_1}(1 - \mathcal{L}\{f_{X_1}\}(\gamma_1))\frac{\gamma_1}{\gamma_1+\gamma_2}\left[1 - \mathcal{L}\{f_{X_2}\}(\gamma_1+\gamma_2)\right].\end{aligned}$$

Hence, when patience times are identically distributed with rate γ

$$E(T_{\{2,1\}}) - E(T_{\{1,2\}}) \geq 0 \Leftrightarrow \frac{1-\mathcal{L}\{f_{X_1}\}(2\gamma)}{1-\mathcal{L}\{f_{X_1}\}(\gamma)} \geq \frac{1-\mathcal{L}\{f_{X_2}\}(2\gamma)}{1-\mathcal{L}\{f_{X_2}\}(\gamma)}.$$

As a consequence, if jobs can be ordered such that

i) $w_1\frac{\mathcal{L}\{f_{X_1}\}(\gamma)}{1-\mathcal{L}\{f_{X_1}\}(2\gamma)} \geq w_2\frac{\mathcal{L}\{f_{X_2}\}(\gamma)}{1-\mathcal{L}\{f_{X_2}\}(2\gamma)}$ and

ii) $\frac{1-\mathcal{L}\{f_{X_1}\}(2\gamma)}{1-\mathcal{L}\{f_{X_1}\}(\gamma)} \geq \frac{1-\mathcal{L}\{f_{X_2}\}(2\gamma)}{1-\mathcal{L}\{f_{X_2}\}(\gamma)}$,

then schedule $\{1,2\}$ is less expensive and clears the system faster than schedule $\{2,1\}$. When there are n jobs to process, the proof is made by induction on the number of jobs remaining. The sequence of arguments for this proof are the same as for the proof of

Theorem 11 without arrivals (page 67). Except that the argument that allows us to use the induction hypothesis was the memoryless property of the exponential distribution. In this case, because all processing times are ICR, when the job of lowest index is being processed it is not optimal to revoke this decision until the end of its process, and the induction hypothesis can be applied. This ends the proof of Theorem 13. □

Again, the same theorem can be derived for IBS.

Theorem 14. *Let us consider a scheduling problem with n jobs all of them available at time zero. A Job j has an arbitrarily distributed processing time that is ICR and an abandonment cost w_j. Patience times are i.i.d. and exponentially distributed with mean $1/\gamma$. If Jobs can be ordered such that*

$$\textit{i)} \quad \frac{w_1}{1-\mathcal{L}\{f_{X_1}\}(\gamma)} \geq \frac{w_2}{1-\mathcal{L}\{f_{X_2}\}(\gamma)} \geq \cdots \geq \frac{w_n}{1-\mathcal{L}\{f_{X_n}\}(\gamma)} \qquad \textit{and}$$

$$\textit{ii)} \quad \frac{E(X_1)}{1-\mathcal{L}\{f_{X_1}\}(\gamma)} \leq \frac{E(X_2)}{1-\mathcal{L}\{f_{X_2}\}(\gamma)} \leq \cdots \leq \frac{E(X_n)}{1-\mathcal{L}\{f_{X_n}\}(\gamma)},$$

then scheduling jobs in non-decreasing order of the indices minimizes the expected abandonment costs in the class of dynamic policies with IBS.

Proof. Same as for Theorem 13. □

This result is also consistent with the previous lemmas. For 2 jobs, when the processing times are exponentially distributed, the two conditions reduce to

$$\textbf{i)} \quad w_1(\mu_1+\gamma) \geq w_2(\mu_2+\gamma) \qquad \text{and}$$

$$\textbf{ii)} \quad \frac{1}{\mu_1(\mu_2+\gamma)} \geq \frac{1}{\mu_2(\mu_1+\gamma)},$$

which are consistent with both Lemma 4 (page 64) and Lemma 6 (page 66).

4.6 Numerical results

In this section we present a numerical study that compare several heuristics. This is motivated by the fact that the set of two conditions of Theorem 11 for IES (page 67), and Theorem 12 for IBS (page 68), are not respected by all possible instances. Hence, we compare the efficiency of strict priority rules to the optimal dynamic policy. We assume that the system is markovian and that jobs become available according to a Poisson process. This problem is formulated as a Markov decision process (see Appendix C.4 page 136 for details on this formulation). The optimal policy is computed using value iteration and due to computational complexity we consider only two classes of jobs. The optimal policy is then compared to heuristics from the literature and new heuristics that we propose.

The performance of these heuristics are compared to the optimal solution computed using value iteration with a precision of 10^{-8} over a set of 527472 instances. Note that the value iteration algorithm diverges when used to compute the expected cost, but the difference of the costs between two iterations converges to the average cost (up to a constant factor). The parameters of the instances are, for each class j, the arrival rate λ_j, the production rate μ_j, the abandonment rate γ_j and the abandonment cost w_j. Without loss of generality we fix $\lambda_2 = 1$ and $w_2 = 1$. The values of the other parameters are chosen exhaustively among $\{0.1, 0.11, 0.12, 1, 1.1, 1.2, 10, 11, 12\}$. These values generate instances 1) for which parameters can be different but from the same order of magnitude, and 2) for which parameters can be from different orders of magnitude. These values cater the generality of a random instance. Instances for which class 1 and class 2 have the same value for every parameter are not considered. Symmetric instances (two instances are the same up to an interchange of the two classes) are taken only once into account.

Before presenting the numerical results of these heuristics we discuss the results of Theorem 11 and Theorem 12. On this set of instances, these theorems are able to solve to optimality almost one half of the instances. Indeed, Theorem 11 can be applied to 43.54% of the instances and Theorem 12 to 48.56% of the instances.

We also compare Theorem 11 with the optimal priority rule proposed by Down et al. (2011) when $\mu_1 = \mu_2$ (if $w_1\gamma_1 \geq w_2\gamma_2$ and $\gamma_1 \leq \gamma_2$ then it is optimal to give priority to class 1). The number of instances for which $\mu_1 = \mu_2$ is 57996, and Theorem 11 can be applied to 33.84% of such instances whereas the rule of Down et al. (2011) can only be applied to 27.76% of the instances.

Corollary 5, which is a direct generalization of the result of Down et al. (2011) with $\mu_1 \geq \mu_2$, can only be applied to 16.67% of instances from the whole set, instead of 43.54%. Consequently the rule we propose extends significantly the set of instances for which an optimal strict priority rule can be given.

4.6.1 Heuristics

We compare a set of seven heuristics to the optimal policy that are:

- Lemma 3 for IES (page 63) and Lemma 4 for IBS (page 64): priority is given to the less expensive class according to these lemmas;
- $w\gamma$: priority is given to the class of highest $w_j\gamma_j$, this heuristic is inspired by Corollary 5 (page 68);
- μ: priority is given to the class of highest μ_j, this heuristic is inspired by Corollary 5 as well;
- γ: priority is given to the class of highest γ_j, when the processing times are identically distributed and the weights are equal, this heuristic is optimal for both IES (Down et al. 2011) and IBS problems (Argon et al. 2008);
- $w\mu$: priority is given to the class of highest $w_j\mu_j$, this heuristic is optimal in the limit fluid scaling studied by Atar et al. (2010);

- $w\gamma\mu$: priority is given to the class of highest $w_j\gamma_j\mu_j$, this heuristic is the "$c\mu$"-rule considering abandonments costs, this rule is optimal for minimizing the total flow time;
- w: priority is given to the class of highest weight.

In addition to these seven heuristics, we present the results of three artificial heuristics:

- best: takes the same decision as the best of the seven heuristics presented above;
- one: priority is given to class 1;
- two: priority is given to class 2.

The results are given in Table 7 for instances with IES, and in Table 8 for instances with IBS. The tables are organized in three parts. The first part presents the results for all the instances. The second part presents the results for instances such that the optimal policy is a strict priority rule (priority is given to the same class on the whole state space) which represents 99.79% of all IES instances and 99.84% of all IBS instances. The third part presents the results for instances such that the optimal policy is of a threshold type (priority is given to a class below some threshold and to the other class above this threshold) which represents 0.21% of all IES instances and 0.16% of all IBS instances. All three parts contain the mean error (in %) between the cost of the optimal policy and the cost of the heuristic (as well as the standard deviation of this error). In the two first parts, we give the percentage of instances for which the heuristic leads to the optimal policy.

From Table 7, the best heuristics to solve a problem with IES are Lemma 3 which is in average at 0.34% of the optimal policy, second comes the priority to the highest $w\gamma\mu$ and third comes the priority to the highest $w\mu$. Even though there are only two possible priority policies (priority to class 1 or to class 2), the p-values of t-tests performed on the errors show that their mean are significantly different (the p-values are all lower than 10^{-6}). Table 9 gives the number of instances for which one of the three best heuristics performs better than the other. These figures confirm the ranking we presented previously. Even the worst heuristic, the priority to the highest γ which is at 9.45% of the optimum, performs better than Lemma 3 for 5.21% of the instances. Note that the number of instances for which the optimal policy is of a threshold type is small compared to all the instances we investigates. It justifies the search for a good performing strict priority rule because priority threshold appear less common. For such instances, the heuristics are close to the optimum ($\leq 1.01\%$). Moreover one can note that the ranking of the different heuristics is not the same as for the whole set of instances; the best heuristic is to give priority to the class of highest γ and Lemma 3 ends in the fifth position.

From Table 8, the three best heuristics to solve a problem with IBS are the same as for IES problems (again, a comparison between these three heuristics is given in Table 10). However the heuristic that gives priority to the class of highest $w\mu$ performs better than the heuristic giving priority to the class of highest $w\gamma\mu$. All heuristics are further from the optimal policy for IBS: 1.73% for the best heuristic and 53.79% for the worst. Again, the worst heuristic sometimes outperform the best heuristic (2.08% of the instances). When

Heuristic	All (527472 instances)			Strict (99.79%)			Other (0.21%)	
	Error (%)	(std. dev.)	Opt. (%)	Error (%)	(std. dev.)	Opt. (%)	Error (%)	(std. dev.)
Lemma 3	0.34	(4.54)	89.51	0.31	(4.54)	89.69	0.82	(3.26)
$w\gamma\mu$	0.59	(5.80)	88.52	0.59	(5.80)	88.70	0.78	(3.25)
$w\mu$	0.65	(6.22)	86.26	0.65	(6.22)	86.44	0.95	(2.96)
μ	2.57	(20.32)	73.34	2.57	(20.34)	73.49	0.80	(2.76)
w	3.58	(16.08)	70.74	3.58	(16.09)	70.89	1.01	(3.26)
$w\gamma$	2.91	(13.27)	72.07	2.91	(13.29)	72.22	0.74	(3.15)
γ	9.45	(57.97)	60.94	9.47	(58.04)	61.06	0.60	(2.68)
best	< 0.01	(0.86)	96.33	< 0.01	(0.86)	96.53	0.10	(0.34)
one	18.12	(81.01)	50.99	18.15	(81.09)	51.09	0.95	(3.29)
two	12.60	(62.71)	48.80	12.63	(62.77)	48.91	0.51	(1.44)

Table 7: Performance of heuristics for problems with IES

Heuristic	All (527472 instances)			Strict (99.84%)			Other (0.16%)	
	Error (%)	(std. dev.)	Opt. (%)	Error (%)	(std. dev.)	Opt. (%)	Error (%)	(std. dev.)
Lemma 4	1.73	(11.80)	91.21	1.73	(11.81)	91.34	1.41	(3.61)
$w\gamma\mu$	3.37	(17.56)	85.89	3.38	(17.57)	86.02	2.23	(6.67)
$w\mu$	3.05	(19.72)	87.86	3.04	(19.70)	88.00	11.12	(28.17)
μ	13.00	(68.72)	71.62	13.01	(68.77)	71.73	9.52	(26.82)
w	16.05	(70.04)	74.24	16.07	(70.09)	74.36	5.50	(15.89)
$w\gamma$	27.35	(67.93)	72.35	27.39	(68.06)	72.46	1.33	(3.36)
γ	53.79	(155.15)	58.50	53.87	(155.34)	58.57	1.95	(6.36)
best	< 0.01	(1.06)	97.37	< 0.01	(1.07)	97.52	0.35	(0.79)
one	94.32	(292.65)	53.06	94.46	(292.94)	53.14	4.11	(11.97)
two	56.73	(91.95)	46.79	56.81	(92.09)	46.86	8.34	(26.44)

Table 8: Performance of heuristics for problems with IBS

H \ H'	Lemma 3	$w\gamma\mu$	$w\mu$
Lemma 3		5.97	11.14
$w\gamma\mu$	5.88		11.73
$w\mu$	8.78	9.46	

Table 9: Percentage of instances for which heuristic H performs better than heuristic H' when IES is considered

H \ H'	Lemma 4	$w\gamma\mu$	$w\mu$
Lemma 4		8.34	10.31
$w\gamma\mu$	3.48		8.50
$w\mu$	9.61	12.66	

Table 10: Percentage of instances for which heuristic H performs better than heuristic H' when IBS is considered

we consider instances for which the optimal policy is of a threshold type, which represents 819 instances (0.16%), the priority corresponding to Lemma 4 performs at 1.41% of the optimal policy, slightly worse than $w\gamma$ which is at 1.33%, but the priority to the highest $w\mu$ has a poor performance at 10.64%. It shows that for such instances, the difference of costs between the different heuristics is quite significant.

Table 11 gives the ability of a heuristic to find the optimal policy with respect to the type of impatience. For each heuristic, we give the percentage of instances for which it gives the optimal policy for both IES and IBS problems, for IES problems only, for IBS problems only, or for neither IES nor IBS problems. It shows again that the three same heuristics have a good performance and that the heuristics based on the two lemmas are better than the other. Note that the optimal policy gives priority to the same class in IES and IBS for 95.43% of the instances, that priority is given to one class in IES and to the other in IBS for 4.56% of the instances and that the optimal policy is of a threshold type for both type of impatience for less than 0.01% of the instances. However Lemma 3 is more robust against the type of impatience than Lemma 4. Indeed, Lemma 3 is optimal neither for IES nor for IBS only for 2.03% of the instances, whereas Lemma 4 is optimal neither for IES nor for IBS for 7.45% of the instances.

From these results, the new heuristics that we propose, based on Lemma 3 for IES (page 63) and on Lemma 4 for IBS (page 64), perform better than other heuristics from the literature ($w\gamma\mu$ and $w\mu$). These results also teach us that decision makers should be sure about what kind of impatience they are facing (whether IBS or IES), in order to be aware of the performance of the heuristics they are implementing. Indeed, heuristics are

Heuristic	Optimality for (% of cases)			
	both IES and IBS	IES only	IBS only	neither IES nor IBS
Lemma 3	93.45	3.27	1.26	2.03
Lemma 4	88.04	1.41	3.10	7.45
$w\gamma\mu$	84.97	3.55	0.92	10.56
$w\mu$	84.88	1.38	2.98	10.76
μ	70.31	3.03	1.31	25.36
w	70.28	0.46	3.97	25.29
$w\gamma$	69.96	2.10	2.39	25.55
γ	57.48	3.45	1.00	38.07

Table 11: Performance of heuristics for both IES and IBS problems on the same set of parameters

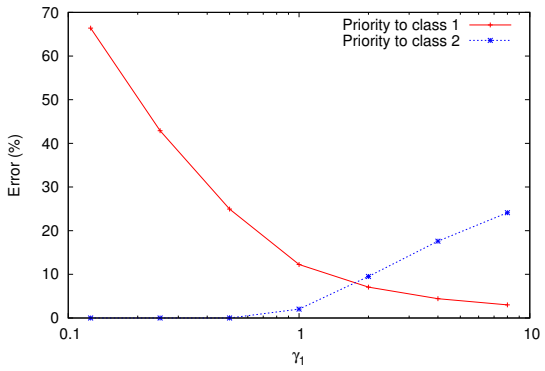
able to solve a comparable amount of instances to optimality, but their performances from one type of impatience to the other are not from the same order of magnitude.

4.6.2 Optimality as a function of impatience: a case study

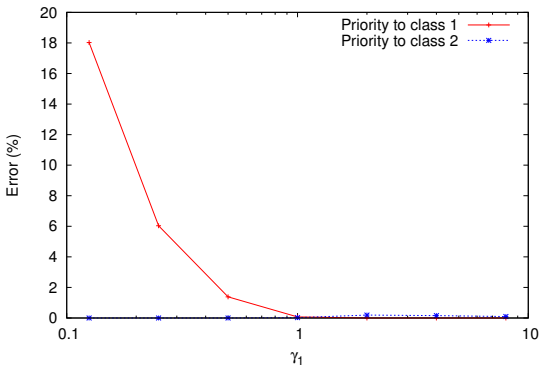
In this section we discuss the effect of the type of impatience, and the effect of the impatience rate, on the performance of strict priority rules. We compare the performance of the two possible strict priority rules (giving priority to class 1, or to class 2) on two instances with the same parameters, except that one instance considers a problem with IES and the other a problem with IBS.

Figure 14 and Figure 15 depict the evolution of the error, compared to the optimal policy, of the two possible strict priority rules (priority to class 1 or to class 2) starting from the same instance for both type of impatience ($\lambda_1 = 0.1$, $\lambda_2 = 1$, $\mu_1 = 0.11$, $\mu_2 = 11$, $\gamma_1 = 1$, $\gamma_2 = 0.1$, $w_1 = 11$, $w_2 = 1$). First we observe that for both types of impatience, when γ_j tends to infinity, priority should be given to class j and when γ_j tends to zero priority should be given to the other class. This observation makes sense for IBS because when processing the most impatient class first, there is a higher chance that more jobs are still in the system when the processing ends. Therefore the number of abandonments are minimized (Argon et al. 2008). For IES, giving priority to the less patient job seem not to be optimal because the process has lower chances to end before its patience time. However if it becomes impatient very quickly the decision maker lose virtually no time trying to process it. This is consistent with a result from Down et al. (2011) which gives priority to class 1 if $w_1 \geq w_2$, $\gamma_1 \geq \gamma_2$ and $\mu_1 = \mu_2$.

These figures confirm the results of the systematic tests: the cost of heuristics are more variable and more sensitive to changes of parameters when solving problems with IBS than with IES (see the scaling of the different figures). We observe that the range of parameters

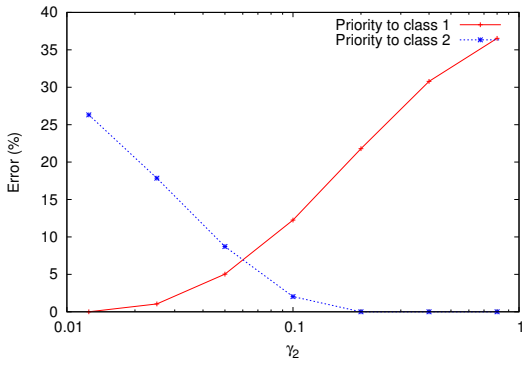


Problem with IBS

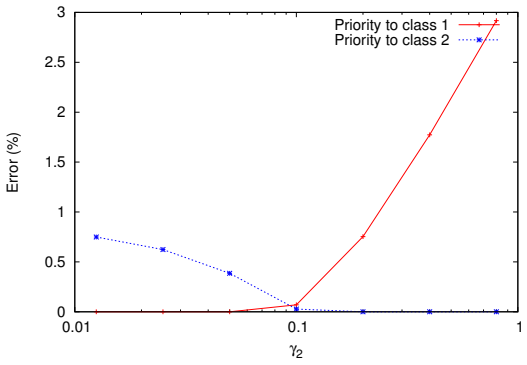


Problem with IES

Figure 14: Evolution of the error (in %) as a function of γ_1 between the two possible strict priority rules and the optimal policy for the two types of impatience. Parameters of the instance: $\lambda_1 = 0.1$, $\lambda_2 = 1$, $\mu_1 = 0.11$, $\mu_2 = 11$, $\gamma_2 = 0.1$, $w_1 = 11$, $w_2 = 1$



Problem with IBS



Problem with IES

Figure 15: Evolution of the error (in %) as a function of γ_2 between the two possible strict priority rules and the optimal policy for the two types of impatience. Parameters of the instance: $\lambda_1 = 0.1$, $\lambda_2 = 1$, $\mu_1 = 0.11$, $\mu_2 = 11$, $\gamma_1 = 1$, $w_1 = 11$, $w_2 = 1$

for which the optimal policy is of a threshold type (neither priority to class 1 nor to class 2 has an error equal to zero) is wider for problems with IBS than for IES, and that the worst case error is much larger.

When the optimal policy has a priority threshold, this threshold seems to be in general decreasing in function of the number of customers of class 1, as shown in figures 16, 17 and 18. Figures 16 and 17 show the optimal policy for the instances from our study having the greatest error between the best strict priority policy and the optimal policy (6.7% for the IBS problem and 0.16% for the IES problem). We see that the threshold is located near the origin. On the other hand when the threshold tends to get further away from the origin as in Figure 18, the difference between the best strict priority policy and the optimal policy becomes insignificant (around 10^{-5}). This is due to the fact that the probability of

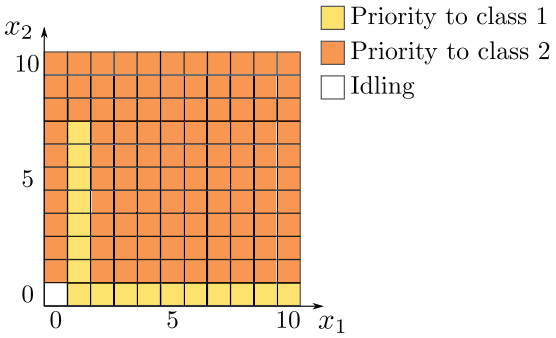


Figure 16: Optimal IBS policy. Parameters of the instance: $\lambda_1 = 0.1$, $\lambda_2 = 1$, $\mu_1 = 0.11$, $\mu_2 = 11$, $\gamma_1 = 1$, $\gamma_2 = 0.06$, $w_1 = 11$, $w_2 = 1$

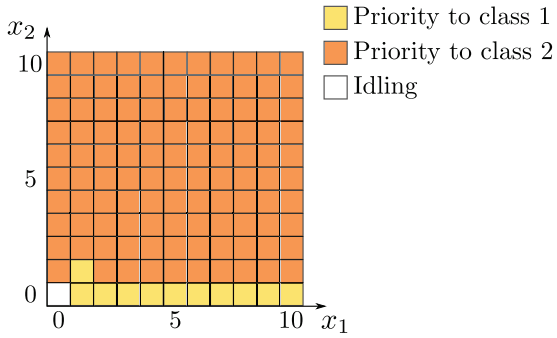


Figure 17: Optimal IES policy. Parameters of the instance: $\lambda_1 = 0.1$, $\lambda_2 = 1$, $\mu_1 = 1$, $\mu_2 = 10$, $\gamma_1 = 1.1$, $\gamma_2 = 0.115$, $w_1 = 1.2$, $w_2 = 1$

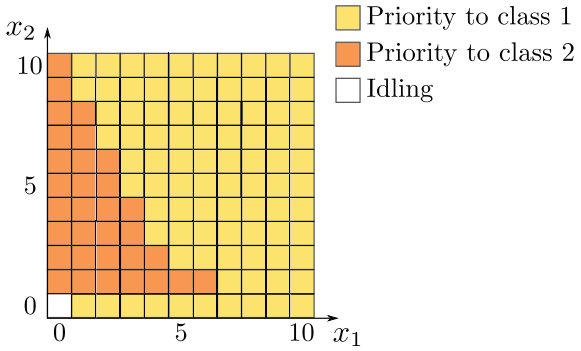


Figure 18: Optimal IBS policy. Parameters of the instance: $\lambda_1 = 1$, $\lambda_2 = 1$, $\mu_1 = 1$, $\mu_2 = 1$, $\gamma_1 = 1$, $\gamma_2 = 2$, $w_1 = 1.01$, $w_2 = 1$

being in a state decreases rapidly as this state gets further from the origin. Consequently, changing a decision near the origin as a much significant impact than e.g. in state $(10, 10)$ because the weight of this cost on the expected cost is much higher.

4.7 Conclusion

In this chapter we present three results concerning scheduling jobs with due dates. We propose a strict priority rule that minimizes the expected costs of abandonment and that extends the results of the literature. The results we found are consistent with the priority rule from Down et al. (2011). The second result is an equivalence between two cost models. When due dates are exponentially distributed, considering a cost for each late job or a holding cost per unit of time for each job waiting in the system (or both costs), is equivalent. This result allows to study the scheduling problem with due dates, considering only one type of costs. The third result comes from the numerical study. We propose a heuristic for both IES and IBS problems, based on Lemma 3 and Lemma 4 (pages 63 and 64), that performs better than previous strict priority policies proposed in

the literature. In addition, we outline that the two kinds of impatience have a different impact in terms of variability of costs. We show that IBS problems are more sensitive to changes of parameters and that the best heuristic is further from the optimum. The same heuristic performs well for IBS and IES, but knowing exactly what kind of impatience the system is facing is essential for an accurate assessment of the performance of the system.

Chapitre 5

A discussion on variants and extensions

In this chapter we discuss two main avenues for future research on stochastic scheduling with impatience, and whether it is possible or not to apply the same techniques that we used in Chapter 3 and Chapter 4 in order to solve these problems. These extensions involve problems without preemption, and/or problems with several machines which process jobs in parallel, as shown in Figure 19.

In Section 5.1 we consider static scheduling policies for parallel machines problems. Some results exist in the IES case, we discuss the case of IBS and the case of machines having different processing times.

In Section 5.2 we consider dynamic scheduling problems without preemption. In the previous chapters, either preemption is authorized in a dynamic markovian problem and decisions are made at each new event, or all jobs are available at time zero in a static problem and preemption is not necessary. In this chapter we discuss the case of dynamic scheduling without preemption, in both cases of single and parallel machines problems.

5.1 Static scheduling policies for multi-machine problems with impatience

5.1.1 Problem description and literature review

We consider a stochastic scheduling problem with impatience, where n jobs have to be processed and m parallel machines with different speed are available. Job j has a weight w_j a random processing time $X_{i,j}$ and a random due date D_j: $Qm \mid X_{i,j}, D_j \mid E(\sum w_j U_j)$ (or $E(\sum w_j \tilde{U}_j)$). The optimal policy is searched among the class of non preemptive static scheduling policies. We assume that for static scheduling problems, the decision maker has to build a priority list for each machine. That is at time 0 every job is inserted into a list and associated to one machine, no jockeying is allowed between any two machines after

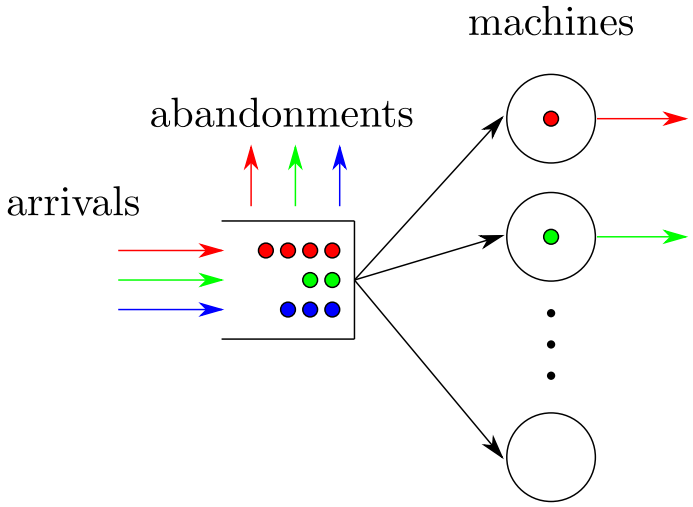


Figure 19: Problem with several classes of jobs and several machine in parallel

time 0. Consequently, assuming that the machine associated with each job is known, the problem of building an optimal schedule on each machine can be solved using the techniques presented in Chapter 3. But deciding on which machine a job should be processed is a new decision to make.

For the problem without abandonments, when the objective is to minimize the flow time, Weber et al. (1986) prove that SEPT is optimal on parallel machines in the class of non preemptive dynamic policies, if the processing times are stochastically ordered: $Pm \mid j < j' \Rightarrow X_j \leq_{st} X_{j'} \mid E(\sum C_j)$. Agrawala et al. (1984) prove that when processing times are exponentially distributed with rate μ_i on machine i: $Qm \mid X_{i,j} \sim \exp(\mu_i) \mid E(\sum C_j)$, then the optimal non preemptive dynamic policy has a threshold. A job is processed on machine i only if the number of jobs waiting is greater than some threshold $t(i)$. When jobs have due dates, Emmons and Pinedo (1990) study the problem $Pm \mid X_j \sim \exp(1), D_j \sim \exp(\gamma_j) \mid E(\sum w_j U_j)$. They prove that the optimal static scheduling policy can be obtained by solving an assignment problem. We study the variant with IBS and we extend their result for parallel machines where $X_j \sim \exp(\mu_i)$. Chang et al. (1992) prove that the $c\mu$ rule is optimal for the problem $Pm \mid X_j \sim \exp(\mu_j), D_j = D \mid E(\sum w_j U_j)$ in the class of preemptive dynamic policies.

In this section we discuss the complexity of stochastic scheduling problems as well. In order to derive these complexity results, we assume that the Laplace transform of any c.d.f. can be computed in constant time. Furthermore, we consider problems where processing times depend only on the machine on which the job is processed and not on the job itself (for any job j on machine i, we have $X_j \sim F_i$).

5.1.2 Single class

The single class problem, where n identical jobs have to be processed on m parallel machines, is widely studied in optimal control of queues. The assignment of jobs to the machines is trivial, but the difficulty lies in deciding when a machine should be turned on or off. In the dynamic case without abandonments, the problem has been studied by Hajek (1984) and Koole (1995) (in the literature of optimal control of queues) and by Agrawala et al. (1984) (in the literature on scheduling problems). We provide a new result for this simple case for both type of impatience, but we show that the algorithm we propose is not polynomial in the size of an instance. However, we prove that the objective function is convex and we propose a solution in constant time for the problem with two machines.

Theorem 15. *Consider a static scheduling problem with either IES or IBS, where n identical jobs have to be processed on m different machines. Each job has an exponentially distributed due date with rate γ, and processing time on machine i is generally distributed, $X_{i,j} \sim F_{X_i}$: $Qm \mid X_{i,j} \sim F_{X_i}, D_j \sim \exp(\gamma) \mid \sum U_j$ (or $\sum \tilde{U}_j$). Denote $c_{i,k} = 1 - \mathcal{L}\{f_{X_i}\}(\gamma)^k$ for IES and $c_{i,k} = 1 - \mathcal{L}\{f_{X_i}\}(\gamma)^{k-1}$ for IBS, for $i \in \{1, 2, \ldots, m\}$ and $k \in \{1, 2, \ldots, n\}$, the probability that a job ends late when it is processed on machine i in position k (from Corollary 2 page 52). We define $\sigma : \{1, nm\} \to \{1, n\} \times \{1, m\}$ such that if $r < s$, then $c_{\sigma(r)} \leq c_{\sigma(s)}$. If r, such that $\sigma(r) = (i, k)$, is less or equal to n, then it is optimal to process a job on machine i in position k.*

Proof. Let $x_{i,k}$, $i \in \{1, 2, \ldots, m\}$ and $k \in \{1, 2, \ldots, n\}$, be equal to 1 if a job is processed on machine i in position k and 0 if machine i is free in position k. An optimal assignment of jobs for this problem is given by the solution of the following integer linear program:

$$\begin{aligned} \min & \sum_{i=1}^{m} \sum_{k=1}^{n} c_{i,k} x_{i,k} \\ \text{s.t.} & \sum_{i=1}^{m} \sum_{k=1}^{n} x_{i,k} = n \\ & x_{i,k} \in \{0, 1\}, \forall i \in \{1, 2, \ldots, m\}, \forall k \in \{1, 2, \ldots, n\} \end{aligned}$$

With this formulation, it is clear that if $r \leq n$ then $x_{\sigma(r)} = 1$. □

Note that in such a solution there is no idle time. The values of $c_{i,k}$ being increasing in k, any solution such that $x_{i,k} = 0$ and $x_{i,k'} = 1$ for any $k < k'$ is suboptimal.

As a consequence, an optimal solution for this problem can be computed in time $O(nm \log nm)$ (assuming that Laplace transforms can be computed in $O(1)$). This complexity is the result of sorting all the nm values of $c_{i,k}$. However, it is not clear that the complexity of this algorithm is polynomial in the size of the instance or not, since complexity is not clearly defined for stochastic problems. For example, consider such a problem having exponentially distributed processing times with rate μ_i on machine i. The size of

such an instance is $O(\log n + \log \gamma + m \log(\max_i \mu_i))$, if the rate is sufficient to describe an exponential distribution. In this case the complexity of the algorithm we propose is clearly not polynomial in the size of an instance. This is a typical example of high multiplicity scheduling. Other methods should then be proposed to solve this problem more efficiently.

However, for an arbitrary number of machines and for exponentially distributed processing times, we are able to prove that the objective function is convex.

Property 4. *Consider a static scheduling problem with IES, where n identical jobs have to be processed on m machines. Each job has an exponentially distributed due date with rate γ, and processing time on machine i is exponentially distributed with rate μ_i, such that $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_m$. Denote $C(n_1, n_2, \ldots, n_{m-1})$ the cost of the schedule where n_1 jobs are processed on machine 1, n_2 on machine 2, ... and $n - \sum_{i=0}^{m-1} n_i$ on machine m. Then the real valued function $C(n_1, n_2, \ldots, n_{m-1})$ (where $\{n_1, n_2, \ldots, n_{m-1}\}$ are real numbers) is strictly convex.*

Proof. Let $C_i(n)$, for all $i \in \{1, 2, \ldots, m\}$, be the cost of scheduling n jobs on machine i. From Corollary 2 (page 52)

$$\begin{aligned} C_k(n_k) &= n_k - \sum_{i=1}^{n_k} \left(\frac{\mu_k}{\mu_k + \gamma} \right)^i \\ &= n_k - p_k \frac{1 - p_k^{n_k}}{1 - p_k} \\ &= n_k - \frac{p_k}{1 - p_k} + \frac{p_k^{n_k+1}}{1 - p_k}. \end{aligned} \tag{5.1}$$

Then the total cost of the schedule is

$$\begin{aligned} C(n_1, n_2, \cdots, n_{m-1}) &= \sum_{k=1}^{m-1} C_k(n_k) + C_m \left(n - \sum_{i=0}^{m-1} n_i \right) \\ &= n - \sum_{k=1}^{m} \frac{p_k}{1 - p_k} + \sum_{k=1}^{m-1} \frac{p_k^{n_k+1}}{1 - p_k} + \frac{p_m^{n - \sum_{k=1}^{m-1} n_k + 1}}{1 - p_m}, \end{aligned}$$

the last is separated because n_m is not defined. First order derivatives are given by

$$\frac{\partial C}{\partial n_i}(n_1, n_2, \cdots, n_{m-1}) = \ln(p_i) \frac{p_i^{n_i+1}}{1 - p_i} - \ln(p_m) \frac{p_m^{n - \sum_{k=1}^{m-1} n_k + 1}}{1 - p_m}$$

and second order derivatives by

$$\frac{\partial^2 C}{\partial n_i^2}(n_1, n_2, \cdots, n_{m-1}) = (\ln(p_i))^2 \frac{p_i^{n_i+1}}{1 - p_i} + (\ln(p_m))^2 \frac{p_m^{n - \sum_{k=1}^{m-1} n_k + 1}}{1 - p_m} > 0 \qquad \text{and}$$

$$\frac{\partial^2 C}{\partial n_i \partial n_j}(n_1, n_2, \cdots, n_{m-1}) = (\ln(p_m))^2 \frac{p_m^{n - \sum_{k=1}^{m-1} n_k + 1}}{1 - p_m} > 0.$$

Let $a_i = (\ln(p_i))^2 \frac{p_i^{n_i+1}}{1-p_i}$ for all $i \in \{1, 2, \cdots m-1\}$ and $a_m = (\ln(p_m))^2 \frac{p_m^{n-\sum_{k=1}^{m-1} n_k+1}}{1-p_m}$. The second order derivatives can be rewritten as

$$\frac{\partial^2 C}{\partial n_i^2}(n_1, n_2, \cdots, n_{m-1}) = a_i + a_m \qquad \text{and}$$
$$\frac{\partial^2 C}{\partial n_i \partial n_j}(n_1, n_2, \cdots, n_{m-1}) = a_m.$$

As a consequence, the Hessian matrix of C, denoted $H(C)$, can be decomposed in

$$H(C) = D + a_m U_{m-1}$$

where $D = \text{diag}(a_1, a_2, \cdots, a_{m-1})$ and U_{m-1} is the square matrix of order $m-1$ with all entries equal to 1. Matrix D is clearly positive definite because all the a_i are strictly positive (we consider only non degenerated systems, where $\gamma \neq 0$ and $\mu \neq 0$). Let us show that U_{m-1} is positive-semidefinite. Let $x = (x_1, x_2, \cdots, x_{m-1})$ be a non zero column vector, then

$$\begin{aligned}
{}^t x U_{m-1} x &= \begin{bmatrix} x_1 & x_2 & \cdots & x_{m-1} \end{bmatrix} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{m-1} \end{bmatrix} \\
&= \begin{bmatrix} x_1 & x_2 & \cdots & x_{m-1} \end{bmatrix} \begin{bmatrix} \sum_{i=1}^{m-1} x_i \\ \sum_{i=1}^{m-1} x_i \\ \vdots \\ \sum_{i=1}^{m-1} x_i \end{bmatrix} \\
&= \left(\sum_{i=1}^{m-1} x_i \right)^2 \geq 0.
\end{aligned}$$

Then for all $x \neq 0$

$$\begin{aligned}
{}^t x H(C) x &= {}^t x (D + a_m U_{m-1}) x \\
&= \underbrace{{}^t x D x}_{>0} + a_m \underbrace{{}^t x U_{m-1} x}_{\geq 0} > 0.
\end{aligned}$$

To conclude, $H(C)$ is definite positive and consequently C is strictly convex. □

The same result can be proved for IBS, again the proof is omitted.

Property 5. *If the same assumptions than in Property 4 are made, except that the impatience is IBS, then the real valued cost function for IBS $\tilde{C}(n_1, n_2, \ldots, n_{m-1})$ is strictly*

convex in any direction.

The convexity of the objective function allows to use efficient techniques to find the minimum of the real valued function. However, even if the optimal of this function could be found in polynomial time, it is still difficult to find the minimum integer point of this function. Rounding the real valued solution is difficult; there are 2^{m-1} such roundings. Then enumerative methods, like branch and bound, must be used in order to find this point. We did not investigate further this question, since the technical needs to find the real valued solution is beyond the scope of this thesis. Nevertheless, for the problem involving only two machines, the solution is given by the following theorem for a problem with IES.

Theorem 16. *Consider a static scheduling problem with IES, where n identical jobs have to be processed on 2 machines. Each job has an exponentially distributed due date with rate γ, and processing time on machine i is exponentially distributed with rates μ_i (without loss of generality, let $\mu_1 \geq \mu_2$).*

An optimal schedule for this problem is to process $n_1^* = \arg\min\{C(\lfloor n_1' \rfloor), C(\lceil n_1' \rceil)\}$ jobs on machine 1 and $n - n_1^*$ on machine 2, where

- *$C(n_1)$ is the value of the schedule where n_1 jobs are processed on machine 1 and $n - n_1$ on machine 2 ($n = n_1 + n_2$);*
- *$p_i = \mu_i/(\mu_i + \gamma)$ and*

$$n_1' = \frac{1}{\ln(p_1 p_2)} \ln\left[\frac{\ln(p_2)}{\ln(p_1)} \times \frac{1-p_1}{1-p_2} \times \frac{p_2^{n+1}}{p_1}\right].$$

Proof. Denote $C_i(n)$ the cost of scheduling n jobs on machine i. From Corollary 2 (page 52):

$$C_1(n_1) = n_1 - \sum_{k=1}^{n_1} p_1^k$$

$$C_2(n - n_1) = (n - n_1) - \sum_{k=1}^{n-n_1} p_2^k.$$

Then the total cost is expressed as

$$\begin{aligned} C(n_1) &= n - \sum_{k=1}^{n_1} p_1^k - \sum_{k=1}^{n-n_1} p_2^k \\ &= n - p_1 \frac{1-p_1^{n_1}}{1-p_1} - p_2 \frac{1-p_2^{n-n_1}}{1-p_2} \\ &= n - \frac{p_1}{1-p_1} - \frac{p_2}{1-p_2} + \frac{p_1^{n_1+1}}{1-p_1} + \frac{p_2^{n-n_1+1}}{1-p_2}. \end{aligned}$$

Now we assume that n_1 can take real values, the first and second order derivatives are:

$$\frac{dC(n_1)}{dn_1} = \ln(p_1)\frac{p_1^{n_1+1}}{1-p_1} - \ln(p_2)\frac{p_2^{n-n_1+1}}{1-p_2} \quad \text{and}$$
$$\frac{d^2C(n_1)}{dn_1^2} = \ln(p_1)^2\frac{p_1^{n_1+1}}{1-p_1} + \ln(p_2)^2\frac{p_2^{n-n_1+1}}{1-p_2} > 0.$$

Hence the function is strictly convex and its minimum obtained for n_1' such that

$$\frac{dC(n_1')}{dn_1} = 0 \Leftrightarrow (p_1p_2)^{n_1'} = \frac{\ln(p_2)}{\ln(p_1)} \times \frac{1-p_1}{1-p_2} \times \frac{p_2^{n+1}}{p_1}$$
$$\Leftrightarrow n_1' = \frac{1}{\ln(p_1p_2)} \ln\left[\frac{\ln(p_2)}{\ln(p_1)} \times \frac{1-p_1}{1-p_2} \times \frac{p_2^{n+1}}{p_1}\right].$$

It follows that the integer valued function takes its minimum either in $\lfloor n_1' \rfloor$ or in $\lceil n_1' \rceil$. □

A similar theorem can be proved for problems with IBS, considering the expression of the cost of a schedule with IBS in Corollary 2 (page 52). The proof is omitted.

Theorem 17. *Consider a static scheduling problem with IBS, where n identical jobs have to be processed on 2 machines. Each job has an exponentially distributed due date with rate γ, and processing time on machine i is exponentially distributed with rates μ_i (without loss of generality, let $\mu_1 \geq \mu_2$).*

An optimal schedule for this problem is to process $n_1^* = \arg\min\{C(\lfloor n_1' \rfloor), C(\lceil n_1' \rceil)\}$ jobs on machine 1 and $n - n_1^*$ on machine 2, where

- *$C(n_1)$ is the value of the schedule where n_1 jobs are processed on machine 1 and $n - n_1$ on machine 2 ($n = n_1 + n_2$);*
- *$p_i = \mu_i/(\mu_i + \gamma)$ and*

$$n_1' = \frac{1}{\ln(p_1p_2)} \ln\left[\frac{\ln(p_2)}{\ln(p_1)} \times \frac{1-p_1}{1-p_2} \times p_2\right].$$

With Theorem 16 and Theorem 17, we show that an optimal solution for the problem of scheduling n identical jobs on two exponential machines can be given in constant time. More generally, these two theorems can be extended to the case of a fixed number of machines.

5.1.3 Multi-class

In the literature, only IES problems have been investigated. Emmons and Pinedo (1990) prove that when the m parallel machines have identically distributed processing times, the optimal static schedule for a problem with due dates can be given by solving an assignment problem. Such a schedule is represented in Figure 20. In this section, we adapt this result for IBS.

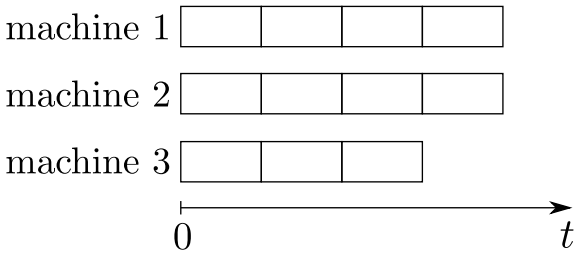


Figure 20: Static scheduling on 3 identical machines

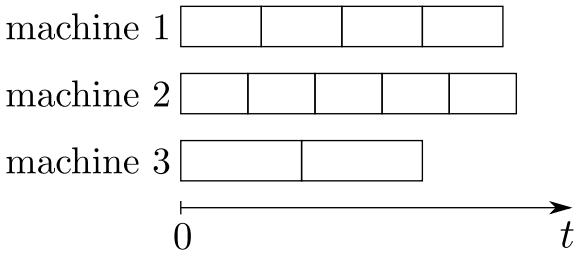


Figure 21: Static scheduling on 3 parallel machines

Theorem 18. *Consider the problem of scheduling n jobs on m parallel machines with IBS, where job j has a general and identically distributed processing time, $X_j \sim F_X$, an exponentially distributed patience time with rate γ_j, and a weight w_j: $Pm \mid X_j \sim F_X, D_j \sim \exp(\gamma_j) \mid E(\sum w_j \tilde{U}_j)$. This problem can be solved in polynomial time (assuming that Laplace transforms can be computed in $O(1)$) by solving an assignment problem with the following costs : if job j is processed in position k the cost is $w_j(1 - \mathcal{L}\{f_X\}(\gamma_j)^{\lfloor k \rfloor})$.*

Proof. With IBS, any job that is processed in the first position on any machine ends on time. The cost of the other jobs is given by Corollary 2. Since the processing times are identically distributed and that we consider the static problem, with impatience but without abandonment, the cost induced by a job does only depend on its position in the schedule and not on the previous jobs. Hence, we only consider the n first positions on the m machines. This ends the proof of Theorem 18. □

However, these results (for IES and IBS) can be generalized assuming that each machine has its own identically distributed processing time needed to process a job, and that this processing time does not depend on the job being processed. Such a schedule is represented in Figure 21. To the best of our knowledge, this problem has never been studied in the literature. In this setting, the cost induced by a job j in position k on machine i does not depend on the jobs previously processed either and the following theorem holds.

Theorem 19. *Consider the problem of scheduling n jobs on m parallel machines with IBS, where machine i needs a general and identically distributed processing time to process a job j, $X_j \sim F_{X_i}$, a job j has an exponentially distributed patience time with rate γ_j, and a weight w_j. This problem can be solved in polynomial time by solving a minimum cost flow problem on the graph $G = (V = (V_1, V_2) \cup \{s, t\}, A)$, as given in Figure 22, where:*

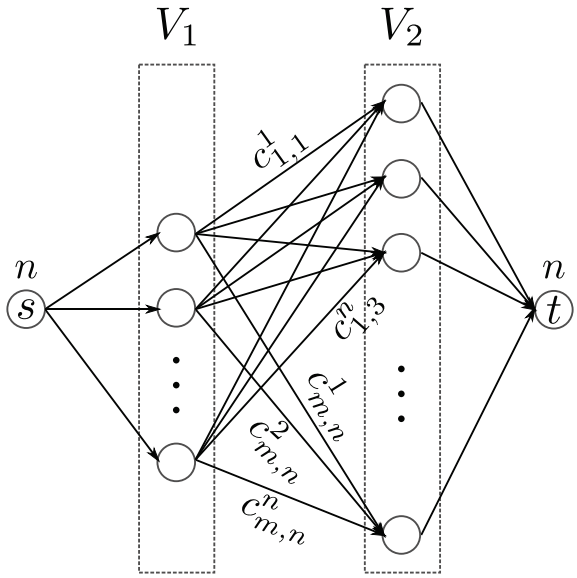


Figure 22: Representation of the n tasks m machines problem as a minimum cost flow problem

- $|V_1| = n$ *and* $|V_2| = nm$*;*
- s *is the source,* t *the sink and the imposed value of the flow is* n*;*
- *there are* n *arcs from* s *to any vertex in* V_1*,* nm *arcs from any vertex in* V_1 *to any vertex in* V_2 *and* nm *arcs from any vertex in* V_2 *to* t*;*
- *all arcs have capacity 1;*
- *every arc* $(v_j, v_{i,k})$*,* $v_j \in V_1$ *and* $v_{i,k} \in V_2$*, has weight* $c^j_{i,k} = w_j(1 - \mathcal{L}\{f_{X_i}\}(\gamma_j)^{k-1})$*, all other arcs have a cost equal to zero.*

If the optimal flow gets through arc $(v_j, v_{i,k})$*, then Job* j *is processed in position* i *on machine* k*.*

Proof. For all $j \in \{1, 2, \ldots, n\}$, $v_j \in V_1$ ($|V_1| = n$) represents Job j and for all $i \in \{1, 2, \ldots, m\}$ and $k \in \{1, 2, \ldots, n\}$, $v_{i,k} \in V_2$ ($|V_2| = nm$) represents position k on machine i. The unitary capacities and the imposed flow of value n ensure that arcs used by the flow form a matching between V_1 and V_2. The cost $c^j_{i,k} = w_j(1 - \mathcal{L}\{f_{X_i}\}(\gamma_j)^{k-1})$ is the cost generated by Job j when processed on machine i in position n. Hence, if the flow gets through arc $(v_j, v_{i,k})$ which corresponds to the assignment of job j to machine i in position k, then $c^j_{i,k}$ is added to the value of the objective function. And minimum cost flow problems can be solved in polynomial time as well (e.g. Ahuja et al. 1993). □

5.2 Non preemptive dynamic scheduling policies for problems with abandonments

We first present a property showing that in some cases, the parallel machine problem can be reduced to a single machine problem, and solved using techniques of Chapter 4. (In particular, if the processing times are exponentially distributed with rate μ_i on machine i, the problem is equivalent to a single machine problem with rate $\sum_{i=1}^{m} \mu_i$. We will not discuss further this problem.)

Property 6. *Consider a dynamic scheduling problem on m machines either with IES or with IBS, when preemption as well as jockeying between machines are allowed. And assume that the time needed to process a job depends only on the machine on which the job is processed and not on the job itself. If the instance of this problem can be solved on a single machine (e.g. Theorem 11 page 67 or Theorem 12 page 68 can be applied), then the parallel machines problem is solved following the same priority of execution.*

Proof. Let us consider a dynamic scheduling problem on m machines either with IES or with IBS, when preemption as well as jockeying between machines are allowed. In such a case, the single machine priority list is build, and the m jobs at the top of this list are processed on the m machines, the fastest machine being assigned to the job in top of the list. Whenever an event occurs (end of execution or abandonment of a job) every job is moved up on a faster machine (allowed by jockeying and preemption) and the first job waiting in the queue is processed on the slowest machine. □

Now consider a dynamic scheduling problem on m machines with IES, when neither preemption nor jockeying between machines are allowed. Let us assume that there is exactly one job to be processed in the system. It is clear that this job will be processed on the fastest machine in order to minimize the probability that this job ends late. Now assume that there are exactly two identical jobs to be processed. Again, it is clear that one job should be assigned to the fastest machine. The decision to make for the second job is whether it should be assigned to the second fastest machine, or wait for the first machine to be freed. The decision of waiting can be interesting to make, especially when the probability that the job ends late on the slower machine is greater than the probability that this job ends late while it waits in the queue or while it is in process on the fastest machine. This situation can arise when the execution time on the second machine is very slow compared to the execution time on the first machine and when there are few jobs waiting for the first machine. In this problem there is clearly a trade-off between an immediate slow process and a fast but delayed process. Slow machines can be kept idling while there are jobs available for processing in the system. This problem is investigated in Section 5.2.1.

For dynamic scheduling problems on m machines with IBS, when neither preemption nor jockeying between machines are allowed, decisions to make are slightly different. For

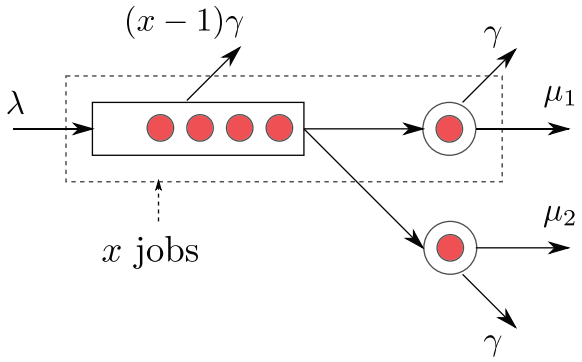


Figure 23: Problem with a single class of jobs and with two machines of different exponential rates

example, when jobs are identical, all machines should be used, because once a job is processed it is on time. However, when there are different classes of jobs, the decisions to make are the same as for IES problems.

5.2.1 Single class, multi-machine

Consider a scheduling problem with a single class of jobs. Jobs become available according to a Poisson process of rate λ and wait for process in a queue of capacity M. If a job becomes available while the queue already contains M jobs, the new job is rejected without additional cost. Jobs that are in the system are subject to impatience to the end of service, their patience time is exponentially distributed with rate γ. We assume that the system incurs an abandonment cost w for each job that becomes impatient before the end of its service. Since there is only one class of job, we could consider without loss of generality that this cost is equal to 1, but it makes the formulation clearer. The system contains two machines with exponentially distributed processing times of rate μ_1 and μ_2, preemption is not allowed. Without loss of generality, we assume that $\mu_1 > \mu_2$. The objective of the decision maker is to decide, at any point in time, to process a job (or not) and to decide on which machine it will be processed, in order to minimize the long run expected abandonment costs. This system is represented in Figure 23.

First, note that the fastest machine is always busy if jobs are pending. Suppose that there is one job in the system. Since the system is markovian, once a decision is made it is not modified until the next event. Consequently, if it has been decided not to process the job, it will abandon with probability 1. If it has been decided to process the job on machine i, it will abandon with probability $\gamma/(\mu_i + \gamma)$. Hence, the job is assigned to the fastest server.

With these considerations, the state of the system is given by $(x(t), u(t))$ at time t, where $x(t) \in \mathbb{N}$ is the number of jobs waiting for service in the queue plus the job which is processed on machine 1 (there are $x-1$ jobs waiting and one job in process), and $u(t) \in \{0, 1\}$ denote the state of the second server. If $u(t) = 1$ a job is processed on machine

2 and if $u(t) = 0$ machine 2 is free. Since all durations are exponentially distributed, the system is markovian and does not depend on the time index. Therefore the state of the system is fully described by (x, u). Let $e_x = (1, 0)$ and $e_u = (0, 1)$ be the basis vectors. The size of the state space for this system is $2M$. (The size of the queue is limited to M for tractability reasons; however it is a reasonable approximation of this problem as M is allowed to be arbitrarily large.)

Following Lippman (1975), the long run expected abandonment costs for such system verifies

$$v^*(x, u) = \frac{1}{C} T v^*(x, u),$$

where the meta-operator T is defined as

$$Tv(x, u) = T_d \circ (T^1_{prod} v(x, u) + T^2_{prod} v(x, u) + \gamma T_{ab} v(x, u) + \lambda T_{arr} v(x, u))$$

and $C = \alpha + \mu_1 + \mu_2 + (M + 1)\gamma + \lambda$, with α the discount factor. Since the state space is finite, the structure of the optimal policy in the average cost case can be derived from properties of the discounted cost case by letting α tend to 0 (Ross 1970).The abandonment operator T_{ab} consider abandonments of jobs that are waiting for service and abandonment of the job in service on machine 1 and is defined by

$$T_{ab} = \begin{cases} xv(x-1, u) + (M - x)v(x, u) + wx & \text{if } x > 0 \\ Mv(x, u) & \text{if } x = 0. \end{cases}$$

The production operators are defined as follows (the production operator for machine 2 takes into account the abandonment of the job being processed)

$$T^1_{prod} = \begin{cases} \mu_1 v(x-1, u) & \text{if } x > 0 \\ \mu_1 v(x, u) & \text{if } x = 0, \end{cases}$$

$$T^2_{prod} = \begin{cases} (\mu_2 + \gamma)v(x, 0) + w\gamma & \text{if } u = 1 \\ (\mu_2 + \gamma)v(x, u) & \text{if } u = 0. \end{cases}$$

And the arrival operator is defined by

$$T_{arr} = \begin{cases} v(x+1, u) & \text{if } x < M \\ v(x, u) & \text{if } x = M. \end{cases}$$

The decision maker can decide to turn on machine 2 or not, this decision is described by operator T_d. Given that the system is markovian, this decision has only to be made after an event, therefore this operator is composed with the sum of the other operators. The

decision operator is defined by

$$T_d = \begin{cases} \min\{v(x,u), v(x-1,1)\} & \text{if } u = 0 \text{ and } x > 0 \\ v(x,u) & \text{otherwise.} \end{cases}$$

Since it is always possible to rescale time by letting $C = 1$, the optimality equation verifies $v^*(x,u) = Tv^*(x,u)$. Note that like in Chapter 4, in this problem an abandonment cost w is equivalent to a cost $w\gamma$ paid per unit of time for each job that is in the system. Let $c(x,u) = w\gamma(x+u)$ be the cost function corresponding to the abandonment costs, then $Tv(x,u)$ can be rewritten as

$$Tv(x,u) = c(x,u) + T_d \circ (\mu_1 T^1_{prod} v(x,u) + (\mu_2 + \gamma) T^2_{prod} v(x,u) + \gamma T_{ab} v(x,u) + \lambda T_{arr} v(x,u)).$$

The production operator for machine 2 and the abandonment operator are rewritten as

$$T^2_{prod} = (\mu_2 + \gamma) v(x,0)$$

$$T_{ab} = \begin{cases} xv(x-1,u) + (M-x)v(x,u) & \text{if } x > 0 \\ Mv(x,u) & \text{if } x = 0. \end{cases}$$

Hajek (1984) and then Koole (1995) consider a special case of this problem. They considered a problem without abandonments on an unbounded state space. Thus operator T_{ab} is not used to describe their system and the arrival operator does not depend on the state on which it is applied. The authors prove that the optimal value function respects the three following properties, if the cost function is (e.g.) linear in x and in u,

Property 1: $\Delta_{e_x} v(x,u) \geq 0, \forall x \geq 0, \forall u \in \{0,1\}$
with $\Delta_{e_x} v(x,u) = v(x+1,u) - v(x,u)$;

Property 2: $\Delta_{e_x,e_u} v(x,0) \geq 0, \forall x \geq 0$
with $\Delta_{e_x,e_u} v(x,0) = \Delta_{e_x}\Delta_{e_u} v(x,0) = v(x+1,1) - v(x+1,0) - v(x,1) + v(x,0)$;

Property 3: $\Delta_{e_x,-e_x+e_u} v(x+1,0) \leq 0, \forall x \geq 0$
with $\Delta_{e_x,-e_x+e_u} v(x,0) = v(x+1,1) - v(x+2,0) - v(x,1) + v(x+1,0)$.

Property 1 implies that the optimal cost is increasing in the number of jobs waiting in the queue. Property 2 corresponds to supermodularity. Property 3 implies that the optimal control policy is of a threshold type. If $v(x-1,1) - v(x,0) \leq 0$, then a job should be placed on machine 2 for processing (see the decision operator T_d), and Property 3 states that this quantity is decreasing as the number of jobs in the queue increase. Consequently if x is large enough, machine 2 should be used and the optimal policy has a threshold.

The structure of the optimal policy mentioned above, seems to be reasonable even if the system is subject to abandonments. Koole (1995) proved that the operators T_d, T^1_{prod}

and T^2_{prod} propagate the three properties. In order to prove that the optimal control policy for the same problem with abandonments has a threshold, operators T_{arr} and T_{ab} should prove to propagate the three properties as well.

The optimal value for this problem respects Property 1 and Property 2, the proof is given in Appendix D.1 (page 139).

However we provide a numerical example where Property 3 is not respected. The instance that we consider has the following parameters : $M = 100$, $\mu_1 = 3$, $\mu_2 = 0.5$, $\lambda = 2$ and $\gamma = 1$. This instance is solved to optimality using value iteration with a precision of 10^{-8} (as in Section 4.6, we used the modified value iteration algorithm in order to be able to compute the average costs). If the abandonment cost is equal to 1, the optimal control policy for this instance has a long run average cost of 1.19 and it is optimal to turn on machine 2 if and only if there are at least 7 jobs in the system. Although there is a visual evidence that the optimal policy for this instance is of a threshold type, it does not ensure that Property 3 holds. Figure 24 represents the value of $v(x-1,1) - v(x,0)$ that we truncated vertically for visual insights. This figure is consistent with the presence of a threshold, because when there are 7 jobs in the queue, the value is greater than 0. But the structure of the optimal policy for the problem with abandonment is not as regular as for the model studied by Koole (1995). In his paper, the system respects Property 3 ($\Delta_{e_x,-e_x+e_u} v(x+1,0) \leq 0$), which guarantees the existence of a threshold. The values of $\Delta_{e_x,-e_x+e_u} v(x+1,0)$ are given in Figure 25 for the system under consideration. This value is negative when machine 2 is turned off, but becomes positive as machine 2 is turned on (an untruncated version of these figures are given in Appendix D.2 page 141). However, it seems from these two figures that even if $\Delta_{e_x,-e_x+e_u} v(x+1,0)$ is positive, the value of $v(x-1,1) - v(x,0)$ tends to zero but remains negative.

In this system, the costs are induced by the number of jobs in the system. Consequently, the fastest the system is cleared, the less expensive the policy is. That is why there is a threshold, because using the second machine is only interesting when there are enough jobs in the system. However, as the number of jobs increases, the rate at which abandonments occur increases. And abandonments clear the system fast enough so that the value of $v(x-1,1) - v(x,0)$ decrease.

The avenue for future research for this problem is to provide a property, or a set of properties, which can describe this behavior. In our belief it will bring useful insights on systems with abandonments as it appears in such simple system.

5.2.2 Multi-class, single machine

In the previous section, we presented results on scheduling a single class of jobs on several machines without preemption, and we presented evidences that this problem is difficult to solve. In this section we discuss the problem of scheduling two classes of jobs on a single machine with abandonments. We consider the same model as in Chapter 4 but preemption is not authorized anymore. Not authorizing preemption implies that the

Figure 24: Value of $v(x-1,1)-v(x,0)$ for the instance $M=25$, $\mu_1=3$, $\mu_2=0.5$, $\lambda=2$ and $\gamma=1$

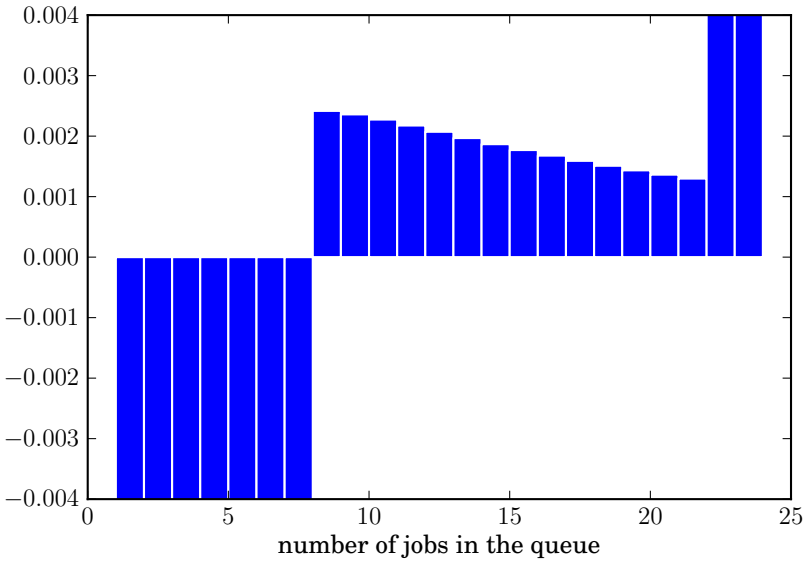


Figure 25: Value of $\Delta_{e_x,-e_x+e_u}v(x+1,0)$ for the instance $M=25$, $\mu_1=3$, $\mu_2=0.5$, $\lambda=2$ and $\gamma=1$

optimal scheduling policy may idle: if only low priority jobs are in the queue, it may worth not processing them but waiting for a higher priority job.

Let us consider the case of two jobs only. Job 2 is available at time 0 and Job 1 becomes available after an exponentially distributed amount of time with rate λ. The parameters of the jobs are

- λ_1, the arrival rate of Job 1;
- μ_1, μ_2, the processing rates;
- γ_1, γ_2, the abandonment rates;
- w_1, w_2, the abandonment costs.

We compare two schedules without preemption:

- $\{2,1\}$: Job 2 is processed first, then Job 1 is processed if it has not abandoned yet;
- $\{1\}$: the machine is kept idling while Job 1 is not available, when Job 1 becomes available it is processed right away and Job 2 is not processed after Job 1 left the system, even if Job 2 has not abandoned yet.

Schedule $\{1\}$ mimics the behavior of a dynamic priority rule which does never process jobs from class 2, but waits for new jobs from class 1 to become available. We compare the cost of these two schedules in order to derive some insights on what makes an optimal policy idle while there are jobs waiting for service.

Problems with IBS

$$C(\{2,1\}) = \frac{w_1\gamma_1}{\mu_2+\gamma_1} \times \frac{\lambda_1}{\lambda_1+\mu_2}$$
$$C(\{1\}) = w_2$$

It worths waiting for Job 1 if

$$C(\{2,1\}) - C(\{1\}) \geq 0 \Leftrightarrow \frac{w_1\gamma_1}{\mu_2+\gamma_1} \times \frac{\lambda_1}{\lambda_1+\mu_2} \geq w_2.$$

It is all the more interesting to wait for Job 1, that

1. w_1 is greater than w_2;
2. Job 1 becomes available quickly compared to the processing time of Job 2;
3. Job 1 becomes impatient quickly compared to the processing time of Job 2.

From this equality, it is necessary but not sufficient that $w_1 > w_2$ to wait for Job 1 while Job 2 waits for process.

Problems with IES

$$C(\{2,1\}) = \frac{w_1\gamma_1}{\mu_1+\gamma_1}\left(1+\frac{\lambda_1\mu_1}{(\lambda_1+\mu_2+\gamma_2)(\mu_2+\gamma_1+\gamma_2)}\right)+\frac{w_2\gamma_2}{\mu_2+\gamma_2}$$
$$C(\{1\}) = \frac{w_1\gamma_1}{\mu_1+\gamma_1}+w_2$$

It worths waiting for Job 1 if

$$C(\{2,1\})-C(\{1\})\geq 0 \Leftrightarrow \frac{w_1\gamma_1}{\mu_1+\gamma_1}\times\frac{\lambda_1\mu_1}{(\lambda_1+\mu_2+\gamma_2)(\mu_2+\gamma_1+\gamma_2)}\geq\frac{w_2\mu_2}{\mu_2+\gamma_2}$$
$$\Leftrightarrow w_1\frac{\mu_1}{\mu_1+\gamma_1}\times\frac{\gamma_1}{\mu_2+\gamma_1+\gamma_2}\times\frac{\lambda_1}{\lambda_1+\mu_2+\gamma_2}\geq w_2\frac{\mu_2}{\mu_2+\gamma_2}.$$

It is all the more interesting to wait for Job 1, that

1. w_1 is greater than w_2;
2. Job 1 becomes available quickly compared to the time that Job 2 spends in the system (Job 2 leaves the system even after an abandonment or after the end of its process);
3. Job 1 becomes impatient quickly compared to the time that Job 2 spends in the system;
4. Job 1 has a higher chance than Job 2 to end on time.

As in problems with preemption, IES and IBS problems have comparable behavior. The first three remarks lead to the same insights: the weight of Job 1 must be greater than the weight of Job 2 and Job 1 must become available/impatient quickly compared to the time that Job 2 spends in the system if it is processed first. In IES problems the probability that the process end before jobs get impatient is taken into account. When the system is markovian and jobs arrive according to a Poisson process, the scheduling problem is a Markov decision process for which the optimal non preemptive dynamic policy can be approximated by value iteration. The dynamic programming formulation is given in Appendix D.3 (page 142). We compute the optimal policy for the instance with IES $\lambda_1 = \lambda_2 = 1$, $\gamma_1 = \gamma_2 = 1$, $\mu_1 = 3$, $\mu_2 = 0.5$, $w_1 = 1$ and $w_2 = 0.5$, with precision 10^{-8}. The optimal policy without preemption gives strict priority to jobs from class 1. When there are no jobs from class 1 and jobs from class 2 are waiting for service, the machine is kept idling. Such an idling policy is represented in Figure 26. Whereas the optimal policy with preemption does not keep the machine idle. With preemption, the long run expected abandonment cost is 0.69 and without preemption the cost is a bit higher 0.79. It is not always the case that the optimal non preemptive dynamic policy keeps the machine idle. For example, the instance $\lambda_1 = \lambda_2 = 1$, $\gamma_1 = \gamma_2 = 1$, $\mu_1 = 3$, $\mu_2 = 0.5$, $w_1 = 1.5$ and $w_2 = 1$ leads to a non idling policy as represented in Figure 27. (As discussed above,

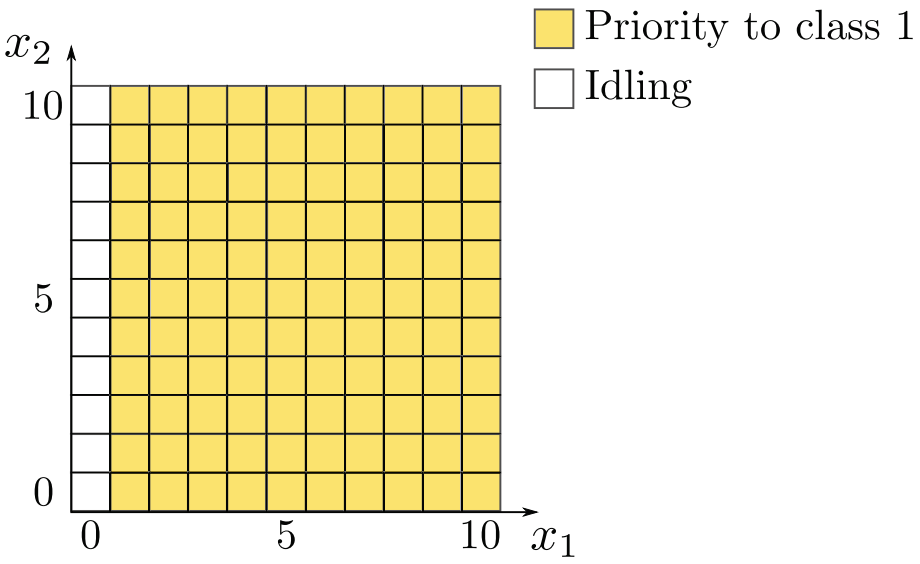


Figure 26: Example of idling policy

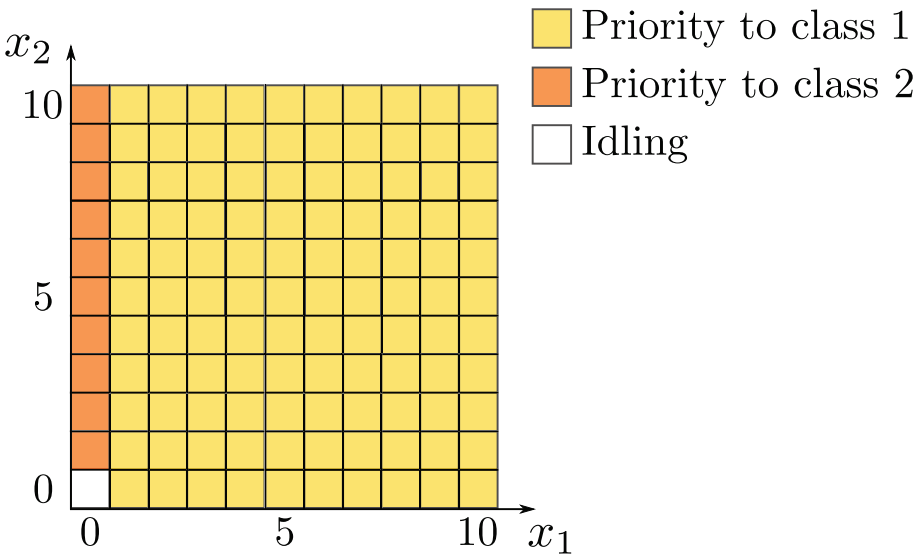


Figure 27: Example of non idling policy

this example shows that $w_1 > w_2$ is not sufficient to wait for jobs from class 1.) For this instance again, when preemption is allowed, the optimal cost is 1.23 and when it is not the optimal cost is 1.42. This difference is due to the fact that in the preemptive policy, when a job from class 1 becomes available while the machine is processing a job from class 2, the optimal policy switches from class 2 to class 1, which reduces the overall costs.

However, to the best of our knowledge, it is an open question to be able to give conditions that guarantees that idling is (not) optimal given the parameters of an instance.

Chapitre 6

Conclusion et perspectives de recherche

Contributions

Dans cette thèse, nous avons étudié des modèles décrivant l'impatience dans des systèmes de production ou de service. Nous avons modélisé les problèmes étudiés comme des problèmes d'ordonnancement avec des dates d'échéance, où ces dernières représentent l'impatience des tâches ou des clients. Nous avons mis en place un cadre d'étude rendant compte des différentes nuances que soulèvent ces problèmes. Nous avons par exemple fait la distinction entre l'impatience, le fait d'avoir attendu trop longtemps, et l'abandon, le fait de quitter le système suite à l'impatience. Nous avons aussi pris soin de distinguer l'impatience jusqu'au début et jusqu'à la fin du service. Toutes ces nuances que nous avons utilisées traduisent des caractéristiques différentes que l'on retrouve dans ces systèmes. Selon les classes de politiques que l'on considère, ces nuances peuvent entraîner des modélisations différentes et par conséquent, elles peuvent aussi entraîner des politiques de service optimales différentes. Nous proposons aussi une étude numérique mettant en évidence ces différences et des extensions possibles pour ces problèmes.

En particulier, dans le chapitre 3, nous avons étudié un problème d'ordonnancement à une machine, avec impatience, dans la classe des politiques statiques sans préemption. Nous avons montré que pour le problème déterministe, les cas IES et IBS sont équivalents, mais les ordonnancements optimaux que nous proposons montrent bien que ce n'est pas le cas lorsque le problème est stochastique. L'originalité de nos travaux dans ce chapitre, tient au fait que l'on considère des problèmes plus généraux que ceux qui ont été étudiés jusqu'à présent dans la littérature. Par exemple, les problèmes étudiés par Pinedo (1983) ou par Boxma et Forst (1986) ne considèrent que le cas IES. De plus, nous avons présenté nos résultats pour les deux types d'impatience, IES et IBS.

Dans le chapitre 4, nous avons étudié un problème d'ordonnancement à une machine, avec abandon, dans la classe des politiques dynamiques avec préemption. Nous avons dé-

montré que lorsque la variable aléatoire modélisant la durée d'impatience suit une distribution exponentielle, alors les coûts de mise en attente (pour chaque tâche, un coût est payé par unité de temps passé dans le système) et les coûts d'abandon sont équivalents. Ceci est un résultat plus général que celui proposé précédemment par Atar et al. (2010). En effet, ils considèrent un système markovien (toutes les variables aléatoires sont exponentielles) alors que dans notre cas, seules les durées d'impatience sont exponentielles, les autres durées sont des variables aléatoires quelconques. Nous avons aussi donné une politique stricte optimale plus générale que celle proposée par Down et al. (2011). En effet, ces auteurs proposent une règle de priorité stricte optimale pour deux classes de clients alors que nous considérons un nombre quelconque de classes. De plus, notre résultat s'applique à un plus grand nombre d'instances. L'étude théorique que nous avons menée sur ce problème nous a aussi permis de proposer une nouvelle heuristique qui, selon notre étude expérimentale, est plus performante que celles que l'on retrouve dans la littérature. Cette étude numérique, nous a aussi permis d'exhiber l'influence du type d'impatience et de certains paramètres sur l'efficacité de résolution des heuristiques.

Enfin, dans le chapitre 5, nous avons discuté des extensions naturelles des problèmes traités dans les chapitres 3 et 4. Dans la classe des politiques statiques, nous avons étendu au cas IBS un résultat connu pour IES, lorsque n tâches sont à exécuter sur m machines parallèles. Ce résultat a été adapté au cas avec m machines parallèles, pour lesquelles la durée d'exécution d'une tâche dépend uniquement de la machine sur laquelle elle est exécutée. Nous discutons aussi du problème avec une seule classe de tâches sur m machines parallèles, qui lui reste un problème a priori difficile à résoudre efficacement pour un nombre quelconque de machines. Dans le cas dynamique, nous avons discuté du problème sans préemption avec une classe de tâches, dans un système markovien à deux machines ayant des taux différents. Le problème sans impatience correspondant a été étudié par Hajek (1984) et Koole (1995). Ces auteurs donnent la structure de ce problème, modélisé à l'aide d'un processus de décision de Markov. Ils prouvent, entre autre, que l'on doit toujours utiliser la machine la plus rapide et que l'on met en marche l'autre machine, seulement si le nombre de tâches présentes dans le système dépasse un certain seuil. Lorsque les tâches sont sujettes à l'impatience, ce seuil peut encore être observé et nous donnons un exemple numérique le mettant en évidence. Nous montrons aussi que certaines caractéristiques de la structure de ce problème sont conservées, mais pas celle garantissant l'existence du seuil. Pour le problème avec deux classes de tâches sur une machine lorsque la préemption n'est pas autorisée, nous donnons des intuitions sur des situations pouvant rendre une politique avec temps morts optimale. Nous donnons aussi un exemple d'instance pour laquelle la politique optimale n'exécute jamais un type de tâches, même si elles sont les seules présentes dans le système.

Perspectives

En étudiant des problèmes d'ordonnancement avec impatience, et en modélisant cette impatience par des dates d'échéances, nous avons ouvert de nouvelles perspectives concernant les problèmes d'ordonnancement stochastiques avec dates d'échéances. En effet, au vu des résultats obtenus dans le chapitre 4, les techniques utilisées en ordonnancement sont tout à fait pertinentes pour modéliser et résoudre des problèmes difficiles, qui apparaissent surtout dans la littérature sur le contrôle optimal de files d'attente. De plus, les résultats que nous avons présentés dans cette thèses ouvrent aussi de nouvelles perspectives, permettant d'approfondir les connaissances sur les problèmes avec impatience.

Dans le chapitre 3, nous donnons des ordonnancements optimaux dans la classe des politiques statiques et ces résultats dépendent des distributions des différentes variables aléatoires. Une perspective serait de donner des résultats pour des distributions plus générales, en particulier pour le cas où les durées d'exécution et les dates d'échéance ne sont pas i.i.d. (cas traité dans le théorème 6, page 54). Les conditions que doivent respecter les variables aléatoires pour ce problème, sont proches de celles que l'on retrouve dans le corollaire donnant un ordonnancement dynamique optimal dans le chapitre 4 (corollaire 5). Or, ce corollaire est un cas particulier de deux théorèmes plus généraux, donnant eux aussi un ordonnancement dynamique optimal (théorème 11 et 12). Il serait donc pertinent d'étudier le cas du problème statique, afin de savoir si de telles conditions, plus générales, peuvent aussi être déduites. En parallèle, une étude numérique pourrait être menée, afin de vérifier l'efficacité des théorèmes donnant un ordonnancement dynamique optimal, lorsqu'ils sont utilisés comme heuristique pour le problème statique.

Dans le chapitre 4, nous donnons une conjecture pour le passage à l'horizon infini qu'il nous faut approfondir. L'intuition raisonnable que l'on pourrait avoir est que ce résultat soit juste, mais il reste des détails techniques à vérifier pour en compléter la preuve. Dans ce chapitre, nous donnons aussi une règle de priorité stricte optimale pour un problème d'ordonnancement dynamique avec abandon, lorsque la préemption est autorisée. En revanche, nous ne donnons aucun résultat lorsque la politique optimale a un seuil de priorité. En effet, aucun résultat n'existe, que ce soit sur l'existence de ce seuil, sa forme, sa position et encore moins son comportement face à des variations des valeurs des paramètres. Ce problème est encore totalement ouvert. Concernant l'étude numérique, il reste encore à comprendre pourquoi une politique stricte est optimale dans la majorité des cas et pourquoi la différence de coût entre la politique optimale à seuil et la meilleure des heuristiques strictes est si faible pour un problème avec IES. Un autre problème totalement ouvert est celui où la préemption n'est pas autorisée. Pour ce problème, une des perspectives de recherche est de trouver des conditions suffisantes garantissant que l'on n'exécute jamais une certaine classe de tâches.

Dans le chapitre 5, nous présentons des variantes et des extensions des problèmes étudiés dans les chapitres 3 et 4. Nous avons étudié le problème statique avec n tâches identiques

sur m machines. Nous avons donné une solution en temps constant pour le problème avec 2 machines, et une solution en temps exponentiel pour le problème à m machines. Une question ouverte est donc de savoir s'il existe un algorithme en temps polynomial pour résoudre ce problème à l'optimal. De plus, le problème avec n tâches sur m machines indépendantes est lui aussi un problème ouvert. Dans le cas déterministe, ce problème est $\mathcal{NP}$-difficile (Sitters 2001), cependant il peut exister des cas particuliers stochastiques qui se résolvent en temps polynomial. Qui plus est, tous les problèmes statiques présentés ont un point commun : les dates d'échéances sont des variables aléatoires exponentielles. Une autre question se pose alors, est-il possible de montrer des résultats en omettant cette hypothèse ? Enfin, dans la classe des politiques dynamiques, il reste à caractériser la structure de la politique optimale pour le problème avec 2 machines sans préemption. En effet, l'on observe la présence d'un seuil à partir duquel le serveur le plus lent est utilisé, mais il semble difficile de prouver ce comportement lorsque le problème est modélisé par un processus de décision de Markov.

La compréhension des phénomènes observés dans ces différents problèmes simples, de part leur structure, nous semble indispensable à la compréhension des impacts que génère l'impatience dans les problèmes d'ordonnancement ou de contrôle optimal de files d'attentes.

Bibliographie

Agrawala, A.K., Jr. Coffman, E.G., M.R. Garey, S.K. Tripathi. 1984. A stochastic optimization algorithm minimizing expected flow times on uniforn processors. *Computers, IEEE Transactions on* **C-33**(4) 351–356. doi:10.1109/TC.1984.1676440.

Ahuja, R.K., T.L. Magnanti, J.B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall.

Argon, N.T., S. Ziya, R. Righter. 2008. Scheduling impatient jobs in a clearing system with insights on patient triage in mass casualty incidents. *Probability in the Engineering and Informational Sciences* **22** 301–332.

Artalejo, J.R., V. Pla. 2009. On the impact of customer balking, impatience and retrials in telecommunication systems. *Computers and Mathematics with Applications* **57** 217–229.

Atar, R., C. Giat, N. Shimkin. 2010. The $c\mu/\theta$ rule for many-server queues with abandonment. *Operations Research* **58** 1427–1439. doi:10.1287/opre.1100.0826.

Baptiste, P. 1999a. An $O(n^4)$ algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. *Operations Research Letters* **24**(4) 175–180.

Baptiste, P. 1999b. Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine when processing times are equal. *Journal of Scheduling* **2** 245–252.

Benjaafar, S., M. Elhafsi. 2011. A production-inventory system with both patient and impatient demand classes. *IEEE Transactions on Automation Science and Engineering* **9** 148–159.

Benjaafar, S., J.-P. Gayon, S. Tepex. 2010. Optimal control of a production-inventory system with customer impatience. *Operations Research Letters* **38**(4) 267–272.

Blok, H., F. Spieksma, S. Bhulai. 2012. K competing queues with customer impatience: optimality of the μc rule by the smoothed rate truncation principle. *StochMod '12*. École Centrale Paris, France.

Boxma, O.J., F.G. Forst. 1986. Minimizing the expected weighted number of tardy jobs in stochastic flow shops. *Operations Research Letters* **5**(3) 119–126.

Boxma, O.J., D. Perry, W. Stadje. 2011. The $M/G/1 + G$ queue revisited. *Queueing Systems* **67** 207–220.

Brucker, P. 2007. *Scheduling Algorithms*. Springer.

Buyukkoc, C, P Varaiya, J Walrand. 1985. The $c\mu$ rule revisited. *Advances in applied probability* **17**(1) 237–238.

Carlier, J. 1981. Problème à une machine et algorithmes polynomiaux. *Questio* **5** 219–228.

Chang, C.S., X.L. Chao, M.L. Pinedo, R. Weber. 1992. On the optimality of LEPT and $c\mu$ rules for machines in parallel. *Journal of Applied Probability* **29**(3) 667–681. doi:10.2307/3214903.

Chang, CS, R Nelson, M.L. Pinedo. 1991. Scheduling 2 classes of exponential jobs on parallel processors - structural results and worst-case analysis. *Advances in Applied Probability* **23**(4) 925–944. doi:10.2307/1427684.

Cobham, Alan. 1954. Priority assignment in waiting line problems. *Operations Research* **2**(1) 70–76.

Cooper, Robert B. 1972. *Introduction to queueing theory*. Macmillan New York.

Dalal, Amy Csizmar, Scott Jordan. 2001. Improving user-perceived performance at a world wide web server. *Global Telecommunications Conference, 2001. GLOBECOM'01. IEEE*, vol. 4. IEEE, 2465–2469.

Dauzère-Pérès, S. 1995. Minimizing late jobs in the general one machine scheduling problem. *European Journal of Operational Research* **81** 134–142.

Dauzère-Pérès, S., M. Sevaux. 2003a. Genetic algorithms to minimize the weighted number of late jobs on a single machine. *European Journal of Operational Research* **151**(2) 296–306.

Dauzère-Pérès, S., M. Sevaux. 2003b. Using lagrangean relaxation to minimize the weighted number of late jobs on a single machine. *Naval Research Logistics* **50**(3) 273–288.

Down, D.G., G. Koole, M.E. Lewis. 2011. Dynamic control of a single-server system with abandonments. *Queueing Systems* **67** 63–90.

Economopoulos, A.A., V.S. Kouikoglou, E. Grigoroudis. 2011. The base stock/base backlog control policy for a make-to-stock system with impatient customers. *IEEE Transactions On Automation Science And Engineering* **8** 243–249.

Emmons, H., M.L. Pinedo. 1990. Scheduling stochastic jobs with due dates on parallel machines. *European Journal of operational research* **47**(1) 49–55.

Fan, B., Y. Sun, R. Chen, G. Tang. 2012. A revised proof of the optimality for the Kise-Ibaraki-Mine algorithm. *Optimization Letters* **6**(8) 1951–1955.

Gens, G.V., E.V. Levner. 1981. Fast approximation algorithm for job sequencing with deadlines. *Discrete Applied Mathematics* **3**(4) 313–318.

Graham, R.L., E.L. Lawler, J.K. Lenstra, A.H.G. Rimooy Kan. 1979. Optimization and approximation in deterministic sequencing and scheduling : a survey. *Annals of Discrete Mathematics* **5** 287–326.

Gross, Donald, Carl M Harris. 1998. *Fundamentals of queuing theory*. Wiley New York.

Hajek, Bruce. 1984. Optimal control of two interacting service stations. *Automatic Control, IEEE Transactions on* **29**(6) 491–499.

Harrison, J.M. 1975a. Dynamic scheduling of a multiclass queue: Discount optimality. *Operations Research* **23**(2) 270–282.

Harrison, J.M. 1975b. A priority queue with discounted linear costs. *Operations Research* **23**(2) 260–269.

Ibarra, O.H., C.E. Kim. 1978. Approximation algorithms for certain scheduling problems. *Mathematics of Operations Research* **3**(3) 197–204.

Iravani, F., B. Balcioglu. 2008. On priority queues with impatient customers. *Queueing Systems* **58**(4) 239–260. doi:10.1007/s11134-008-9069-6.

Jang, W. 2002. Dynamic scheduling of stochastic jobs on a single machine. *European Journal of Operational Research* **138**(3) 518–530.

Jang, W., C.M. Klein. 2002. Minimizing the expected number of tardy jobs when processing times are normally distributed. *Operations Research Letters* **30**(2) 100–106.

Kallmes, M.H., D. Towsley, C. Cassandras. 1989. Optimality of the last-in-first-out (lifo) service discipline in queueing systems with real-time constraints. Tech. rep., University of Massachusetts.

Kendall, D.G. 1953. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain. *Annals of Mathematical Statistics* **3** 338–354.

Kise, H., T. Ibaraki, H. Mine. 1978. A solvable case of one-machine scheduling problem with ready and due times. *Operations Research* **26**(1) 121–126.

Kofman, E., S.A. Lippman. 1981. An $M/M/1$ dynamic priority queue with optional promotion. *Operations Research* **29**(1) 174–188. doi:{10.1287/opre.29.1.174}.

Koole, Ger. 1995. A simple proof of the optimality of a threshold policy in a two-server queueing system. *Systems and Control Letters* **26**(5) 301–303.

Koole, Ger. 2006. Monotonicity in markov reward and decision chains: Theory and applications. *Foundations and Trends in Stochastic Systems* **1**(1) 1–76.

Kuhfittig, P.K.F. 1978. *Introduction to the Laplace Transform*. PLENUM Publishing Co., N.Y.

Labetoulle, J., E.L. Lawler, J.K. Lenstra. 1984. *Preemptive Scheduling of Uniform Machines subject to Release dates*. Academic Press, NY, 245–261.

Lawler, E.L. 1990. A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. *Annals of Operations Research* **26** 125–133.

Lawler, E.L. 1994. Knapsack-like scheduling problems, the Moore-Hodgson algorithm and the tower of sets property. *Mathematical and Computer Modelling* **20**(2) 91–106.

Lawler, E.L., M.J. Moore. 1969. A functional equation and its application to resource allocation and sequencing problems. *Management Science* **16** 77–84.

Lenstra, J.K., A.H.G.R. Kan, P. Brucker. 1977. Complexity of machine scheduling problems. *Annals of Discrete Mathematics* **1** 343–362.

Li, S., Z.-L. Chen, G. Tang. 2010. A note on the optimality proof of the Kise-Ibaraki-Mine algorithm. *Operations Research* **58**(2) 508–509.

Lillo, R.E. 2001. Optimal control of an $M/G/1$ queue with impatient priority customers. *Naval Research Logistics* **48**(3) 201–209.

Lippman, S.A. 1975. Applying a new device in optimization of exponential queueing systems. *Operations Research* **23**(4) 687–710.

Mandelbaum, A., S. Zeltyn. 2009. Staffing many-server queues with impatient customers: Constraint satisfaction in call centers. *Operations Research* **57**(5) 1189–1205.

M'Hallah, R., R.L. Bulfin. 2007. Minimizing the weighted number of tardy jobs on a single machine with release dates. *European Journal of Operational Research* **176**(2) 727–744.

Moore, M.J. 1968. An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science* **15** 102–109.

Movaghar, A. 1998. On queueing with customer impatience until the beginning of service. *Queueing Systems* **29**(2-4) 337–350.

Movaghar, A. 2005. Optimal control of parallel queues with impatient customers. *Performance Evaluation* **60**(1-4) 327–343. doi:10.1016/j.peva.2004.10.016.

Movaghar, A. 2006. On queueing with customer impatience until the end of service. *Stochastic Models* **22**(1) 149–173. doi:10.1080/15326340500481804.

Nain, P., P. Tsoucas, J. Walrand. 1989. Interchange arguments in stochastic scheduling. *Journal of Applied Probability* 815–826.

Panwar, S.S., D. Towsley, J.K. Wolf. 1988. Optimal scheduling policies for a class of queues with customer deadlines to the beginning of service. *Journal of the ACM* **35**(4) 832–844.

Péridy, L., E. Pinson, D. Rivreau. 2003. Using short-term memory to minimize the weighted number of late jobs on a single machine. *European Journal of Operational Research* **148**(3) 591–603.

Pinedo, M.L. 1983. Stochastic scheduling with release dates and due dates. *Operations Research* **31**(3) 559–572.

Pinedo, M.L. 2008. *Scheduling: Theory, Algorithms, and Systems*. Springer.

Pinedo, M.L., E. Rammouz. 1988. A note on stochastic scheduling on a single machine subject to breakdown and repair. *Probability in the Engineering and Informational Sciences* **2**(01) 41–49.

Potts, C.N., L.N. Van Wassenhove. 1988. Algorithms for scheduling a single machine to minimize the weighted number of late jobs. *Management Science* **34**(7) 843–858.

Puterman, M.L. 1994. *Markov Decision Processes*. John Wiley and Sons Inc.

Ross, Sheldon M. 1970. *Applied probability models with optimization applications*. Courier Dover Publications.

Ross, Sheldon M. 2006. *Introduction to probability models*. Academic press.

Rothkopf, M.H. 1966. Scheduling independent tasks on parallel processors. *Management Science* **12**(5) 437–447.

Sadykov, R. 2008. A branch-and-check algorithm for minimizing the weighted number of late jobs on a single machine with release dates. *European Journal of Operational Research* **189**(3) 1284–1304.

Salch, A., J.-P. Gayon, P. Lemaire. 2011. Stochastic scheduling with impatience. *Cahiers Leibniz* **194**. `http://hal.archives-ouvertes.fr/hal-00641681/fr/`.

Salch, A., J.-P. Gayon, P. Lemaire. 2012a. An optimal static priority rule for stochastic scheduling with impatience. *14^{th} IFAC Symposium on Information Control Problems in Manufacturing*, vol. 14. 105–110.

Salch, A., J.-P. Gayon, P. Lemaire. 2012b. Stochastic scheduling with abandonments. *4^{th} International Conference on Information Systems, Logistics and Supply Chain*.

Salch, A., J.-P. Gayon, P. Lemaire. 2013. Optimal static priority rules for stochastic scheduling with impatience. *Operations Research Letters* **41**(1) 81 – 85. doi:10.1016/j.orl.2012.11.008.

Schrage, L.E. 1968. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research* **16**(3) 687–690.

Schrage, L.E., L.W. Miller. 1966. The queue $M/G/1$ with the shortest remaining processing time discipline. *Operations Research* **14**(4) 670–684. doi:10.1287/opre.14.4.670.

Seo, D.K., C.M. Klein, W. Jang. 2005. Single machine stochastic scheduling to minimize the expected number of tardy jobs using mathematical programming models. *Computers and Industrial Engineering* **48**(2) 153–161.

Sitters, René. 2001. Two $\mathcal{NP}$-hardness results for preemptive minsum scheduling of unrelated parallel machines. *Integer Programming and Combinatorial Optimization*. Springer, 396–405.

Smith, W.E. 1956. Various optimizers for single-stage production. *Naval Research Logistics Quarterly* **3**(1-2) 59–66.

Trietsch, D., K.R. Baker. 2008. Minimizing the number of tardy jobs with stochastically-ordered processing times. *Journal of Scheduling* **11**(1) 71–73.

van den Akker, M., H. Hoogeveen. 2008. Minimizing the number of late jobs in a stochastic setting using a chance constraint. *Journal of Scheduling* **11**(1) 59–69.

Ward, A.R., S. Kumar. 2008. Asymptotically optimal admission control of a queue with impatient customers. *Mathematics of Operations Research* **33**(1) 167–202.

Weber, R.R., P. Varaiya, J. Walrand. 1986. Scheduling jobs with stochastically ordered processing times on parallel machines to minimize expected flowtime. *Journal of Applied Probability* **23**(3) 841–847. doi:10.2307/3214023.

Weiss, G. 1992. Turnpike optimality of smith rule in parallel machines stochastic scheduling. *Mathematics of Operations Research* **17**(2) 255–270. doi:10.1287/moor.17.2.255.

Zeltyn, S. 2004. Call centers with impatient customers: exact analysis and many-server asymptotics of the $M/M/n+G$ queue. Ph.D. thesis, Israel Institute of Technology.

Zeltyn, S., Z. Feldman, S. Wasserkrug. 2009. Waiting and sojourn times in a multi-server queue with mixed priorities. *Queueing Systems* **61**(4) 305–328.

Zeltyn, S., A. Mandelbaum. 2005. Call centers with impatient customers: many-server asymptotics of the $M/M/n+G$ queue. *Queueing Systems* **51**(3-4) 361–402.

Zhao, Z.-X., S.S. Panwar, D. Towsley. 1991. Queueing performance with impatient customers. *INFOCOM '91. Proceedings. Tenth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking in the 90s., IEEE*. 400–409 vol.1. doi:10.1109/INFCOM.1991.147530.

Zhao, Z.-X., S.S. Panwar, D. Towsley. 1993. Optimal scheduling policy of queues with impatient customers. Tech. rep., University of Massachusetts.

Annexe A

Résultats fondamentaux de probabilité

A.1 Loi exponentielle

La loi exponentielle est souvent utilisée dans les processus stochastiques car c'est une loi sans mémoire : la probabilité que la variable aléatoire se réalise avant la date $t + s$ sachant qu'elle ne s'est pas réalisée avant la date t, est la même que la probabilité que la variable aléatoire se réalise avant la date s. Dans un processus aléatoire, si toutes les variables aléatoires suivent des lois exponentielles, le système tout entier est sans mémoire, on dit que c'est un système markovien. Cette hypothèse simplifie considérablement l'étude de ces systèmes, car leur comportement est indépendant d'événements qui se sont produits dans le passé.

Dans cette section, nous présentons quelques propriétés de la loi exponentielle qui sont utilisées au long du document.

A.1.1 Définitions

Notation

Si une variable aléatoire X suit une loi exponentielle, on note $X \sim \exp(\lambda)$.

Densité de probabilité

La densité de probabilité d'une variable aléatoire exponentielle $X \sim \exp(\lambda)$ est donnée par

$$f_X(t) = \begin{cases} \lambda e^{-\lambda t} & \text{si } t \geq 0 \\ 0 & \text{si } t < 0. \end{cases}$$

Fonction de répartition

La fonction de répartition d'une variable aléatoire exponentielle $X \sim \exp(\lambda)$ est donnée par

$$F_X(t) = \mathbb{P}(X \leq t) = \begin{cases} 1 - e^{-\lambda t} & \text{si } t \geq 0 \\ 0 & \text{si } t < 0. \end{cases}$$

Espérance

L'espérance d'une variable aléatoire exponentielle $X \sim \exp(\lambda)$ est

$$E(X) = \int_{t=0}^{+\infty} t\lambda e^{-\lambda t} dt = \frac{1}{\lambda}.$$

A.1.2 Propriété sans mémoire

Une variable aléatoire exponentielle $X \sim \exp(\lambda)$ respecte la propriété sans mémoire, à savoir

$$\forall t, s \geq 0, \mathbb{P}(X \geq t + s \mid X \geq s) = \mathbb{P}(X \geq t).$$

Démonstration. En utilisant la formule des probabilités conditionnelles, il vient

$$\begin{aligned} \mathbb{P}(X \geq t + s \mid X \geq s) &= \frac{\mathbb{P}(X \geq t + s)}{\mathbb{P}(X \geq s)} \\ &= \frac{1 - F_X(t + s)}{1 - F_X(s)} \\ &= \frac{e^{-\lambda(t+s)}}{e^{-\lambda s}} \\ &= 1 - F_X(t) \\ &= \mathbb{P}(X \geq t). \end{aligned}$$

□

A.1.3 Minimum de deux lois exponentielles

Soient $X \sim \exp(\lambda)$ et $Y \sim \exp(\mu)$ deux variables aléatoires indépendantes suivant des lois exponentielles, on a alors

$$\min(X, Y) \sim \exp(\lambda + \mu).$$

Démonstration. Notons $Z = \min(X, Y)$, on a alors

$$\begin{aligned} F_Z(t) &= \mathbb{P}(\min(X,Y) \leq t) \\ &= 1 - \mathbb{P}(\min(X,Y) \geq t) \\ &= 1 - \mathbb{P}(X \geq t)\mathbb{P}(Y \geq t) \\ &= 1 - (1 - F_X(t))(1 - F_Y(t)) \\ &= 1 - e^{-\lambda t}e^{-\mu t} \\ &= 1 - e^{-(\lambda+\mu)t}. \end{aligned}$$

□

A.1.4 Comparaison de deux lois exponentielles

Soient $X \sim \exp(\lambda)$ et $Y \sim \exp(\mu)$ deux variables aléatoires indépendantes suivant des lois exponentielles, on a alors

$$\mathbb{P}(X < Y) = \frac{\lambda}{\lambda + \mu}.$$

Démonstration.

$$\begin{aligned} \mathbb{P}(X < Y) &= \int_{x=0}^{+\infty} F_Y(x) f_X(x) dx \\ &= \int_{x=0}^{+\infty} (1 - e^{-\lambda})\mu e^{-\mu} dx \\ &= 1 - \mu \int_{x=0}^{+\infty} e^{-(\lambda+\mu)x} dx \\ &= 1 + \frac{\mu}{\lambda + \mu} \left[e^{-(\lambda+\mu)x} \right]_{x=0}^{+\infty} \\ &= \frac{\lambda}{\lambda + \mu}. \end{aligned}$$

□

A.1.5 Processus de Poisson

Le processus de Poisson est un des processus les plus utilisé en théorie des files d'attentes, où il peut modéliser l'arrivée des clients dans le système. Notons $\{T_n\}$, $n \in \mathbb{N}$, l'ensemble de ces dates d'arrivées. Dans un processus de Poisson de paramètre λ, nous avons pour tout n, $S_n = T_{n+1} - T_n \sim \exp(\lambda)$. Pour conséquent, le temps entre deux arrivées suit une loi exponentielle de paramètre λ et le processus de Poisson modélise un phénomène d'arrivées indépendantes sans mémoire.

A.2 Ordre stochastique

A.2.1 Définition

Soient X et Y deux variables aléatoires. On dit que X est plus petit que Y au sens de l'ordre stochastique, et on note $X \leq_{st} Y$, si pour tout t

$$\mathbb{P}(X > t) \leq \mathbb{P}(Y > t).$$

A.2.2 Propriété

Soient X_i, X_u et Z trois variables aléatoires indépendantes telles que $X_i \leq_{st} X_u$ et Z une variable aléatoire à valeurs dans $\mathbb{R}^+$, alors $Z + X_i \leq_{st} Z + X_u$.

Démonstration. Si $X_i \leq_{st} X_u$, alors par définition $\mathbb{P}(X_i \leq t) \geq \mathbb{P}(X_u \leq t)\ \forall t \geq 0$. Dans ce cas, pour tout $t \geq 0$ et $0 \leq z \leq t$, $\mathbb{P}(X_i \leq t - z \mid Z = z) \geq \mathbb{P}(X_u \leq t - z \mid Z = z)$ car Z et les X_j sont indépendants. En utilisant la formule des probabilités conditionnelles $\mathbb{P}(Z + X_i \leq t) = \mathbb{P}(X_i \leq t - z \mid Z = z)\mathbb{P}(Z = z) \geq \mathbb{P}(X_u \leq t - z \mid Z = z)\mathbb{P}(Z = z) = \mathbb{P}(Z + X_u \leq t)$. Par conséquence, $F_{Z+X_i}(t) \geq F_{Z+X_u}(t)\ \forall t \geq 0$. □

A.3 Transformée de Laplace

La transformation de Laplace est une opération de transformation sur l'ensemble des fonctions à support positif (e.g. Kuhfittig 1978). Dans cette thèse, nous n'utiliserons cette transformation que sur des fonctions continues en zéro, à support positif et à valeurs dans $\mathbb{R}$, en des fonctions à valeurs dans $\mathbb{R}$. Cela nous permet de simplifier quelques définitions concernant les transformées de Laplace. Cette transformation est utilisée car certaines opérations sont plus simples sur les fonctions transformées que sur les fonctions originales. En effet, la dérivation devient une multiplication et le produit de convolution un produit simple. Nous présentons dans cette section quelques définitions et propriétés de la transformation de Laplace et ensuite nous montrons en quoi elle est utile dans le domaine des probabilités.

A.3.1 Notation et définition

Soit $f : \mathbb{R}^+ \to \mathbb{R}$ une fonction intégrable, nous notons $\mathcal{L}\{f\}(s)$ la transformée de Laplace de f en s définie par :

$$\mathcal{L}\{f\}(s) = \int_0^{+\infty} e^{-st} f(t) dt.$$

A.3.2 Exemple : $\mathcal{L}\{f(t) = \lambda e^{-\lambda t}\}$

Une variable aléatoire suivant une loi exponentielle de paramètre λ a une fonction de densité $f(t) = \lambda e^{-\lambda t}$. Sa transformée de Laplace est

$$\mathcal{L}\{f(t) = \lambda e^{-\lambda t}\}(s) = \frac{\lambda}{\lambda + s}.$$

Démonstration.

$$\begin{aligned}\mathcal{L}\{f(t) = \lambda e^{-\lambda t}\} &= \int_0^{+\infty} e^{-st} \lambda e^{-\lambda t} dt \\ &= -\frac{\lambda}{\lambda + s} \left[e^{-(\lambda+s)t}\right]_{t=0}^{+\infty} \\ &= \frac{\lambda}{\lambda + s}.\end{aligned}$$

□

A.3.3 Propriétés

Linéarité

Soient $f, g : \mathbb{R}^+ \to \mathbb{R}$ deux fonctions intégrables et $\alpha, \beta \in \mathbb{R}$, alors

$$\mathcal{L}\{\alpha f + \beta g\} = \alpha \mathcal{L}\{f\} + \beta \mathcal{L}\{g\}.$$

Dérivation

Soit $f : \mathbb{R}^+ \to \mathbb{R}$ une fonction intégrable et dérivable, alors

$$\mathcal{L}\{f'\}(s) = s\mathcal{L}\{f\}(s) + f(0).$$

Démonstration. En intégrant par parties, il vient :

$$\begin{aligned}\mathcal{L}\{f'\}(s) &= \int_0^{+\infty} e^{-st} f'(t) dt \\ &= [e^{-st} f(t)]_0^{+\infty} - \int_0^{+\infty} (-s) e^{-st} f(t) dt \\ &= f(0) + s\mathcal{L}\{f\}(s).\end{aligned}$$

□

Produit de convolution

Soient $f, g : \mathbb{R}^+ \to \mathbb{R}$ deux fonctions intégrables, on note $(f*g)(t) = \int_{x=0}^{t} f(t-x)g(x)dx$ le produit de convolution de f et de g, alors

$$\mathcal{L}\{f * g\} = \mathcal{L}\{f\}\mathcal{L}\{g\}.$$

Démonstration.

$$\begin{aligned}
\mathcal{L}\{f * g\}(s) &= \int_{t=0}^{+\infty} e^{-st}(f * g)(t)dt \\
&= \int_{t=0}^{+\infty} e^{-st}\left(\int_{x=0}^{t} f(t-x)g(x)dx\right)dt \\
&= \int_{x=0}^{t} g(x)\left(\int_{t=0}^{+\infty} e^{-st}f(t-x)dt\right)dx.
\end{aligned}$$

On admet que $\mathcal{L}\{f(t-x)\}(s) = e^{-sx}\mathcal{L}\{f\}(s)$, il vient alors

$$\begin{aligned}
\mathcal{L}\{f * g\}(s) &= \int_{x=0}^{t} g(x)e^{-sx}\mathcal{L}\{f\}(t)dx \\
&= \mathcal{L}\{f\}(s)\mathcal{L}\{g\}(s).
\end{aligned}$$

□

A.3.4 Lien avec les probabilités

Dans cette partie, nous allons donner deux exemples qui montrent l'intérêt des transformées de Laplace dans l'expression de certaines probabilités. Tout d'abord, on remarque que la transformée de Laplace s'applique à des fonctions à support positif, or les fonctions de densités des variables aléatoires représentant des durées sont aussi à support positif.

Soient X, Y deux variables aléatoires. On suppose que la fonction de densité de X est intégrable et que $Y \sim \exp(\lambda)$. On a alors

$$\mathbb{P}(X \leq Y) = \mathcal{L}\{f_X\}(\lambda).$$

Démonstration.

$$\begin{aligned}
\mathbb{P}(X \leq Y) &= \int_{x=0}^{+\infty}\int_{y=x}^{+\infty} f_X(x)f_Y(y)dydx \\
&= \int_{x=0}^{+\infty} f_X(x)(1 - F_Y(x))dx \\
&= \int_{x=0}^{+\infty} f_X(x)e^{-\lambda x}dx \\
&= \mathcal{L}\{f_X\}(\lambda).
\end{aligned}$$

□

Soient X, Y deux variables aléatoires dont les fonctions de densités sont intégrables. On sait que $f_{X+Y} = f_X * f_Y$, par conséquent, en appliquant la transformée de Laplace à cette égalité, on obtient

$$\mathcal{L}\{f_{X+Y}\} = \mathcal{L}\{f_X\}\mathcal{L}\{f_Y\}.$$

Annexe B

Preuves du chapitre 2

B.1 Probabilités stationnaires d'un processus de naissance et mort

On considère une chaîne de Markov ayant comme état les entiers $j \in \mathbb{N}$. Pour tout $j > 0$, on passe d'un état j à un état $j+1$ avec un taux λ, et d'un état j à un état $j-1$ avec un taux $j\gamma$ et on sort de l'état 0 avec un taux λ. Un schéma de cette chaîne de Markov est donné dans la figure 28. Cette chaîne décrit un système dans lequel des tâches deviennent disponibles suivant un processus de Poisson de taux λ, qui s'impatientent suivant une loi exponentielle de taux γ, mais qui ne sont jamais exécutées. Notons π_j, pour tout j, la probabilité stationnaire de se trouver dans l'état j dans un tel système, et π'_j dans un système où les tâches sont exécutées sur une machine de durée d'exécution exponentielle de taux μ. Dans ce cas, nous avons pour tout j, $\pi_j > \pi'_j$, car si l'on sert des tâches, le système se vide plus vite.

Dans ce cas, les probabilités stationnaires respectent le système d'équations suivant

$$\begin{cases} \lambda\pi_j = (j+1)\gamma\pi_{j+1} & \forall j \in \mathbb{N} \\ \sum_{j=0}^{+\infty} \pi_j = 1 \end{cases} \Leftrightarrow \begin{cases} \pi_j = \frac{(\lambda/\gamma)^j}{j!}\pi_0 & \forall j \in \mathbb{N} \\ \sum_{j=0}^{+\infty} \pi_j = 1. \end{cases}$$

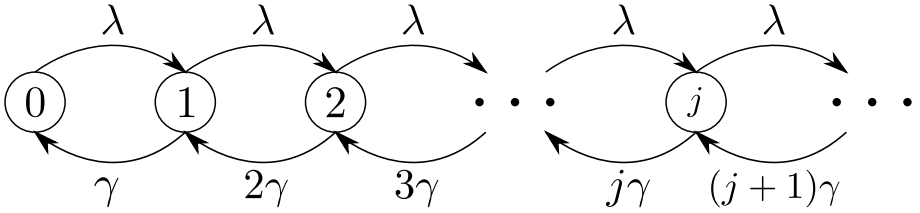


FIGURE 28 – Chaîne de Markov d'un processus de naissance et mort

On en déduit que

$$\pi_0 \sum_{j=0}^{+\infty} \frac{(\lambda/\gamma)^j}{j!} = 1,$$

donc que $\pi_0 = e^{-\lambda/\gamma}$ et finalement

$$\pi_j = \frac{(\lambda/\gamma)^j}{j!} e^{-\lambda/\gamma}.$$

B.2 Détails de la preuve du théorème 4

Le théorème 4 se trouve page 35.

Détaillons l'écriture du coût de l'ordonnancement $\{1, 2\}$:

$$\begin{aligned} C(1,2) &= w_1 \mathbb{P}(X_1 > d) + w_2 \mathbb{P}(X_1 + X_2 > d) \\ &= w_1 e^{-\mu_1 d} + w_2 (1 - F_{X_1+X_2}(d)). \end{aligned}$$

La fonction de répartition de $X_1 + X_2$ est donnée par

$$\begin{aligned} F_{X_1+X_2}(d) &= \int_{x_1=0}^{d} \int_{x_2=0}^{d-x_1} f_{X_1}(x_1) f_{X_2}(x_2) dx_2 dx_1 \\ &= \int_{x_1=0}^{d} f_{X_1}(x_1) \left(\int_{x_2=0}^{d-x_1} f_{X_2}(x_2) dx_2 \right) dx_1 \\ &= \int_{x_1=0}^{d} f_{X_1}(x_1) F_{X_2}(d - x_1) dx_1 \\ &= \int_{x_1=0}^{d} f_{X_1}(x_1) \left(1 - e^{-\mu_2 (d-x_1)} \right) dx_1 \\ &= F_{X_1}(d) - \int_{x_1=0}^{d} \mu_1 e^{-\mu_1 x_1} e^{-\mu_2 (d-x_1)} dx_1 \\ &= F_{X_1}(d) + \frac{\mu_1}{\mu_1 - \mu_2} e^{-\mu_2 d} \left[e^{-(\mu_1 - \mu_2) x_1} \right]_{x_1=0}^{d} \\ &= 1 - e^{-\mu_1 d} + \frac{\mu_1}{\mu_1 - \mu_2} \left(e^{-\mu_1 d} - e^{-\mu_2 d} \right). \end{aligned}$$

Par conséquent

$$\begin{aligned} C(1,2) &= w_1 e^{-\mu_1 d} + w_2 \left(e^{-\mu_1 d} - \frac{\mu_1}{\mu_1 - \mu_2} \left(e^{-\mu_1 d} - e^{-\mu_2 d} \right) \right) \\ &= (w_1 + w_2) e^{-\mu_1 d} - \frac{w_2 \mu_1}{\mu_1 - \mu_2} \left(e^{-\mu_1 d} - e^{-\mu_2 d} \right) \end{aligned}$$

et par identification

$$C(2,1) = (w_1 + w_2) e^{-\mu_2 d} - \frac{w_1 \mu_2}{\mu_1 - \mu_2} \left(e^{-\mu_1 d} - e^{-\mu_2 d} \right).$$

En faisant la différence de ces deux coûts, on retrouve l'expression donnée dans la démonstration.

B.3 Propagation des propriétés du théorème 5

Le théorème 5 se trouve page 44.

B.3.1 Propagation de la propriété P1 : $\Delta_i v(x) \geq 0$

L'opérateur de production $\Delta_i T_{prod} v(x)$

$$\Delta_1 T_{\text{prod}} v(x) = \Delta_1 v(x) - \begin{cases} \max\{\Delta_1 v(x); \Delta_2 v(x+e_1-e_2); 0\} & \\ -\max\{\Delta_1 v(x-e_1); \Delta_2 v(x-e_2); 0\} & \text{si} \quad x_1 > 0,\, x_2 > 0 \\ \max\{\Delta_1 v(x); \Delta_2 v(x+e_1-e_2); 0\} & \\ -\max\{\Delta_2 v(x-e_2); 0\} & \text{si} \quad x_1 = 0,\, x_2 > 0 \\ \max\{\Delta_1 v(x); 0\} & \\ -\max\{\Delta_1 v(x-e_1); 0\} & \text{si} \quad x_1 > 0,\, x_2 = 0 \\ \max\{\Delta_1 v(x); 0\} & \text{si} \quad x_1 = 0,\, x_2 = 0 \end{cases}$$

Considérons le cas $x_1 > 0$ et $x_2 > 0$ et supposons que $v(x)$ respecte la propriété P1. On peut alors séparer le problème en trois cas :

1. $\Delta_1 v(x) \geq \Delta_2 v(x+e_1-e_2)$
   Alors $\Delta_1 T_{\text{prod}} v(x) = \max\{\Delta_1 v(x-e_1); \Delta_2 v(x-e_2); 0\} \geq 0$

2. $\Delta_1 v(x) \leq \Delta_2 v(x+e_1-e_2)$ et $\Delta_1 v(x-e_1) \geq \Delta_2 v(x-e_2)$
   Alors $\Delta_1 T_{\text{prod}} v(x) = \Delta_1 v(x) - \Delta_2 v(x+e_1-e_2) + \Delta_1 v(x-e_1) = \Delta_1 v(x-e_2) + \Delta_1 v(x-e_1) - \Delta_2 v(x-e_2) \geq 0$

3. $\Delta_1 v(x) \leq \Delta_2 v(x+e_1-e_2)$ et $\Delta_1 v(x-e_1) \leq \Delta_2 v(x-e_2)$
   Alors $\Delta_1 T_{\text{prod}} v(x) = \Delta_1 v(x) - \Delta_2 v(x+e_1-e_2) + \Delta_2 v(x-e_2) = \Delta_1 v(x-e_2) \geq 0$

Les trois autres cas ($x_1 = 0$ et $x_2 > 0$, $x_1 > 0$ et $x_2 = 0$, $x_1 = 0$ et $x_2 = 0$) se traitent de la même manière. On en déduit que l'opérateur de production propage la propriété P1 dans la direction 1.

$$\Delta_2 T_{\text{prod}} v(x) = \Delta_2 v(x) - \begin{cases} \max\{\Delta_1 v(x-e_1+e_2); \Delta_2 v(x); 0\} & \\ -\max\{\Delta_1 v(x-e_1); \Delta_2 v(x-e_2); 0\} & \text{si} \quad x_1 > 0,\, x_2 > 0 \\ \max\{\Delta_2 v(x); 0\} & \\ -\max\{\Delta_2 v(x-e_2); 0\} & \text{si} \quad x_1 = 0,\, x_2 > 0 \\ \max\{\Delta_1 v(x-e_1+e_2); \Delta_2 v(x); 0\} & \\ -\max\{\Delta_1 v(x-e_1); 0\} & \text{si} \quad x_1 > 0,\, x_2 = 0 \\ \max\{\Delta_2 v(x); 0\} & \text{si} \quad x_1 = 0,\, x_2 = 0 \end{cases}$$

Considérons le cas $x_1 > 0$ et $x_2 > 0$ et considérons que $v(x)$ respecte la propriété P1. On peut alors séparer le problème en trois cas :

1. $\Delta_2 v(x) \geq \Delta_1 v(x - e_1 + e_2)$
   Alors $\Delta_2 T_{\text{prod}} v(x) = \max\{\Delta_1 v(x-e_1); \Delta_2 v(x-e_2); 0\} \geq 0$
2. $\Delta_2 v(x) \leq \Delta_1 v(x - e_1 + e_2)$ et $\Delta_1 v(x-e_1) \geq \Delta_2 v(x-e_2)$
   Alors $\Delta_2 T_{\text{prod}} v(x) = \Delta_2 v(x) - \Delta_1 v(x-e_1+e_2) + \Delta_1 v(x-e_1) = \Delta_2 v(x-e_1) \geq 0$
3. $\Delta_2 v(x) \leq \Delta_1 v(x - e_1 + e_2)$ et $\Delta_1 v(x-e_1) \leq \Delta_2 v(x-e_2)$
   Alors $\Delta_2 T_{\text{prod}} v(x) = \Delta_2 v(x) - \Delta_1 v(x-e_1+e_2) + \Delta_2 v(x-e_2) = \Delta_2 v(x-e_1) - \Delta_1 v(x-e_1) + \Delta_2 v(x-e_2) \geq 0$

Les trois autres cas ($x_1 = 0$ et $x_2 > 0$, $x_1 > 0$ et $x_2 = 0$, $x_1 = 0$ et $x_2 = 0$) se traitent de la même manière. On en déduit que l'opérateur de production propage la propriété P1 dans la direction 2.

Les opérateurs d'impatience $\Delta_i T^j_{imp} v(x)$

$$\Delta_1 T^1_{imp} v(x) = \begin{cases} w_1 + x_1 \Delta_1 v(x-e_1) + (M_1 - x_1 - 1)\Delta_1 v(x) & \text{si} \quad x_1 > 0 \\ w_1 + (M_1 - 1)\Delta_1 v(x) & \text{si} \quad x_1 = 0 \end{cases}$$

$$\Delta_2 T^1_{imp} v(x) = \begin{cases} x_1 \Delta_2 v(x-e_1) + (M_1 - x_1)\Delta_2 v(x) & \text{si} \quad x_1 > 0 \\ M_1 \Delta_2 v(x) & \text{si} \quad x_1 = 0 \end{cases}$$

$$\Delta_1 T^2_{imp} v(x) = \begin{cases} x_2 \Delta_1 v(x-e_2) + (M_2 - x_2)\Delta_1 v(x) & \text{si} \quad x_2 > 0 \\ M_2 \Delta_1 v(x) & \text{si} \quad x_2 = 0 \end{cases}$$

$$\Delta_2 T^2_{imp} v(x) = \begin{cases} w_2 + x_2 \Delta_2 v(x-e_2) + (M_2 - x_2 - 1)\Delta_2 v(x) & \text{si} \quad x_2 > 0 \\ w_2 + (M_2 - 1)\Delta_2 v(x) & \text{si} \quad x_2 = 0 \end{cases}$$

On voit facilement que les opérateurs d'impatience propagent la propriété P1 dans les directions 1 et 2.

Les opérateurs d'arrivée $\Delta_i T^j_{arr} v(x)$

$$\Delta_1 T^1_{arr} v(x) = \begin{cases} \Delta_1 v(x+e_1) & \text{si} \quad x_1+1 < M_1 \\ w_1 & \text{si} \quad x_1+1 = M_1 \end{cases}$$

$$\Delta_2 T^1_{arr} v(x) = \begin{cases} \Delta_2 v(x+e_1) & \text{si} \quad x_1 < M_1 \\ \Delta_2 v(x) & \text{si} \quad x_1 = M_1 \end{cases}$$

$$\Delta_1 T^2_{arr} v(x) = \begin{cases} \Delta_1 v(x+e_2) & \text{si} \quad x_2 < M_2 \\ \Delta_1 v(x) & \text{si} \quad x_2 = M_2 \end{cases}$$

$$\Delta_2 T^2_{arr} v(x) = \begin{cases} \Delta_2 v(x+e_2) & \text{si} \quad x_2+1 < M_2 \\ 0 & \text{si} \quad x_2+1 = M_2 \end{cases}$$

On voit facilement que les opérateurs d'arrivée propagent la propriété P1 dans les directions 1 et 2.

Propagation de P1 : $\Delta_i v(x) \geq 0$

Tous les coûts marginaux des méta-opérateurs sont positifs suivant 1 et 2. En faisant une somme pondérée de ces coûts marginaux, on trouve encore un résultat positif. On a donc bien $\Delta_i T_{IES} v(x) \geq 0$ pour $i \in \{1,2\}$ et pour tout x. Donc l'opérateur T_{IES} propage bien la propriété P1.

B.3.2 Propagation de la propriété P2 : $\Delta_i v(x) \leq w_i$

L'opérateur de production $\Delta_i T_{prod} v(x)$

En supposant que v respecte les propriétés P1 et P3, on simplifie l'écriture de l'opérateur de production.

$$\Delta_1 T_{\text{prod}} v(x) = \begin{cases} \Delta_1 v(x-e_1) & \text{si} \quad x_1 > 0 \\ \Delta_2 v(x-e_2) & \text{si} \quad x_1 = 0,\ x_2 > 0 \\ 0 & \text{si} \quad x_1 = 0,\ x_2 = 0 \end{cases}$$

Comme les hypothèses que l'on a posées entraînent que $w_1 \geq w_2$, on en déduit que l'opérateur de production propage la propriété P2 dans la direction 1.

$$\Delta_2 T_{\text{prod}} v(x) = \begin{cases} \Delta_2 v(x - e_1) & \text{si} \quad x_1 > 0 \\ \Delta_2 v(x - e_2) & \text{si} \quad x_1 = 0,\, x_2 > 0 \\ 0 & \text{si} \quad x_1 = 0,\, x_2 = 0 \end{cases}$$

L'opérateur de production propage bien la propriété P2 dans la direction 2.

Les opérateurs d'impatience $\Delta_i T^j_{imp} v(x)$

$$\Delta_1 T^1_{imp} v(x) = \begin{cases} w_1 + x_1 \Delta_1 v(x - e_1) + (M_1 - x_1 - 1)\Delta_1 v(x) & \text{si} \quad x_1 > 0 \\ w_1 + (M_1 - 1)\Delta_1 v(x) & \text{si} \quad x_1 = 0 \end{cases}$$

$$\Delta_2 T^1_{imp} v(x) = \begin{cases} x_1 \Delta_2 v(x - e_1) + (M_1 - x_1)\Delta_2 v(x) & \text{si} \quad x_1 > 0 \\ M_1 \Delta_2 v(x) & \text{si} \quad x_1 = 0 \end{cases}$$

$$\Delta_1 T^2_{imp} v(x) = \begin{cases} x_2 \Delta_1 v(x - e_2) + (M_2 - x_2)\Delta_1 v(x) & \text{si} \quad x_2 > 0 \\ M_2 \Delta_1 v(x) & \text{si} \quad x_2 = 0 \end{cases}$$

$$\Delta_2 T^2_{imp} v(x) = \begin{cases} w_2 + x_2 \Delta_2 v(x - e_2) + (M_2 - x_2 - 1)\Delta_2 v(x) & \text{si} \quad x_2 > 0 \\ w_2 + (M_2 - 1)\Delta_2 v(x) & \text{si} \quad x_2 = 0 \end{cases}$$

On voit facilement que les opérateurs d'impatience propagent la propriété P2 dans les directions 1 et 2.

Les opérateurs d'arrivée $\Delta_i T^j_{arr} v(x)$

$$\Delta_1 T^1_{arr} v(x) = \begin{cases} \Delta_1 v(x + e_1) & \text{si} \quad x_1 + 1 < M_1 \\ w_1 & \text{si} \quad x_1 + 1 = M_1 \end{cases}$$

$$\Delta_2 T^1_{arr} v(x) = \begin{cases} \Delta_2 v(x + e_1) & \text{si} \quad x_1 < M_1 \\ \Delta_2 v(x) & \text{si} \quad x_1 = M_1 \end{cases}$$

$$\Delta_1 T^2_{arr} v(x) = \begin{cases} \Delta_1 v(x + e_2) & \text{si} \quad x_2 < M_2 \\ \Delta_1 v(x) & \text{si} \quad x_2 = M_2 \end{cases}$$

$$\Delta_2 T^2_{arr} v(x) = \begin{cases} \Delta_2 v(x+e_2) & \text{si} \quad x_2+1 < M_2 \\ 0 & \text{si} \quad x_2+1 = M_2 \end{cases}$$

On voit facilement que les opérateurs d'arrivée propagent la propriété P2 dans les directions 1 et 2.

Propagation de P2 : $\Delta_i v(x) \leq w_i$

Tous les coûts marginaux des méta-opérateurs sont inférieurs ou égaux à w_i suivant i. En faisant une somme pondérée de ces coûts marginaux, on trouve grâce au redimensionnement de l'échelle des temps

$$\Delta_i T_{IES} \leq \left(\mu + \sum_{j \in \{1,2\}} (\lambda_j + M_j \gamma_j) \right) w_i \leq w_i.$$

On a donc bien $\Delta_i T_{IES} v(x) \leq w_i$ pour $i \in \{1,2\}$ et pour tout x. Donc l'opérateur T_{IES} propage bien la propriété P2.

B.3.3 Propagation de la propriété P3 : $\Delta^3 v(x) = \Delta_1 v(x-e_1) - \Delta_2 v(x-e_2) \geq 0$

L'opérateur de production $\Delta^3 T_{\mathbf{prod}} v(x)$

Comme v respecte la propriété P3, l'opérateur de production peut directement être simplifié en donnant la priorité aux tâches de la classe 1, s'il y en a dans le système. Enfin, lorsque $x_1 > 0$ et $x_2 > 0$,

$$\Delta^3 T_{\text{prod}} v(x) = \begin{cases} \Delta_1 v(x-2e_1) - \Delta_2 v(x-e_1-e_2) = \Delta^3 v(x-e_1) & \text{si } x_1 > 1 \\ \Delta_2 v(x-e_1-e_2) - \Delta_2 v(x-e_1-e_2) = 0 & \text{si } x_1 = 1 \end{cases}$$

L'opérateur de production propage bien la propriété P3.

Les opérateurs d'impatience $\Delta^3 T^j_{imp} v(x)$

Lorsque $x_1 > 0$ et $x_2 > 0$,

$$\Delta^3 T^1_{imp} v(x) = \begin{cases} w_1 - \Delta_2 v(x-e_1-e_2) \\ \quad +(x_1-1)\Delta^3 v(x-e_1) + (M_1-x_1)\Delta^3 v(x) & \text{si} \quad x_1 > 1 \\ w_1 - \Delta_2 v(x-e_1-e_2) \\ \quad +(M_1-1)\Delta^3 v(x) & \text{si} \quad x_1 = 1 \end{cases}$$

En utilisant la propriété P2 et le fait que $w_1 \geq w_2$, on peut conclure directement que l'opérateur d'impatience de la classe 1 propage la propritété P3. Cependant, on va n'utiliser que la propriété P1 et on suppose pour l'instant que l'on n'a que $\Delta^3 T^1_{imp} v(x) \geq w_1 -$

$\Delta_2 v(x - e_1 - e_2)$.

$$\Delta^3 T^2_{imp} v(x) = \begin{cases} -w_2 + \Delta_1 v(x - e_1 - e_2) & \\ +(x_2 - 1)\Delta^3 v(x - e_2) + (M_2 - x_2)\Delta^3 v(x) & \text{si} \quad x_2 > 1 \\ -w_2 + \Delta_1 v(x - e_1 - e_2) & \\ +(M_2 - 1)\Delta^3 v(x) & \text{si} \quad x_2 = 1 \end{cases}$$

À ce stade, on peut seulement conclure que $\Delta^3 v(x) \geq -w_2$. Comme pour l'opérateur d'impatience de la classe 1, on va n'utiliser que la propriété P1 pour obtenir $\Delta^3 T^2_{imp} v(x) \geq -w_2 + \Delta_1 v(x - e_1 - e_2)$.

Les opérateurs d'arrivée $\Delta^3 T^j_{arr} v(x)$

$$\Delta^3 T^1_{arr} v(x) = \begin{cases} \Delta^3 v(x + e_1) & \text{si} \quad x_1 < M_1 \\ w_1 - \Delta_2 v(x - e_2) & \text{si} \quad x_1 = M_1 \end{cases}$$

En utilisant P2 et le fait que $w_1 \geq w_2$, on en déduit que l'opérateur d'arrivée de la classe 1 propage la propriété P3.

$$\Delta^3 T^2_{arr} v(x) = \begin{cases} \Delta^3 v(x + e_2) & \text{si} \quad x_2 < M_2 \\ \Delta_1 v(x - e_1) & \text{si} \quad x_2 = M_2 \end{cases}$$

On voit que l'opérateur d'arrivée pour la classe 2 propage la propriété P3.

Propagation de P3 : $\Delta_1 v(x - e_1) - \Delta_2 v(x - e_2) \geq 0$

En faisant la somme pondérée des coûts marginaux pour les différents opérateurs, on obtient

$$\begin{aligned} \Delta^3 T_{IES} v(x) \geq\ & \gamma_1 w_1 - \gamma_2 w_2 \\ & + \gamma_2 \Delta_1 v(x - e_1 - e_2) - \gamma_1 \Delta_2 v(x - e_1 - e_2). \end{aligned}$$

Notons que

$$\begin{aligned} & \Delta_1 v(x - e_1 - e_2) - \Delta_2 v(x - e_1 - e_2) \\ & = v(x - e_2) - v(x - e_1 - e_2) - v(x - e_1) + v(x - e_1 - e_2) \\ & = v(x - e_2) - v(x - e_1) \end{aligned}$$

et que

$$\begin{aligned}
&\Delta_1 v(x-e_1) - \Delta_2 v(x-e_2) \\
&= v(x) - v(x-e_1) - v(x) + v(x-e_2) \\
&= v(x-e_2) - v(x-e_1).
\end{aligned}$$

Par conséquent, en supposant que $w_1\gamma_1 \geq w_2\gamma_2$ et que $\gamma_1 \leq \gamma_2$, on a bien $\Delta^3 T_{IES} v(x) \geq 0$. Donc l'opérateur T_{IES} propage bien la propriété P3.

Annexe C

Appendix of Chapter 4

C.1 Preliminary probability results

In this section we provide some probability properties that are useful for the understanding of Chapter 4. In what follows, a random variable Y has a cumulative density function (c.d.f.) F_Y and a probability distribution function (p.d.f.) f_Y.

Property 7. *Let S, X and D be independent random variables. If D is exponentially distributed, then $\mathbb{P}(S + X \geq D \mid S \leq D) = \mathbb{P}(X \geq D)$.*

Proof. The memoryless property of D states that for all non-negative s and t, $\mathbb{P}(t + s \leq D \mid s \leq D) = \mathbb{P}(t \leq D)$, therefore we have

$$\begin{aligned}\mathbb{P}(S + X \geq D \mid S \leq D) &= 1 - \mathbb{P}(S + X \leq D \mid S \leq D) \\ &= 1 - \mathbb{P}(X \leq D) \\ &= \mathbb{P}(X \geq D).\end{aligned}$$

This ends the proof of Property 7. □

Property 8. *Let S, X and D be independent random variables. If D is exponentially distributed, then $E[\min(X, D - S) \mid S \leq D] = E[\min(X, D)]$.*

Proof. As in the proof of Property 7, we use the memoryless property of D to simplify the following expression.

$$\begin{aligned}\mathbb{P}(\min(X, D - S) \geq t \mid S \leq D) &= \mathbb{P}(X \geq t \cap D - S \geq t \mid S \leq D) \\ &= \mathbb{P}(X \geq t \cap S + t \leq D \mid S \leq D) \\ &= \mathbb{P}(X \geq t)\mathbb{P}(S + t \leq D \mid S \leq D)\end{aligned}$$

because X, D and S are independent random variables. Now using Property 7 leads to

$$\begin{aligned}\mathbb{P}(\min(X, D-S) \geq t \mid S \leq D) &= \mathbb{P}(X \geq t)\mathbb{P}(D \geq t) \\ &= \mathbb{P}(X \geq t \cap D \geq t) \\ &= \mathbb{P}(\min(X, D) \geq t).\end{aligned}$$

As both random variables $\min(X, D-S) \mid S \leq D$ and $\min(X, D)$ have the same c.d.f., they have the same expected value as well. This ends the proof of Property 8. □

C.2 Equivalence of cost models

In this section we present a proof of the equivalence of two classical costs models for problems with abandonments, one with abandonment costs w_j and the other with holding costs per unit of time h_j. In this section, processing times are arbitrarily distributed (e.g. not exponential) but patience times are exponentially distributed.

Before presenting the theorem on equivalence of costs models, the following property must be introduced.

Property 9. *Let X be a continuous random variable which admit a c.d.f. f_X. Let D be an exponentially distributed random variable, then $E(\min(X, D)) = E(D)\mathbb{P}(X \geq D)$.*

Proof. Let us express $\mathbb{P}(X \geq D)$ taking advantage of the memoryless property of $D \sim \exp(\gamma)$.

$$\begin{aligned}\mathbb{P}(X \geq D) &= \int_{t=0}^{+\infty} \int_{x=t}^{+\infty} f_D(t) f_X(x) dx dt \\ &= \int_{t=0}^{+\infty} f_D(t)(1 - F_X(t)) dt \\ &= \gamma \int_{t=0}^{+\infty} e^{-\gamma t}(1 - F_X(t)) dt \\ &= \gamma \mathcal{L}\{1 - F_X\}(\gamma)\end{aligned}$$

by introducing $\mathcal{L}\{1 - F_X\}(\gamma)$, the Laplace transform of $1 - F_X$ in γ. $E(\min(X, D))$ can be expressed using Laplace transforms too and it leads to

$$\begin{aligned}E(\min(X, D)) &= \int_{t=0}^{+\infty} \int_{x=t}^{+\infty} t f_D(t) f_X(x) dx dt + \int_{t=0}^{+\infty} \int_{x=t}^{+\infty} t f_X(t) f_D(x) dx dt \\ &= \int_{x=0}^{+\infty} t f_D(t)(1 - F_X(t)) dt + \int_{t=0}^{+\infty} t f_X(t)(1 - F_D(t)) dt \\ &= \int_{x=0}^{+\infty} t\gamma e^{-\gamma t}(1 - F_X(t)) dt + \int_{t=0}^{+\infty} t f_X(t) e^{-\gamma t} dt \\ &= \gamma \mathcal{L}\big\{f(t) = t(1 - F_X(t))\big\}(\gamma) + \mathcal{L}\big\{g(t) = t f_X(t)\big\}(\gamma).\end{aligned}$$

This expression can be further simplified using the derivation property of Laplace transforms. This property, among others, can be found in almost any references on Laplace transforms (e.g. Kuhfittig 1978 or Appendix A.3 page 116). In the present case, we use this property on function f. Hence,

$$E(\min(X,D)) = \mathcal{L}\big\{f'(x) = 1 - F_X(t) - tf_X(t)\big\}(\gamma) + \mathcal{L}\big\{g(t) = tf_X(t)\big\}(\gamma)$$

and the linearity of Laplace transforms leads to $E(\min(X,D)) = \mathcal{L}\{1 - F_X\}(\gamma) = \mathbb{P}(X \geq D)/\gamma = E(D)\mathbb{P}(X \geq D)$. This ends the proof of Property 9. □

Theorem 20. *Let us consider a n jobs scheduling problem with IES. Job j has an arbitrarily distributed processing time, an exponentially distributed patience time with mean $1/\gamma_j$ and an arbitrarily distributed release date. Then the two following costs models, 1) Job j has an abandonment cost w_j and, 2) Job j has a holding cost h_j per unit of time spent in the system, are equivalent. A type 1) (respectively type 2)) problem can be optimally controlled by using a type 2) (respectively type 1)) optimal rule by creating an instance with holding costs $h'_j = w_j\gamma_j$ (respectively abandonment costs $w'_j = h_j/\gamma_j$) for all j in $\{1, 2, \ldots, n\}$.*

This theorem allows to solve problems which mix the two cost models, abandonment costs and holding costs, as one of these. A problem with holding costs h_j and abandonment costs w_j can be switched to a type 1) problem by letting $w'_j = w_j + h_j/\gamma_j$, or to a type 2) problem by letting $h'_j = h_j + w_j\gamma_j$.

This theorem can be interpreted using the memoryless property of the exponential probability distribution. With the patience being exponential, when one considers problems with abandonment costs, one can suppose that the system incurs a cost $w_j\gamma_j$ per unit of time for a type j job because whatever happened previously, during the next time interval dt this job has always the same chance to abandon that can be approximated by $\gamma_j dt$, if dt is small enough, inducing an expected cost of $w_j\gamma_j dt$ over this time interval. Which is consistent with the theorem.

Now let us present the proof of Theorem 20.

Proof. Let us assume that at time $t = 0$, all n jobs are available in the system and none of them have already begun their process. We shall prove that the relation in terms of costs described in Theorem 20 holds for any job in the system. Without loss of generality, we assume that jobs are processed in the order of their indices. Let C_j be the time Job j has spent in the system. Let S_j be the time when Job j begins its process (if it has not abandoned before the beginning of its process) or quit the system (if it has abandoned before the beginning of its process; in such a case, $S_j = D_j$). For the model

with abandonment costs, the expected cost of Job j is expressed as

$$\begin{aligned} C_j^a(S) &= w_j \mathbb{P}(C_j \geq D_j) \\ &= w_j \mathbb{P}(S_j + X_j \geq D_j \mid S_j \geq D_j)\mathbb{P}(S_j \geq D_j) \\ &\quad + w_j \mathbb{P}(S_j + X_j \geq D_j \mid S_j \leq D_j)\mathbb{P}(S_j \leq D_j) \\ &= w_j(\mathbb{P}(S_j \geq D_j) \\ &\quad + \mathbb{P}(S_j + X_j \geq D_j \mid S_j \leq D_j)\mathbb{P}(S_j \leq D_j)). \end{aligned}$$

This expression can be simplified using Property 7, since the due date of Job j follows an exponential distribution. Thus

$$\mathbb{P}(S_j + X_j \geq D_j \mid S_j \leq D_j) = \mathbb{P}(X_j \geq D_j).$$

Then the contribution of Job j is

$$C_j^a(S) = w_j(\mathbb{P}(S_j \geq D_j) + \mathbb{P}(X_j \geq D_j)\mathbb{P}(S_j \leq D_j)).$$

Now let us focus on the expected cost of the same job under the holding cost model: the cost is expressed as

$$\begin{aligned} C_j^h(S) &= h_j E(\min(D_j, C_j)) \\ &= h_j(E(\min(D_j, S_j)) \\ &\quad + E[\min(X_j, D_j - S_j) \mid S_j \leq D_j]\mathbb{P}(S_j \leq D_j)). \end{aligned}$$

Using Property 8, the holding costs expression can be simplified up to

$$C_j^h(S) = h_j(E(\min(D_j, S_j)) + E(\min(X_j, D_j))\mathbb{P}(S_j \leq D_j))$$

and using Property 9 finally gives

$$C_j^h(S) = h_j E(D_j)(\mathbb{P}(S_j \geq D_j) + \mathbb{P}(X_j \geq D_j)\mathbb{P}(S_j \leq D_j)).$$

We have $C_j^h(S) = E(D_j)C_j^a(S)$ and if we let h_j take the value $w_j\gamma_j$ for all j in $\{1, 2, \ldots, n\}$ then the two cost models induce the same costs for each job on the whole schedule.

We assumed previously all jobs to be available at time $t = 0$ in the system. But we need to prove that this equivalence in terms of costs still holds when the system faces arrivals of jobs. Nothing was assumed about S_j or X_j in what precedes. That is to say S_j could represent the time when the execution of job j begins taking into account all possible arrivals of jobs. The same way, X_j could represent the processing time taking into account every possible preemption and execution of jobs in between. Then we can state that these two costs models are equivalent, making the only assumption that all patiences (or due

dates) are memoryless. □

The same theorem holds for IBS, assuming that the holding costs are paid per unit of time spent waiting for service. The proof is the same as for Theorem 20, replacing C_j by S_j, S_j by S_{j-1} and X_j by X_{j-1}.

Theorem 21. *Let us consider a n jobs scheduling problem with IBS. A Job j has an arbitrarily distributed processing time, an exponentially distributed patience time with mean $1/\gamma_j$ and an arbitrarily distributed release date. Then the two following costs models, 1) Job j has an abandonment cost w_j and, 2) Job j has a holding cost h_j per unit of time spent waiting for service, are equivalent. A type 1) (respectively type 2)) problem can be optimally controlled by using a type 2) (respectively type 1)) optimal rule by creating an instance with holding costs $h'_j = w_j\gamma_j$ (respectively abandonment costs $w'_j = h_j/\gamma_j$) for all j in $\{1, 2, \ldots, n\}$.*

C.3 Non markovian probability results

Lets assume that X_1 is an arbitrary random variable, that D_1 and D_2 are exponentially distributed with rate γ_1 and γ_2. In this section we express $\mathbb{P}(C_1 \geq D_2) = \mathbb{P}(\min(X, D_1) \geq D_2)$ using Laplace transforms. x_1, t_1 and t_2 are respectively the integration variables for f_{X_1}, f_{D_1} et f_{D_2}:

$$\begin{aligned}
\mathbb{P}(C_1 \geq D_2) &= \mathbb{P}(X_1 \geq D_2 \cap D_1 \geq D_2) \\
&= \int_{x_1=0}^{+\infty} \int_{t_2=0}^{x_1} \int_{t_1=t_2}^{+\infty} f_{X_1}(x_1) f_{D_2}(t_2) f_{D_1}(t_1) dt_1 dt_2 dx_1 \\
&= \int_{x_1=0}^{+\infty} \int_{t_2=0}^{x_1} f_{X_1}(x_1) f_{D_2}(t_2) e^{-\gamma_1 t_2} dt_2 dx_1.
\end{aligned}$$

Details for the inner integral:

$$\begin{aligned}
\int_{t_2=0}^{x_1} f_{D_2}(t_2) e^{-\gamma_1 t_2} dt_2 &= \int_{t_2=0}^{x_1} \gamma_2 e^{-(\gamma_1+\gamma_2)t_2} dt_2 \\
&= -\frac{\gamma_2}{\gamma_1+\gamma_2} \left[e^{-(\gamma_1+\gamma_2)t_2} \right]_{t_2=0}^{x_1} \\
&= \frac{\gamma_2}{\gamma_1+\gamma_2} \left(1 - e^{-(\gamma_1+\gamma_2)x_1} \right).
\end{aligned}$$

Using this expression:

$$\begin{aligned}
\mathbb{P}(C_1 \geq D_2) &= \int_{x_1=0}^{+\infty} \frac{\gamma_2}{\gamma_1+\gamma_2} \left(1 - e^{-(\gamma_1+\gamma_2)x_1} \right) f_{X_1}(x_1) dx_1 \\
&= \frac{\gamma_2}{\gamma_1+\gamma_2} \left[1 - \mathcal{L}\{f_{X_1}\}(\gamma_1+\gamma_2) \right].
\end{aligned}$$

Note that if $X_1 \sim \exp(\mu_1)$

$$\mathbb{P}(C_1 \geq D_2) = \frac{\gamma_2}{\gamma_1 + \gamma_2}\left[1 - \frac{\mu_1}{\mu_1 + \gamma_1 + \gamma_2}\right] = \frac{\gamma_2}{\mu_1 + \gamma_1 + \gamma_2},$$

which is consistent with previous expressions.

C.4 MDP formulation

The parameters that are used for the MDP formulation are: μ_j the service rate, λ_j the arrival rate, γ_j the impatience rate, w_j the abandonment cost and M_j the capacity of the queue.

We assume that the capacity of the queue is bounded. This assumption is made to keep a finite transition rate which is essential for mathematical tractability. This assumption allows us to treat every problem since the bounds can be set arbitrarily large.

In this section we give the MDP formulation of the problem when there are two classes of jobs in the system, since the numerical study is done in this setting. The formulation can easily be extended to n classes of jobs but this make the problem computationally hard to solve.

Since the system is markovian, the state space at time t depends only on the number of jobs in each class in the system at that time and is therefore denoted $x = (x_1, x_2)$ where x_j is the number of jobs from class j in the system. We also denote $e_1 = (1, 0)$ and $e_2 = (0, 1)$.

Let $v^*(x)$ be the expected optimal cost over an infinite horizon. As Lippman (1975), we consider event based dynamic programming, where events occur a rate $C = \sum_{j=1}^{2} \lambda_j + \mu_j + M_j\gamma_j$. The time horizon can be rescaled so that $C = 1$.

C.4.1 IES cost function

The optimal IES cost function respects

$$T_{IES}v(x) = T_{\text{prod}}v(x) + \sum_{j\in\{1,2\}} \Big(\gamma_j T^j_{\text{imp}}v(x) + \lambda_j T^j_{\text{arr}}v(x)\Big),$$

where operator T_{prod} corresponds to the production decision, T^j_{imp} to abandonments and T^j_{arr} to arrivals and they are defined as follows

$$T_{\text{prod}}v(x) = \begin{cases} \min \left\{ \begin{array}{l} \mu_1 v(x-e_1) + \mu_2 v(x); \\ \mu_1 v(x) + \mu_2 v(x-e_2); \\ (\mu_1+\mu_2)v(x) \end{array} \right\} & \text{if} \quad x_1 > 0,\, x_2 > 0 \\ \min \left\{ \begin{array}{l} \mu_1 v(x) + \mu_2 v(x-e_2); \\ (\mu_1+\mu_2)v(x) \end{array} \right\} & \text{if} \quad x_1 = 0,\, x_2 > 0 \\ \min \left\{ \begin{array}{l} \mu_1 v(x-e_1) + \mu_2 v(x); \\ (\mu_1+\mu_2)v(x) \end{array} \right\} & \text{if} \quad x_1 > 0,\, x_2 = 0 \\ (\mu_1+\mu_2)v(x) & \text{if} \quad x_1 = 0,\, x_2 = 0 \end{cases}$$

$$T^j_{\text{imp}}v(x) = \begin{cases} x_j(v(x-e_j) + w_j) + (M_j - x_j)v(x) & \text{if} \quad x_j > 0 \\ M_j v(x) & \text{if} \quad x_j = 0 \end{cases}$$

$$T^j_{\text{arr}}v(x) = \begin{cases} v(x+e_j) & \text{if} \quad x_j < M_j \\ v(x) & \text{if} \quad x_j = M_j. \end{cases}$$

C.4.2 IBS cost function

The optimal IBS cost function respects

$$T_{IBS}v(x) = \tilde{T}_{\text{prod}}v(x) + \sum_{j\in\{1,2\}} \Big(\gamma_j \tilde{T}^j_{\text{imp}}v(x) + \lambda_j T^j_{\text{arr}}v(x)\Big),$$

where operator $\tilde{T}_{\text{prod}}$ corresponds to the production decision, $\tilde{T}^j_{\text{imp}}$ to abandonments and T^j_{arr} to arrivals and they are defined as follows

$$\tilde{T}_{\text{prod}}v(x)$$

$$= \begin{cases} \min \begin{cases} \mu_1 v(x-e_1) + \gamma_1 v(x) + \mu_2 v(x) + \gamma_2(v(x-e_2)+w_2); \\ \mu_1 v(x) + \gamma_1(v(x-e_1)+w_1) + \mu_2 v(x-e_2) + \gamma_2 v(x); \\ (\mu_1+\mu_2)v(x) + \gamma_1(v(x-e_1)+w_1) + \gamma_2(v(x-e_2)+w_2) \end{cases} & \text{if} \quad x_1 > 0,\, x_2 > 0 \\ \min \begin{cases} \mu_1 v(x) + \gamma_1 v(x) + \mu_2 v(x-e_2) + \gamma_2 v(x); \\ (\mu_1+\mu_2)v(x) + \gamma_1(v(x-e_1)+w_1) + \gamma_2(v(x-e_2)+w_2) \end{cases} & \text{if} \quad x_1 = 0,\, x_2 > 0 \\ \min \begin{cases} \mu_1 v(x-e_1) + \gamma_1 v(x) + \mu_2 v(x) + \gamma_2(v(x-e_2)+w_2); \\ (\mu_1+\mu_2)v(x) + \gamma_1(v(x-e_1)+w_1) + \gamma_2(v(x-e_2)+w_2) \end{cases} & \text{if} \quad x_1 > 0,\, x_2 = 0 \\ (\mu_1+\mu_2)v(x) + \gamma_1(v(x-e_1)+w_1) + \gamma_2(v(x-e_2)+w_2) & \text{if} \quad x_1 = 0,\, x_2 = 0 \end{cases}$$

$$\tilde{T}^j_{\text{imp}}v(x)$$

$$= \begin{cases} (x_j - 1)(v(x-e_j)+w_j) + (M_j - x_j)v(x) & \text{if} \quad x_j > 0 \\ M_j v(x) & \text{if} \quad x_j = 0 \end{cases}$$

$$T^j_{\text{arr}}v(x)$$

$$= \begin{cases} v(x+e_j) & \text{if} \quad x_j < M_j \\ v(x) & \text{if} \quad x_j = M_j. \end{cases}$$

Annexe D

Appendix of Chapter 5

D.1 Proof of Section 5.2.1

In this section, we present the proof that the impatience operator and the arrival operator introduced in Section 5.2.1 (page 93) propagate the two following properties

Property 1: $\Delta_{e_x} v(x,u) \geq 0, \forall x \geq 0, \forall i \in \{0,1\}$
with $\Delta_{e_x} v(x,u) = v(x+1,u) - v(x,u)$;

Property 2: $\Delta_{e_x,e_u} v(x,0) \geq 0, \forall x \geq 0$
with $\Delta_{e_x,e_u} v(x,0) = \Delta_{e_x}\Delta_{e_u} v(x,0) = v(x+1,1) - v(x+1,0) - v(x,1) + v(x,0)$.

We assume that v satisfies Property 1 and Property 2 and we prove that $T_{ab}v(x,u)$ and $T_{arr}v(x,u)$ satisfies these properties as well.

D.1.1 Impatience operator

Property 1

If $x = 0$, then

$$\begin{aligned}\Delta_{e_x} T_{ab} v(0,u) &= T_{ab}v(1,u) - T_{ab}v(0,u) \\ &= v(0,u) + (M-1)v(1,u) - Mv(0,u) \\ &= (M-1)\Delta_{e_x} v(0,u) \geq 0.\end{aligned}$$

If $0 < x < M$, then

$$\begin{aligned}\Delta_{e_x} T_{ab} v(x,u) &= T_{ab}v(x+1,u) - T_{ab}v(x,u) \\ &= (x+1)v(x,u) + (M-x-1)v(x+1,u) \\ &\quad - xv(x-1,u) - (M-x)v(x,u) \\ &= (M-x-1)\Delta_{e_x} v(x,u)) + x\Delta_{e_x} v(x-1,u) \geq 0.\end{aligned}$$

Thus, the impatience operator propagates Property 1.

Property 2

If $x = 0$, then

$$\begin{aligned}\Delta_{e_x,e_u} T_{ab} v(0,0) &= T_{ab} v(1,1) - T_{ab} v(0,1) - T_{ab} v(1,0) + T_{ab} v(0,0) \\ &= v(0,1) + (M-1)v(1,1) - Mv(0,1) \\ &\quad - v(0,0) - (M-1)v(1,0) + Mv(0,0) \\ &= (M-1)\Delta_{e_x,e_u} v(0,0) \geq 0.\end{aligned}$$

If $0 < x < M$, then

$$\begin{aligned}\Delta_{e_x,e_u} T_{ab} v(x,0) &= T_{ab} v(x+1,1) - T_{ab} v(x,1) - T_{ab} v(x+1,0) + T_{ab} v(x,0) \\ &= (x+1)v(x,1) + (M-x-1)v(x+1,1) \\ &\quad - xv(x-1,1) - (M-x)v(x,1) \\ &\quad - (x+1)v(x,0) - (M-x-1)v(x+1,0) \\ &\quad + xv(x-1,0) + (M-x)v(x,0) \\ &= (M-x-1)\Delta_{e_x,e_u} v(x,0) + x\Delta_{e_x,e_u} v(x-1,0) \geq 0.\end{aligned}$$

Thus, the impatience operator propagates Property 2.

D.1.2 Arrival operator

Property 1

If $x = M - 1$, then

$$\begin{aligned}\Delta_{e_x} T_{arr} v(M-1,u) &= T_{arr} v(M,u) - T_{arr} v(M-1,u) \\ &= v(M,u) - v(M,u) = 0.\end{aligned}$$

If $0 \leq x < M - 1$, then

$$\begin{aligned}\Delta_{e_x} T_{arr} v(x,u) &= T_{arr} v(x+1,u) - T_{arr} v(x,u) \\ &= v(x+2,u) - v(x+1,u) \\ &= \Delta_{e_x} v(x+1,u) \geq 0.\end{aligned}$$

Thus, the arrival operator propagates Property 1.

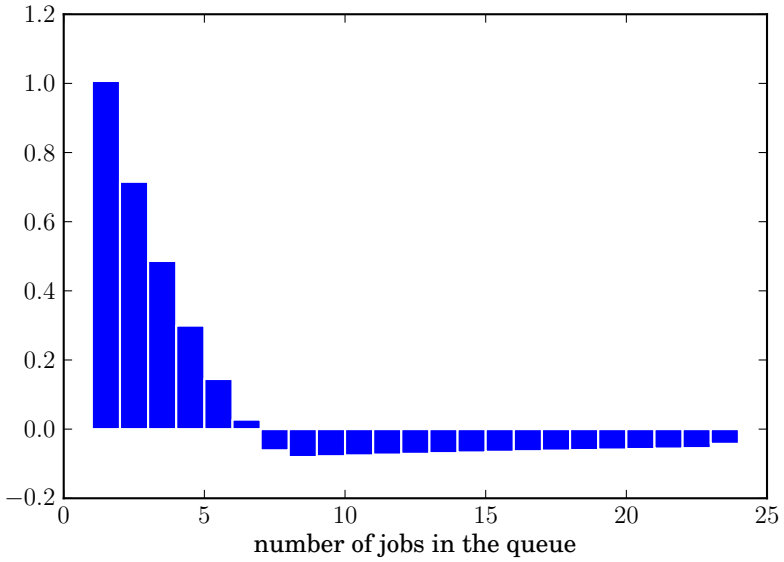


Figure 29: Value of $v(x-1,1)-v(x,0)$ for the instance $M=25$, $\mu_1=3$, $\mu_2=0.5$, $\lambda=2$ and $\gamma=1$

Property 2

If $x=M-1$, then

$$\begin{aligned}\Delta_{e_x,e_u}T_{arr}v(M-1,0) &= T_{arr}v(M,1)-T_{arr}v(M-1,1)-T_{arr}v(M,0)+T_{arr}v(M-1,0)\\ &= v(M,1)-v(M,1)-v(M,0)+v(M,0)=0.\end{aligned}$$

If $0\leq x<M-1$, then

$$\begin{aligned}\Delta_{e_x,e_u}T_{arr}v(x,0) &= T_{arr}v(x+1,1)-T_{arr}v(x,1)-T_{arr}v(x+1,0)+T_{arr}v(x,0)\\ &= v(x+2,1)-v(x+1,1)-v(x+2,0)+v(x+1,0)\\ &= \Delta_{e_x,e_u}v(x+1,0)\geq 0.\end{aligned}$$

Thus, the arrival operator propagates Property 2.

D.2 Additionnal figures of Section 5.2.1

The additional figures of Section 5.2.1, page 97, are Figure 29, which is the untruncated version of Figure 24, and Figure 30, which is the untruncated version of Figure 25.

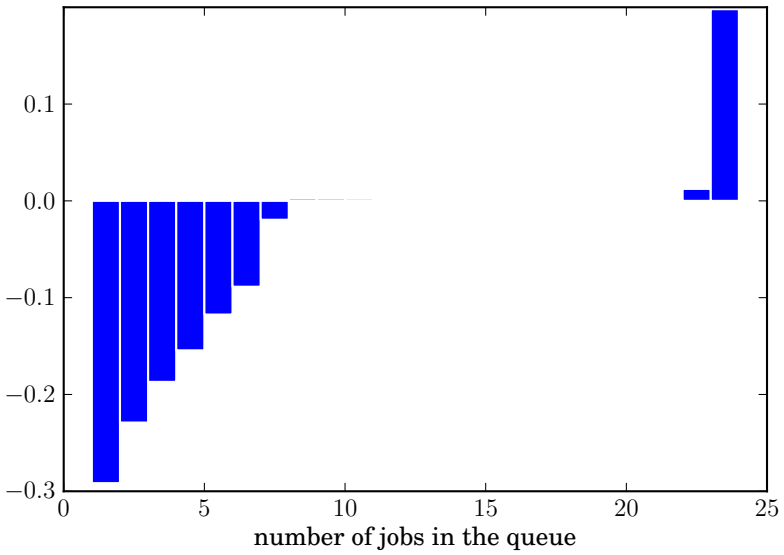


Figure 30: Value of $\Delta_{e_x,-e_x+e_u} v(x+1,0)$ for the instance $M = 25$, $\mu_1 = 3$, $\mu_2 = 0.5$, $\lambda = 2$ and $\gamma = 1$

D.3 MDP formulation without preemption

The parameters that are used for the MDP formulation are: μ_i the service rate, λ_i the arrival rate, γ_i the impatience rate, w_i the abandonment cost and M_i the capacity of the queue.

Again, we assume that the capacity of the queue is bounded. This assumption is made to keep a finite transition rate which is essential for mathematical tractability, but this is not restrictive since the bounds can be set arbitrarily large.

In this section we give the MDP formulation of the problem when there are two classes of jobs in the system and the process is not preemptive.

Since the system is markovian, the state space at time t depends only on the number of jobs from each class in the system at that time and which class of job is in process. The state of the system is therefore denoted (x, u), where $x = (x_1, x_2)$ and x_i is the number of jobs from class i waiting in the system for process and $u \in \{0, 1, 2\}$ denotes the state of the machine. If $u = 0$, the machine is idling and if $u = j$ the machine is processing a job from class j, $j \in \{1, 2\}$. We also denote $e_1 = ((1, 0), 0)$, $e_2 = ((0, 1), 0)$, $u_1 = ((0, 0), 1)$ and $u_2 = ((0, 0), 2)$.

Let $v^*(x, u)$ be the expected optimal cost over an infinite horizon. We consider event based dynamic programming (Koole 2006), where events occur at a rate $C = \sum_{i=1}^{2} \lambda_i + \mu_i + (M_i + 1)\gamma_i$. The time horizon can always be rescaled so that $C = 1$, then the optimal

IES cost function respects

$$T_{IES}v(x,u) = T_{\mathrm{d}} \circ \left(\sum_{j\in\{1,2\}} \left(\mu_j T^j_{\mathrm{prod}} v(x,u) + \gamma_j T^j_{\mathrm{imp}} v(x,u) + \lambda_j T^j_{\mathrm{arr}} v(x,u) \right) \right),$$

where operator T_{d} correspond to the decision of processing a job from class 1 or from class 2 (such a decision can only be made after an event, when the machine idles, or after the end of a process), T_{prod} corresponds to the processing, T^j_{imp} to abandonments and T^j_{arr} to arrivals and they are defined as follows

$$T_{\mathrm{d}}v(x,u) = \begin{cases} \min\{v(x,0); v(x-e_1,1); v(x-e_2,2)\} & \text{if} \quad u=0 \\ v(x,u) & \text{otherwise} \end{cases}$$

$$T^j_{\mathrm{prod}}v(x,u) = \begin{cases} (\mu_j+\gamma_j)v(x,0) + \gamma_j w_j & \text{if} \quad u=j \\ (\mu_j+\gamma_j)v(x,u) & \text{otherwise} \end{cases}$$

$$T^j_{\mathrm{imp}}v(x,u) = \begin{cases} x_i(v(x-e_i,u)+w_i) + (M_i-x_i)v(x,u) & \text{if} \quad x_i>0 \\ M_i v(x,u) & \text{if} \quad x_i=0 \end{cases}$$

$$T^j_{\mathrm{arr}}v(x,u) = \begin{cases} v(x+e_i,u) & \text{if} \quad x_i<M_i \\ v(x,u) & \text{if} \quad x_i=M_i. \end{cases}$$

Résumé

Le cadre de cette thèse est l'étude de systèmes de production avec impatience. Ces systèmes sont modélisés comme des problèmes d'ordonnancement stochastiques avec des dates d'échéance. Dans la littérature, peu de résultats existent sur le contrôle optimal de ce genre de systèmes. C'est dans ce cadre que s'inscrit cette thèse. Nous considérons un système générique avec une machine, sur laquelle des tâches sont à exécuter. Les durées d'exécution, les dates d'échéance (ou durées d'impatience) et les dates de disponibilité des tâches sont des variables aléatoires. À chaque tâche est associé un poids et l'objectif est de minimiser l'espérance du nombre pondéré de tâches en retard.

Dans notre étude, nous utilisons différentes modélisations, rendant compte des différentes contraintes régissant des systèmes réels. Notamment, nous faisons la différence entre l'impatience, le fait d'avoir attendu trop longtemps, et l'abandon, le fait de quitter le système suite à l'impatience. Dans la classe des politiques statiques, nous donnons des ordonnancements optimaux pour des problèmes avec impatience. Dans la classe des politiques dynamiques avec préemption, nous donnons de nouvelles conditions garantissant l'optimalité d'une politique stricte pour des problèmes avec abandon. Nous proposons aussi une heuristique plus efficace que celles que l'on trouve dans la littérature. Enfin, nous explorons des variantes et des extensions de ces problèmes, lorsque le système comporte plusieurs machines et lorsque la préemption n'est pas autorisée.

Abstract

In this thesis, production systems facing abandonments are studied. These problems are modeled as stochastic scheduling problems with due dates. In the literature, few results exist concerning the optimal control of such systems. This thesis aim at providing optimal control policies for systems with impatience. We consider a generic system with a single machine, on which jobs have to be processed. Processing times, due dates (or patience time) and release dates are random variables. A weight is associated to each job and the objective is to minimize the expected weighted number of late jobs.

In our study, we use different models, taking into account the specific features of real life problems. For example, we make a difference between impatience, when a customer has been waiting for too long, and abandonment, when a customer leaves the system after getting impatient. In the class of static list scheduling policies, we provide optimal schedules for problems with impatience. In the class of preemptive dynamic policies, we specify conditions under which a strict priority rule is optimal and we give a new heuristic, both extending previous results from the literature. We study variants and extensions of these problems, when several machines are available or when preemption is not authorized.