Ordonnancement de tâches pour concilier la minimisation de la consommation d'énergie avec la qualité de service : optimisation et théorie des jeux.

Oscar Carlos Vasquez Perez

THÈSE DE DOCTORAT DE l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

INFORMATIQUE

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

M. Óscar Carlos VÁSQUEZ PÉREZ

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Ordonnancement de tâches pour concilier la minimisation de la consommation d'énergie avec la qualité de service: optimisation et théorie des jeux

soutenue le 23 janvier 2013, devant le jury composé de :

M. Christoph Dürr	Directeur de thèse
M. Rida Laraki	Rapporteur
M. Monaldo Mastrolilli	Rapporteur
M. Laurent Gourvès	Examinateur
Mme. Safia Kedad-Sidhoum	Examinatrice
M. Ameur Soukhal	Examinateur

Remerciements

Tout d'abord, je tiens à exprimer toute ma gratitude à mon directeur de thèse, Christoph Dürr. Il m'émerveille à tous moments par son intelligence, créativité, virtuosité et qualité humaine, qui m'a permis de travailler toutes ces années dans une ambiance à la fois scientifique rigoureuse et chaleureuse dans l'humain. Je n'aurais jamais persisté jusqu'au bout de ce travail sans son inlassable épaulé, patience, dévouement et disponibilité au long de cette thèse. Ma profonde et ma sincère reconnaissance pour l'effort et l'enthousiasme qu'il m'est fourni.

Je suis très reconnaissante envers Rida Laraki et Monaldo Mastrolilli qui ont accepté d'être les rapporteurs de cette thèse, et je veux particulièrement exprimer ma gratitude pour que cette thèse puisse être soutenue dans les meilleurs délais. Je remercie vivement à Safia Kedad-Sidhoum, Laurent Gourvès et Ameur Soukhal, c'est un honneur pour moi qu'ils fassent partie de mon jury de thèse.

Je remercie également à la *Comisión Nacional de Investigación Científica y Tecnológica de Chile (CONICYT)* pour ma bourse de doctorat, grâce à laquelle mes années de thèse ont pu être financés. Je veux remercier aussi à tous les membres de l'équipe RO (LIP6) de m'avoir accueilli dans une ambiance amicale et chaleureuse. Merci !

Je tiens aussi à remercier des nombreuses personnes qui ont été dans mon chemin pendant mon séjour en France. Eduardo, Florence, Francisco (Xico), Jimena, Emilio, Alma, Jack, Jorge (Zambra), Jaime (Jimi), Damien, Cristina, André, Silvia ... Merci beaucoup pour donner celles belles notes de musique dans la chanson de ma vie.

De plus, je veux exprimer ma sincère gratitude au *Departamento de Ingeniería Industrial de la USACH*. En particulier, mes remerciements aux professeurs Miguel Alfaro, Mario Tarride et Juan Sepúlveda pour tout le soutien qu'ils m'ont apportés avant et pendant mon doctorat.

Je ne trouverai jamais les mots pour remercier à ma famille pour leur constant soutien et affection. *A mi tios, tias, primos, primas, abuelita y mamá, simplemente muchas gracias por existir, por estar conmigo.*

Enfin, je remercie à Larissa Conceição dos Santos. *Obrigado pela beleza do silêncio na canção de minha vida, esse silêncio de apoio e entendimento, esse silêncio de um beijo apaixonado de nossas noites de Paris, esse silêncio no meu interior quando penso em você, esse silêncio que vale mais que mil palavras.*

Résumé

Cette thèse est consacrée au problème d'ordonnancement de tâches qui consiste à minimiser la somme de l'énergie consommée et le temps d'attente pondéré total, et l'aborde sous deux différents points de vue : centralisé et décentralisé. Pour l'approche décentralisée, nous avons défini deux types de jeux qui diffèrent dans les actions proposées aux joueurs et avons cherché des moyens de facturer l'énergie consommée aux utilisateurs pour les inciter à adopter un bon comportement. Concrètement nous nous intéressons à l'existence d'équilibres de Nash purs, au temps de convergence vers ces équilibres, et au rapport entre l'énergie consommée et le montant des factures. Pour l'approche centralisée, nous avons réduit le problème de minimisation à un problème d'ordonnancement plus classique avec une fonction de pénalité de retard polynomiale concave, pour lequel peu de résultats sont connus. Après avoir établi un état de l'art sur la famille de problèmes d'ordonnancement pour plusieurs fonctions de pénalité élémentaires et montré qu'une technique de preuve de NP-complétude classique échoue ici, nous nous sommes intéressés à sa résolution exacte. Pour améliorer les performances de l'algorithme A* dans ce contexte, nous avons montré des résultats de règles de dominance. Concrètement, nous avons cherché à déterminer les conditions sous lesquelles une solution optimale devrait ordonnancer une paire de tâches dans un certain ordre. Ces résultats s'appuient sur une étude expérimentale qui évalue l'impact pratique de ces nouvelles règles, par rapport aux règles existantes.

Mots-clé :

gestion de l'énergie, ordonnancement, optimisation, théorie des jeux algorithmique, qualité de service, complexité, règles de dominance, algorithme A*.

Abstract

This thesis focuses on a job scheduling problem with the goal of minimizing the sum of energy consumption and the weighted flow time from two different approaches: centralized and decentralized. In the decentralized setting, we defined two games which differ in the strategies players can choose from and designed cost sharing mechanisms, charging the consumed energy to the users in order to incentive a socially desirable behavior. More precisely we were interested in the existence of pure Nash equilibria, in the convergence time, and the ratio between the consumed energy and the total charged amount. On the other side, for the centralized approach, we reduced the minimization problem to a classical scheduling problem with a polynomial concave penalty function, for which little results were known. We established a state of the art for a family of scheduling problems of this form with different penalty functions and showed that a classical NP-completeness proof technique fails here. Finally we addressed the exact resolution of the problem using the algorithm A*. In this context, we showed new order dominance rules. More precisely, we characterized the conditions under which any optimal solution must schedule a job pair in a certain order. In addition we carried out a computational experience to evaluate the practical impact of these new rules compared to the existing ones.

Keywords

energy management, scheduling, optimization, algorithmic game theory, quality of service, complexity, dominance rules, algorithm A*.

Table des matières

Chapitre 1
Introduction (français)

1.1 Le contexte

Les avancées technologiques ont placé les processeurs dans presque tous les appareils utilisés dans la vie courante du monde occidental, augmentant d'une certaine manière la qualité de vie de l'humanité. Tablettes, notebooks, i-pads, i-pods, en sont de bons exemples. Cependant, l'usage de ces technologies implique une demande croissante en temps de calcul, qui ont fait du secteur informatique un important consommateur en ressources énergétiques au niveau mondial.

Beaucoup d'efforts sont maintenant faits pour limiter l'impact écologique de ces nouvelles technologies. Pour minimiser la consommation d'énergie d'un processeur on peut intervenir à bien des niveaux, et un des aspects est l'ordonnancement des tâches à exécuter. Il s'agit de concevoir des politiques d'ordonnancement qui minimisent la consommation d'énergie tout en garantissant une certaine qualité de service aux consommateurs. Ces politiques sont importantes aussi bien pour des grands centres de calcul qui veulent réduire leur facture d'électricité, que pour des systèmes embarqués qui veulent augmenter l'autonomie énergétique.

L'environnement de calcul étudié concerne les systèmes informatiques avec des microprocesseurs de dernière génération (e.g. Intel SpeedStep, AMD PowerNow ! ou IBM EnergyScale) qui peuvent varier dynamiquement la fréquence de calcul et influencer ainsi à la fois sur la qualité de service, et la consommation d'énergie. L'aspect principal de ces systèmes est que la consommation d'énergie est une fonction convexe qui depend de la vitesse de travail. Ceci est aussi le cas pour des centres de calcul ou des sites de production, où pour satisfaire une demande croissante en travail, il faut faire appel à des ressources supplémentaires, qui ont un coût de production plus élevé.

Cette thèse concerne des systèmes informatiques décentralisés, où des utilisateurs non-coopératifs soumettent leur tâches à un processeur capable de varier librement la vitesse de calcul, et qui doit alors concilier sa propre consommation d'énergie avec la qualité de service rendu. Ceci reflète les modèles économiques adoptés par un nombre croissant d'éditeurs de logiciels, qui proposent non pas des logiciels à installer localement mais mettent à disposition une interface web pour un calcul distant dans le *nuage*. Mais cette situation apparaît aussi dans les smartphones et les ordinateurs personnels, où des applications développées par différents éditeurs tournent en concurrence sur une même machine.

1.1.1 Le modèle d'ordonnancement pour la minimisation de la consommation d'énergie

Les problèmes d'ordonnancement forment une large classe de problèmes d'optimisation, qui ont été étudiés depuis une cinquantaine d'années, et comme tout domaine de recherche comportent des résultats classiques, des fameux problèmes ouverts et sans cesse de nouveaux problèmes posés par des nouveaux domaines d'application.

Un de ces nouveaux domaines d'application est l'ordonnancement pour la minimisation d'énergie, dans le cadre d'un processeur capable de varier sa vitesse. Le modèle simplifié est le suivant. Plutôt que de travailler à une vitesse constante, ce nouveau type de processeur est capable de fonctionner avec une vitesse variable s, où $s(t)$ est la vitesse de travail au moment t. Dans ce contexte dans un intervalle de temps I le travail effectué est tout simplement $\int_{t\in I} s(t)dt$, qui représente le nombre d'instructions exécutées pendant I. En même temps la consommation d'énergie est proportionnelle à $\int_{t\in I} s(t)^\alpha dt$ pour une constante physique $2 \leqslant \alpha \leqslant 3$ donnée [11]. Cette consommation d'énergie est liée à la dissipation de chaleur sur la surface du processeur, qui nécessite un refroidissement continu pour ne pas endommager la machine. D'autres modèles d'ordonnancement de minimisation d'énergie ont été considérés, qui font intervenir différents modes de la machine (inactif, suspension, hibernation, éteint), ou qui considèrent des machines parallèles. Dans cette thèse, nous nous sommes restreints au cas simple d'un processeur à vitesse variable.

L'entrée d'un problème d'ordonnancement consiste généralement en n tâches. Chaque tâche i a une quantité de travail w_i, qui représente le nombre d'instructions générées par son exécution. Mais typiquement la tâche i vient aussi avec une fenêtre de temps $[r_i, d_i)$ dans laquelle elle doit être exécutée. Le temps r_i est un temps de relâchement avant lequel la tâche n'est pas encore disponible et le temps d_i est la date limite de la tâche, qui représente une garantie sur le temps de complétude de la tâche que l'ordonnancement doit respecter.

Un ordonnancement est alors spécifié par deux fonctions, la vitesse $s : \mathbb{R}_+ \to \mathbb{R}_+$, et une affectation job: $\mathbb{R}_+ \to \{\text{idle}, 1, \ldots, n\}$. L'idée est que la tâche i est exécutée pendant tous les instants t tel que $job(t) = i$. Ainsi, le problème — devenu classique dans le domaine de la minimisation d'énergie — consiste à produire un ordonnancement pour n tâches données, qui respecte les fenêtres d'exécutions et qui minimise la consommation d'énergie.

Après la description d'un premier algorithme polynomial pour ce problème en 1995 [63], plusieurs articles ont proposé des améliorations. Actuellement le meilleur algorithme connu a une complexité de $O(n^2 \log n)$ [43] et pour le cas particulier où les tâches n'ont pas de date de relâchement — autrement dit $r_i = 0$ pour tout i — il existe un algorithme en temps $O(n \log n)$.

Les ingrédients à ces algorithmes sont les suivants. La convexité de la fonction de consommation d'énergie $s \mapsto \int_{t\in I} s(t)^\alpha dt, \alpha \in [2, 3]$ implique que pendant tous les instants où une tâche i est exécutée, la vitesse est la même. La preuve de cette proposition est subtile car l'objet de la minimisation, les fonctions s et job décrivant l'ordonnancement sont continues, et donc la preuve est basée sur les conditions Karush-Kuhn-Tucker (KKT) [9]. Par contre si on suppose que ces fonctions sont constantes par morceau, un simple argument de nivellement suffit.

Le deuxième ingrédient considère la densité des intervalles. La densité d'un intervalle I est définie comme la somme de la charge de travail sur toutes les tâches qui doivent s'exécuter en I, le tout divisé par la longueur de I. Il est clair que la vitesse moyenne en I d'un ordonnancement doit être au moins cette densité, car sinon la quantité de travail effectuée est insuffisante

pour ces tâches. La clé pour obtenir un algorithme polynomial est l'observation que dans un ordonnancement optimal, la vitesse pendant un intervalle I de densité maximale, est tout simplement de manière constante à cette même densité. Le principe d'un algorithme polynomial est alors d'identifier un tel intervalle I de densité maximale, d'ordonnancer les tâches forcées en I à vitesse constante, puis d'ignorer I et les tâches affectées à et réitérer.

L'ordonnancement des tâches dans un intervalle I est effectué utilisant la politique *Earliest Deadline First* (EDF), qui consiste à tout moment t du début de I à la fin, d'ordonnancer la tâche i la plus urgente (avec la plus petite date limite d_i) parmi les tâches déjà relâchées et pas encore complétées. Si la vitesse a été choisie comme décrite ci-dessus, un tel ordonnancement respectera toujours les dates limites. Sinon une date limite d_j dépassée pourrait être associée à un intervalle de la forme $[r_i, d_j)$ de densité strictement plus grande que I, contredisant le choix de I. La Figure 1.1 montre un exemple d'un ordonnancement qui minimise la consommation d'énergie.

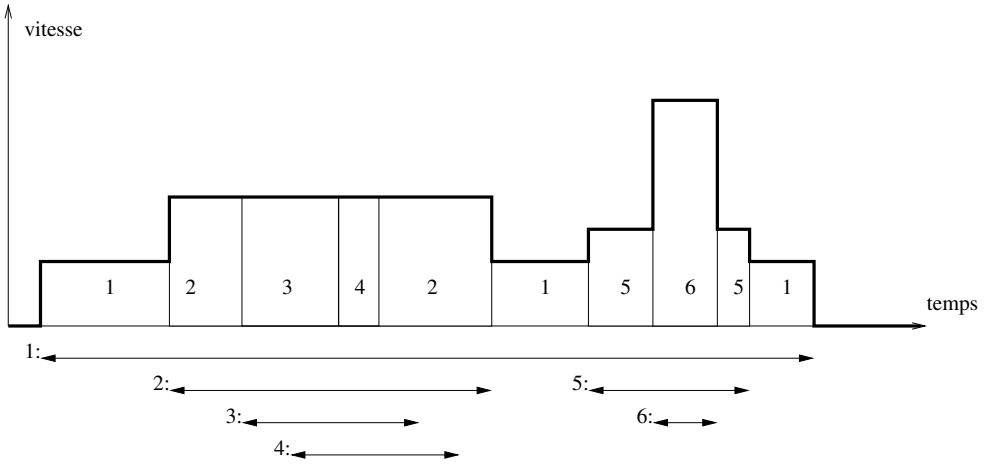


FIGURE 1.1 – Exemple d'un ordonnancement qui minimise la consommation d'énergie pour cinq tâches. Les flèches indiquent l'intervalle $[r_j, d_j]$ pour chaque tâche j, la ligne en gras indique la vitesse $s(t)$ et les étiquettes dans les rectangles indiquent l'affectation $job(t)$.

1.2 Présentation générale et objectifs

Formellement, nous abordons dans cette thèse le problème d'ordonnancement bi-critère suivant. Étant donné un processeur avec vitesse variable et un nombre de tâches données, nous voulons à la fois minimiser l'énergie consommée et le temps d'attente moyen des tâches. Notre objectif alors est de minimiser une somme pondérée de ces quantités, et nous pondérons les temps d'attente des tâches avec un facteur de pénalité donnée. Nous définissons cette quantité comme *le coût social* à optimiser, et l'abordons sous différents angles.

D'une part, nous considérons le problème décentralisé qui consiste à définir des mécanismes de paiement aux bonnes propriétés et d'autre part nous considérons le problème centralisé d'optimisation avec des résultats structurels et une implémentation permettant une comparaison expérimentale.

Spécifiquement, nous abordons le problème sous forme d'un jeu, où chaque joueur possède une tâche qu'il soumet avec certaines informations à une machine. Cette machine produit un

ordonnancement avec ces tâches. Nous sommes alors devant un conflit d'intérêt intéressant. Les joueurs veulent une chose, que leur tâche termine le plus tôt possible. C'est tout à fait naturel, qui aime bien attendre ? Par contre le machine veut une autre chose, que l'ordonnancement produit par le processeur consomme le moins d'énergie possible. C'est aussi naturel, pour des raisons économiques et écologiques.

Dans ce contexte nous cherchons alors à facturer l'énergie consommée aux utilisateurs afin de les sensibiliser à cet aspect de l'ordonnancement. On parle alors de schéma ou de mécanisme de facturation, et le domaine de recherche qui étudie ces schémas s'appelle conception de mécanismes, *mechanism design* en anglais.

Du point de vue du joueur la situation est la suivante. Chaque joueur veut minimiser une combinaison pondérée du temps de complétude de sa tâche et du montant qui lui est facturée. C'est la clé d'un mécanisme de facturation, pour pousser les utilisateurs à avoir un bon comportement pour la société, on doit les faire participer au coût social pour les inciter à minimiser ce coût. Afin de permettre une somme de deux types de quantités différentes, temps et énergie, on les convertit implicitement en une quantité monétaire dont les coefficients sont cachés dans les poids de la combinaison linéaire. Essentiellement chaque joueur i veut minimiser la somme $p_i C_i + b_i$, où p_i est une pondération de priorité du joueur i, C_i est le temps de complétude de sa tâche et b_i est le montant de la facture pour i (b pour *bill*).

Nous sommes alors dans un modèle d'ordonnancement simple où la valeur à minimiser par les joueurs est linéaire en la date de complétude pondérée de sa tâche. Dans la littérature d'ordonnancement bien d'autres fonctions objectives ont été considérées, comme par exemple des fonctions à seuil quand C_i dépasse une certaine date. Pour cette thèse nous nous sommes restreints à cette pénalité linéaire, appelé *flow time* en anglais, et qui est une des fonctions objectives les plus étudiée, d'après des statistiques sur les requêtes de la bibliographie en ligne The Scheduling Zoo [19].

Ce type de jeu s'appèle un *jeu stratégique*. Contrairement à un jeu répété, comme par exemples les jeux de société, les échecs pour citer un exemple, ce jeu se joue en un seul tour. Les joueurs soumettent leurs tâches, avec des valeurs choisies. Ces valeurs s'appellent des *stratégies*. Toutes ces valeurs, et les paramètres des tâches sont publics et donc connus par les autres joueurs, bien qu'il puisse exister des valeurs privées non communiquées.

La machine calcule un ordonnancement pour toutes ces tâches et facture l'énergie consommée aux utilisateurs. L'algorithme utilisé pour ce calcul et le mécanisme de facturation est connu par tous les joueurs. Ainsi un joueur qui connait le choix des autres joueurs peut décider de la meilleure stratégie à adopter. Pour être précis, il dispose de toutes les informations pour cette optimisation, et alors nous nous préoccupons des propriétés qui sont souhaitables d'un tel jeu.

- D'abord il est souhaitable que des *équilibres de Nash purs* existent. C'est à dire on veut que pour un ensemble de joueurs donnés, il existe des stratégies pour chacun (appelées *profil de stratégies*) tel qu'aucun joueur ne veut en dévier. En d'autres termes la stratégie du joueur i minimise sa pénalité quand les stratégies des autres joueurs sont fixées. Une telle stratégie s'appelle *meilleure réponse*. Il est a noter que dans cette définition, on ne considère pas les changements de stratégies simultanés d'un groupe de joueurs, qui s'appellerait une *coalition*. Le terme *pur* est choisi par distinction avec les équilibres *mixtes*, où les joueurs peuvent jouer une distribution probabiliste de stratégies. Le théorème de Nash dit que des équilibres de Nash mixtes existent toujours pour des jeux stratégiques finis, alors que l'existence d'équilibre de Nash pur n'est pas toujours

établi.

- Dans le cas où des équilibres de Nash pur existent, on aimerait qu'il soit possible de les trouver en temps polynomial en fonction du nombre de joueurs. Aussi il est souhaitable que le problème du calcul de la meilleure réponse pour un joueur soit calculable en temps polynomial. Toutes ces restrictions sont clairement importantes pour un jeu réaliste.

- Différents équilibres de Nash peuvent avoir des qualités différentes. Pour une fonction de coût social donnée, on aimerait que le coût social d'un équilibre de Nash soit aussi près que possible de l'optimum social. Il est a noter que l'optimum social n'est peut être pas un équilibre. Une garantie sur le rapport entre ces deux quantités s'appèle le *prix d'anarchie*. Une autre quantité liée au prix d'anarchie est le *prix de stabilité*. C'est une borne supérieure entre le rapport du meilleur équilibre de Nash et l'optimum social. Cette quantité concerne les équilibres qui sont suggérés par le régulateur aux joueurs, plutôt que des laisser trouver librement un équilibre. Les deux mesures ont été introduites par Koutsoupias et Papadimitriou et étudiées pour un grand nombre de jeux depuis, voir les livres [21, 49]. Dans le contexte des mécanismes de facturation un autre terme est utilisé, on dit que le jeu est υ-efficace si tout équilibre de Nash a un coût social au plus υ fois l'optimum.

- D'un point de vue du régulateur de jeu, on veut que le montant total facturé aux joueurs correspond à l'énergie consommée. Dans le cas où ceci n'est pas possible on veut que le montant s'en approche le plus possible. On dit que le mécanisme surfacture à facteur γ — γ-*budget balanced* en anglais — si le montant total des factures est au moins l'énergie consommée et au plus γ fois cette quantité.

- Dans le cas où il existe des informations privées dans le jeu, et que leurs valeurs annoncées par les joueurs peuvent dévier des *vraies* valeurs privées, on aimerait qu'annoncer la véritable valeur soit une meilleure réponse. On dit aussi que cette strategie est dominante. Dans ce cas le jeu est dit être avec véracité garantie, en anglais *truthful* ou *strategy proof*. Cette notion deviendra plus claire avec un exemple par la suite. Une telle propriété a un intérêt pratique pour les joueurs, qui ont alors un choix très simple pour leur stratégie. Et elle est primordiale pour le régulateur, qui peut alors optimiser le coût social, ce qui est impossible si des données ne sont pas vraies.

Dans le context centralisé, nous abordons le problème consistant à produire un ordonnancement minimisant une combinaison linéaire entre la consommation d'énergie et les temps de complétude pondérés total des tâches. Il s'avère que ce problème a la forme d'un problème d'ordonnancement classique, mais avec une fonction objective peu étudiée. Pour cerner la complexité de ce problème nous étudions la complexité pour différentes fonctions, afin de dégager des techniques de preuve de NP-complétude et des réductions menant vers des schémas d'approximation. Puis en l'absence d'algorithmes polynomiaux pour notre problème d'ordonnancement, nous analysons des propriétés de dominance sur l'ordre des tâches pour obtenir de nouvelles régles et développons une étude expérimentale pour évaluer son impact.

1.3 Organisation de la thèse

La thèse est composée de 5 chapitres. Ce chapitre 1, est une introduction et une présentation succincte des résultats. Les chapitres 3 à 5 présentent en détail les contributions de la thèse. Le dernier chapitre 6, conclut avec des propositions de recherches futures.

Au chapitre 3 nous considérons deux types de jeux pour le problème décentralisé de minimisation du coût social. Dans le premier jeu, les joueurs contrôlent le temps de complétude de leurs tâches par l'annonce d'une date limite d_i que la machine doit respecter. Par contre, dans le deuxième jeu, les joueurs ne décident pas directement des temps de complétude de leurs tâches, mais annoncent un facteur de pénalité de retard p_i. Cette pénalité indique l'importance que le joueur apporte au fait que sa tâche termine tôt. Intuitivement si cette quantité est grande, le joueur est prêt à payer beaucoup pour que cette tâche termine tôt. Pour les deux jeux, nous étudions l'existence d'équilibres de Nash purs, le temps de convergence vers ces équilibres, et le rapport entre l'énergie consommée et le montant des factures.

Le chapitre 4 est consacré au problème centralisé de minimisation du coût social. Concrètement, nous montrons qu'il est equivalent à un problème d'ordonnancement noté $1||\sum w_i f(C_i)$ [26] avec $f(t) = t^{(\alpha-1)/\alpha}$ étant une fonction croissante concave particulière. La complexité de ce problème est ouverte, et nous établissons alors un état de l'art sur le problème d'ordonnancement pour une large classe de fonctions de pénalité f, en particulier nous nous intéressons aux fonctions concaves et étudions une technique particulière de preuve de NP-complétude largement utilisée pour des fonctions convexes.

Le chapitre 5 concerne la fonction $f(t) = \text{sign}(\beta) \cdot t^{\beta}$ pour une constante $\beta \in \mathbb{R}$ et où $\text{sign}(\beta) \in \{-1, 0 + 1\}$ est le signe de β. La fonction objective $\sum w_i f(C_i)$ avec cette forme de f a attiré l'attention de la communauté en particulier pour $\beta = 2$. Spécifiquement, nous étudions des propriétés de dominance qui trouvent leur usage dans des résolutions exactes et exhaustives.

Nous présentons dans le chapitre 6 notre bilan et les questions que nous envisageons d'aborder dans nos travaux futurs.

1.4 Les résultats en bref

1.4.1 Optimisation du coût social : deux approches décentralisées

Nous concevons deux jeux définis par la stratégie du joueur participant. Ici, se pose le problème de la conception d'un mécanisme de facturation qui satisfait les propriétés suivantes :

- existence d'un équilibre de Nash
- surfacturation bornée
- véracité garantie

1.4.1.1 Jeu de date limite

Nous étudions un jeu, où les joueurs contrôlent le temps de complétude de leur tâche. Pour illustrer cette possibilité, nous nous référons à la situation où on donne un travail d'impression

de photos dans un magasin, et typiquement le client a le choix d'un temps de complétude de deux heures ou de deux jours avec des tarifs adaptés.

Concrètement le joueur i possède une tâche i, avec une quantité de travail w_i, une date de relâchement r_i et un facteur de pénalité de retard p_i. Avec la soumission de sa tâche, il annonce une date limite d_i que le régulateur doit respecter. Cette dernière valeur représente la stratégie que le joueur i est libre de choisir, tandis que w_i, r_i sont des valeurs données, publiques et hors de contrôle du joueur. Le facteur de pénalité de retard p_i est une valeur privée.

Le régulateur de jeu calcule alors un ordonnancement qui respecte toutes les fenêtres d'exécution et qui minimise l'énergie consommée. Ce problème peut être résolu en temps polynomial, comme expliqué dans la section 1.1.1. Ensuite cette énergie est facturée aux joueurs. Le but des joueurs est de minimiser le montant de leur facture d'énergie plus la pénalité de retard $p_i(d_i - r_i)$.

Notons ici que nous aurions pu définir la pénalité de retard comme $p_i(C_i - r_i)$, sans que cela change essentiellement la nature du jeu. En effet si pour un joueur $C_i < d_i$, alors diminuer la date limite annoncée par i préserve son coût individuel. Pour des raisons de simplicité nous avons choisi $p_i(d_i - r_i)$ comme pénalité de retard. Aussi comme $p_i r_i$ est une constante, nous allons ignorer par la suite cette quantité. En résumé la pénalité du joueur i est

$$p_i d_i + b_i,$$

où b_i est le montant de sa facture.

Le premier mécanisme de facturation qui vient à l'esprit est de facturer précisément l'énergie consommée pendant l'exécution de la tâche. Nous l'appelons *mécanisme de facturation proportionnel*. Ce mécanisme semble équitable, car chaque joueur paye réellement ce qui est consommé par sa tâche. De plus il n'y a pas de surfacturation, le montant total des factures équivaut à l'énergie consommée. Dans ce sens, ce mécanisme semble parfait pour le deuxième critère.

Malheureusement il ne garantit pas l'existence d'un équilibre de Nash. La preuve de cette proposition consiste en un exemple très simple de deux tâches identiques. Notre contribution est une étude de cas des meilleures réponses, qui montrent qu'il n'existe pas de point fixe dans ce jeu. L'idée est que puisqu'ils doivent bien être exécutés dans un certain ordre, l'ordonnancement introduit une asymétrie. Ainsi il existe toujours un joueur qui peut améliorer son coût en changeant de strategie.

Théorème 1.1. *Pour le mécanisme de facturation proportionnelle, l'existence des équilibres de Nash n'est pas garantie.*

Nous proposons un autre mécanisme de facturation où chaque joueur paye la différence entre les ordonnancements optimaux pour tous les joueurs et pour tous les joueurs sauf lui. Intuitivement, nous pouvons imaginer la situation où le joueur i rejoint le jeu. Alors le coût de l'ordonnancement optimal augmente par sa présence, et c'est cette augmentation qui est facturée au joueur i.

Par construction, le jeu devient alors un jeu de potentiel exact. Ceci veut dire qu'il existe une fonction de potentiel qui dépend des stratégies des joueurs, et que si un joueur change de stratégie et modifie son coût d'une valeur Δ alors le potentiel change du même montant Δ. Il s'agit d'une famille de jeu très étudiée, qui admet de nombreuses propriétés.

Pour des jeux stratégiques finis, donc où l'espace des stratégies est fini, les jeux de potentiels exacts admettent toujours un équilibre de Nash pur [47]. Dans notre jeu, nous rencontrons

la difficulté que les dates limites annoncées par les joueurs sont des réels positifs, et donc un simple argument de recherche locale ne permet pas d'établir avec certitude l'existence d'un équilibre de Nash. Il se pourrait qu'une telle recherche locale soit un processus infini, et c'est en effet le cas avec notre jeu.

Cependant l'utilisation du fait que l'espace des stratégies est continu et compact, nous permet de conclure à l'existence d'une valeur qui minimise la fonction potentielle, et alors nous avons un équilibre de Nash pur.

Théorème 1.2. *Pour le mécanisme de facturation marginal, le jeu est un jeu de potentiel exact, qui admet toujours un équilibre de Nash pur. La convergence vers un équilibre peut générer un nombre de changements infini.*

Ce coût est en fait aussi le potentiel du jeu. Donc l'équilibre de Nash qui minimise le potentiel est un optimum social. En utilisant des techniques de majoration, nous avons pu borner la surfacturation.

Théorème 1.3. *Pour le mécanisme de facturation marginal, le jeu est à surfacturation α. Ce facteur est atteint par une famille d'instances de ce jeu.*

En général, ce mécanisme n'a pas de garantie de facteur constant pour l'efficacité, tandis que la propriété de véracité garantie ne s'applique pas.

1.4.1.2 Jeu de pénalité

Nous considérons un jeu différent de la section précédente, où le joueur ne décide pas directement du temps de complétude de sa tâche, mais annonce un facteur de pénalité de retard p_i. Cette pénalité indique l'importance que le joueur apporte au fait que sa tâche termine tôt. Intuitivement si cette quantité est grande, le joueur est prêt à payer beaucoup pour que cette tâche se termine tôt.

Dans ce jeu chaque joueur i soumet sa tâche, et annonce w_i et une pénalité de retard $\hat{p}_i$, qui pourrait être différente de p_i si c'est à l'avantage du joueur. Par contre nous pouvons supposer que le joueur ne peut pas mentir sur la quantité de travail w_i, car elle est vérifiable soit au moment de la soumission, soit au moment de la complétude de la tâche, et le joueur pourrait être puni dans le cas d'un bias, pour le forcer à être honnête sur cette valeur.

Dans ce contexte les tâches n'ont pas de date de relâchement et sont donc disponibles à partir du temps 0. Aussi les tâches ne sont pas munies d'une date limite, et c'est au régulateur de décider librement d'un ordonnancement qui tient à la fois compte du temps de complétude pondéré des tâches et de l'énergie consommée.

Comme toujours, le coût social est défini comme le temps de complétude pondéré totalisé sur toutes les tâches plus l'énergie consommée. Minimiser le coût social est un problème d'ordonnancement, auquel est dédiée la section 1.4.2. Nous mentionnons à cet endroit seulement, que seul un schéma d'approximation polynomial (PTAS) est connu pour ce problème, bien qu'il comporte une structure intéressante, que nous décrivons maintenant.

Puisque les tâches sont toutes disponibles à partir du temps 0, l'ordonnancement optimal exécute toutes les tâches sans temps mort, ni interruption jusqu'à la complétude de la dernière tâche. Il existe donc une permutation π qui décrit l'ordre dans lequel les tâches sont exécutées, où $\pi(i)$ est le rang de la tâche i dans l'ordonnancement. De plus comme toute tâche est exécutée à une vitesse constante, il suffit de préciser dans un vecteur ℓ les durées d'exécution des tâches. Le couple π, ℓ décrit alors entièrement l'ordonnancement.

Deux coûts sont associés à cet ordonnancement. Il y a d'abord la consommation d'énergie

$$E(\ell, w) := \sum_i w_i^\alpha \ell_i^{1-\alpha},$$

qui a cette forme car chaque tâche i est exécutée pendant la durée ℓ_i avec la vitesse w_i/ℓ_i. Le deuxième coût est le temps d'attente total des tâches, appelé *weighted flow time* en anglais, et qui est

$$F(\pi, \ell, p) := \sum_j p_j C_j$$

où C_j est le temps de complétude de la tâche et qui lui, à son tour, est défini comme la somme de ℓ_i sur toutes les tâches i de rang $\pi(i) \leq \pi(j)$.

La difficulté que nous rencontrons dans ce jeu est qu'il nous semble impossible d'avoir les trois propriétés à la fois : la garantie de la véracité, l'efficacité et l'équilibre de budget. Nous proposons alors un mécanisme de facturation qui relâche un facteur constant l'équilibre de budget pour garantir la véracité des pénalités annoncées par les joueurs.

Ce mécanisme n'a malheureusement pas de garantie de facteur constant pour l'efficacité, et nous n'avons pas trouvé une manière de remédier à ce problème. La principale raison est que nous devons fixer un ordre sur les tâches, qui est indépendant des stratégies des joueurs, afin d'empêcher leur influence. Cette propriété est importante pour la véracité garantie.

Notre mécanisme de facturation utilise un ordre d'exécution fixé π sur les tâches. Cet ordre peut être arbitraire, ou aléatoire, c'est sans importance pour la suite. Soit ℓ le vecteur de longueurs d'exécution qui minimise le coût social $E(\ell, w) + F(\pi, \ell, \hat{p})$. Ce vecteur peut être calculé en temps polynomial, car π est fixé. Soit $E(OPT_\pi(\hat{p}))$ la consommation d'énergie de cet ordonnancement, donc $E(\ell, w)$. De plus nous notons $E(OPT_\pi(\hat{p}_{-i}))$ l'énergie de l'ordonnancement optimal pour tous les joueurs sauf i. La différence représente l'augmentation dans l'énergie consommée par la présence de i dans le jeu. La valeur qui est facturée au joueur i est alors définie comme

$$b_i := \alpha \left(E(OPT_\pi(\hat{p})) - E(OPT_\pi(\hat{p}_{-i}))\right) - \hat{p}_i C_i,$$

où C_i est le temps de complétude de la tâche i dans l'ordonnancement défini par π, ℓ. L'idée derrière cette facture est que la pénalité du joueur i devient

$$\alpha \left(E(OPT_\pi(\hat{p})) - E(OPT_\pi(\hat{p}_{-i}))\right) + (p_i - \hat{p}_i) C_i,$$

qui pour la partie dépendante de $\hat{p}_i$ est très proche du coût social. Lier la pénalité individuelle au coût social est la clé pour un mécanisme efficace, car les joueurs minimisent dans la même direction que le coût social. Si un joueur veut minimiser $(p_i - \hat{p}_i)C_i$, alors clairement il va annoncer la véritable valeur $\hat{p}_i = p_i$. Notre contribution est de montrer que s'il veut minimiser sa pénalité, alors la meilleure réponse est aussi la véritable valeur p_i. La preuve passe par la première et deuxième dérivée de la pénalité en $\hat{p}_i$.

Théorème 1.4. *Le jeu où l'ordre des joueurs π est fixé, est à véracité garantie. Il surfacture avec un facteur $(\alpha + 1)$ au plus.*

En ce qui concerne le coût social, par construction, le mécanisme calcule l'ordonnancement qui minimise le coût social $E(\ell, w) + F(\pi, \ell, \hat{p})$, et est donc optimal dans ce sens pour un ordre π fixé. Par contre ce coût peut être arbitrairement grand par rapport à l'optimum social lorsque l'ordre n'est pas fixé.

1.4.2 Optimisation du coût social : une approche centralisée

1.4.2.1 Réduction vers un problème d'ordonnancement

Nous considérons le problème centralisé de la minimisation du coût social, qui consiste à minimiser le temps de complétude pondéré total des tâches plus la consommation d'énergie.

- Dans la section 1.1.1 nous avons mentionné que si les dates limites des tâches sont fixées, le problème de la minimisation d'énergie peut être résolu en temps polynomial [43].
- Si par contre la consommation d'énergie est fixée, le problème de la minimisation du temps d'attente total pondéré est un problème ouvert. Seul le cas des poids unitaires ($w_i = 1$) a été montré polynomial [50].
- Quand aucune des quantités n'est fixée, le problème de la minimisation d'énergie plus temps d'attente pondéré est ouvert.

Nous définissons formellement le problème comme suit.

Problème A Étant donné n tâches aux quantités de travail w et facteurs de pénalité de retard p, trouver un ordonnancement (π, ℓ) qui minimise $A_{w,p}(\pi, \ell) := E(\ell, w) + F(\pi, \ell, p)$.

Ce problème est équivalent à un autre problème d'ordonnancement B. Pour celui-ci, par contre, la machine travaille de manière constante avec la vitesse 1, et le concept de la vitesse est codée dans la fonction objective. Dans les deux problèmes les deux quantités de l'instance w, p jouent des rôles opposés. Ce qui est une quantité de travail pour le problème A devient une priorité pour le problème B, et le facteur de pénalité de retard p du problème A devient une durée d'exécution pour le problème B.

Problème B Étant donné n tâches aux poids de priorité w et durées d'exécution p, trouver un ordonnancement σ qui minimise $B_{w,p}(\sigma) := \sum w_j C_j^{(\alpha-1)/\alpha}$, où la date de complétude C_j est définie comme la somme $\sum p_i$ sur toutes les tâches i de rang $\sigma(i) \leq \sigma(j)$.

Nous montrons l'équivalence entre les problèmes A et B. Le même résultat a été obtenu indépendamment par Megow et Verschae, en utilisant des conditions Karush-Kuhn-Tucker [44]. Notre preuve par contre est basée sur les conditions de Hessian. L'énoncé précis de cette équivalence est le suivant.

Théorème 1.5. *Soient π, σ deux permutation sur $1, \ldots, n$ tel que π est l'ordre inverse de σ, c'est-à-dire $\pi(i) = n + 1 - \sigma(i)$, soit ℓ le vecteur de longueurs d'exécution qui minimise $A_{w,p}(\pi, \ell)$, alors*

$$A_{w,p}(\pi, \ell) = \alpha(\alpha - 1)^{(\alpha-1)/\alpha} B_{w,p}(\sigma).$$

Ce qui est intéressant dans ce résultat est qu'il lie un problème bi-critère à un problème d'ordonnancement, qui à la fois est mono-critère, mais comporte aussi une structure proche des problèmes d'ordonnancement étudiés dans la littérature. On a espéré ainsi pouvoir déterminer la complexité du problème B. Malheureusement le problème reste ouvert, on ne connait ni d'algorithme polynomial ni de preuve de NP-complétude.

L'absence d'algorithme polynomial peut s'expliquer par le fait qu'il manque des structures de dominances qui habituellement permettent une résolution par programmation dynamique,

par programmation linéaire ou par un algorithme glouton. D'un autre côté, la nature continue de la fonction objective, rend difficile le codage d'un problème NP-dur, qui nécessite des décisions binaires.

Le seul résultat connu à ce jour pour le problème B, est un schéma d'approximation polynomial de Megow et Verschae [44]. Par contre le problème B entre dans une classe de problèmes d'ordonnancement qui ont été étudiés dans le passé.

1.4.2.2 Sur la compléxite du problème

Motivé par l'étude du problème d'ordonnancement B, nous établissons un état de l'art et étudions la complexité sur ces fonctions de la forme suivante : l'entrée du problème consiste en n tâches, chacune avec une durée d'exécution p_j et une priorité w_j. Un ordonnancement est simplement un ordre d'exécution de ces tâches, tel que la date de complétude C_j de la tâche j est définie comme la somme $\sum p_i$ sur toutes les tâches i dont le rang est au plus celui de j (incluant j). Le but est de minimiser l'objectif $\sum w_j f(C_j)$ pour une fonction de pénalité f donnée. Ce problème peut être noté par $1||\sum w_j f(C_j)$ selon la notation à trois champs proposés en [26] et sa complexité dépend évidement de la forme de la fonction f.

Cette forme générale est motivée par différentes applications concrètes. D'une part nous avons vu que pour $f(t) = t^{(\alpha-1)/\alpha}$, ce problème correspond au problème de minimisation de l'énergie et du temps d'attente pondéré total. D'autre part cette forme de fonction objective modélise le cas d'une machine dont la vitesse de travail évolue au cours du temps. Cette évolution est codée dans la fonction f de la manière suivante. Si s est la fonction qui indique la vitesse de travail au cours du temps, alors f est l'inverse de l'intégrale de s. Minimiser $\sum w_j f(C_j)$ revient alors à minimiser le temps d'attente moyen $\sum w_j C_j/n$ pour une machine à vitesse variable s, mais dont la variation de vitesse serait fixée d'avance. Ce changement de vitesse peut être dû à un effet d'apprentissage, d'un effet d'usure et de fatigue, d'une mise à jour logiciels, remplacement de matériel, etc.

Nous attirons l'attention sur le fait que la fonction f est indépendante des tâches. La classe des problèmes d'ordonnancement à l'objectif $\sum w_j f_j(C_j)$ est beaucoup trop large pour permettre d'établir un simple état de l'art.

De plus nous nous intéressons à des fonctions de pénalité qui peuvent (mais pas forcément) satisfaire les propriétés suivantes.

- La monotonicité de f, est une propriété naturelle, car elle traduit le fait qu'il est toujours préférable de finir une tâche tôt que tard. Seulement dans des problèmes d'ordonnancement juste à temps, cette propriété n'est pas souhaitable et on veut que les tâches se terminent le plus près d'une date centrale d donnée.

- La convexité ou concavité de f traduit une pénalité de retard qui n'est pas linéaire. Une augmentation de retard d'une valeur δ pourrait être ressentie différemment suivant que la tâche se termine déjà tard ou tôt. Dans un contexte de changement de vitesse de la machine la convexité traduit un ralentissement constant avec le temps, tandis que la concavité traduit une accélération.

- La linéarité par morceaux de f, modélise des changements de vitesse de la machine suite à des interventions ponctuelles. Dans un contexte de production, les pénalités de retard de livraison, changent quand une date critique donneé est dépassée.

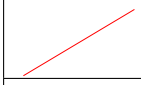
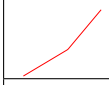
plot	fonction	complexité	plot	fonction	complexité
	t	$O(n \log n)$ [54]		e^{rt}	$O(n \log n)$ [51]
	$\max\{0, t-d\}$	NP-dur au sens faible [64] DP : $O(n^2 d)$ [42] FPTAS : $O((n^6 \log \sum_j w_j)/\varepsilon^3)$ [39], $O(n^2/\varepsilon)$ [37]		$1 - e^{-rt}$	$O(n \log n)$ [51]
	$t + \max\{0, t-d\}$	DP :$O(n^2 d^2)$ [60] FPTAS : $O(n^3 \log d/\varepsilon^2)$ [60]		t^2	Ouvert
	$\max\{0, d-t\}$	NP-dur au sens faible [14, 18] DP : $O(n^2 \sum_j p_j)$ [14] FPTAS :$O(n^4/\varepsilon)$ [14]		$\sqrt{t}$	Ouvert
	$\min\{t, d\}$	DP : $O(nd)$ FPTAS :$O(n^2/\epsilon)$		$\lvert d-t \rvert$	NP-dur au sens faible [27, 33] DP : $O(n^2 d)$ [33], $O(n \sum_j p_j)$ [27] FPTAS : $O(n^6/\varepsilon^3)$ [40]
	piecewise linear	NP-dur au sens fort [32] PTAS : $2^{O(\log(1/\varepsilon)/\varepsilon^2)} \log(\sum_j w_j) n$ [44]		$(d-t)^2$	des résultats existent seulement pour le cas non pondéré DP : $O(n \sum_j p_j)$ [61], $O(n^2(d + \max p_j))$ [46], $O(nd)$ [55]

TABLEAU 1.1 – Résultats de complexité pour différentes fonctions de pénalité f. Des algorithmes basés sur la programmation dynamique sont marqués par DP.

Le tableau 1.1 résume des résultats de complexité connus pour différentes fonctions de pénalité f données. Nous écrivons f en fonction du temps t, et d, r sont des paramètres constants.

Nous observons que beaucoup de preuves de NP-complétude en ordonnancement sont basées sur une réduction du problème de *Partition* [25]. Nous illustrons la technique avec le problème d'ordonnancement avec la fonction de pénalité $f(t) = \max\{0, t-d\}$ (voir [64]).

Une instance du problème de PARTITION consiste en des entiers $a_1, \ldots, a_n$, et le problème consiste à décider s'il est possible de partitionner $\{1, \ldots, n\}$ en deux ensembles $S, \bar{S}$ tel que

$$\sum_{i \in S} a_i = \sum_{i \in \bar{S}} a_i. \tag{1.1}$$

Pour coder une instance de *Partition* en une instance du problème d'ordonnancement, nous allons donc utiliser le fait qu'une tâche a une contribution à la fonction objective seulement si elle ne se termine pas apres d. Donc la date limite d partitionne naturellement les tâches en deux ensembles. Comme dans l'équation (1.1) il n'y a pas de notion d'ordre des tâches, nous aimerions qu'il en soit de même dans le problème d'ordonnancement. Pour ce faire, chaque élément a_i génère une tâche i avec la durée $p_i := a_i$ et le poids $w_i := a_i$ également. Par le résultat de Smith [54], la valeur objective est indépendante de l'ordre des tâches terminant après d et bien sûr aussi de l'ordre des tâches terminant avant d. La réduction est complétée

par une $n+1$ ème tâche avec des caractéristiques qui la forcent à se terminer au temps d dans un ordonnancement optimal. Cette tâche aura une longueur suffisamment petite et une densité w_{n+1}/p_{n+1} plus grande que 1. Pour rendre la preuve complète, toutes les quantités sont multipliées par un facteur pour les rendre entières.

La réduction précédente montre que le problème d'ordonnancement pour la fonction de pénalité $f(t) = \max\{0, t-d\}$ est NP-dur déjà pour des instances de densité pratiquement uniforme qui sont des instances dont toutes les tâches i sauf éventuellement une, ont la même densité w_i/p_i.

Ce que nous avons réussi à montrer est que cette classe d'instances est facile pour toute fonction de pénalité croissante et concave.

Théorème 1.6. *Soit f une fonction de pénalité concave et croissante. Le problème d'ordonnancement qui consiste à minimiser $\sum w_j f(C_j)$ peut être résolu en temps $O(n \log n)$ pour des instances de densité pratiquement uniforme.*

Bien sûr que cela n'exclut pas le fait que des problèmes avec ce type de fonctions pourraient être NP-durs, mais la communauté d'ordonnancement manque de techniques de preuve de NP-complétude d'une nature profondément différente que celle décrite dans la section précédente.

Étant donné le dernier résultat, nous étudions les méthodes de solution au problème. En particulier, nous nous sommes consacrés aux fonctions de pénalité croissante et concave $f(t) = \min\{t, d\}$. Elles correspondent au problème d'ordonnancement avec minimisation de temps total pondéré des temps de complétude pour une machine qui fonctionne avec une vitesse 1 jusqu'à un temps d où elle travaille soudainement avec une vitesse infinie.

Pour ce problème la communauté d'ordonnancement ne connaît pas d'algorithme en temps polynomial. Par contre, nous avons proposé une réduction à un problème de *minimisation de demi-produits* qui permet d'obtenir un schéma d'approximation fortement polynomial (FPTAS).

Définition 1.1. *Un demi-produit est une fonction de la forme*

$$F(x) := \sum_{j=1}^{n} a_j x_j \sum_{i=1}^{j-1} b_i x_i - \sum_{j=1}^{n} c_j x_j + D$$

où $x \in \{0,1\}^n$ et $a = (a_1, \ldots, a_n), b = (b_1, \ldots, b_{n-1})$ entiers non-négatifs et $c = (c_1, \ldots, c_n), D$ entiers arbitraires.

Le nom de cette fonction vient du fait qu'il ne contient qu'une partie des éléments du produit complet $(a \cdot x)(b \cdot x) + c \cdot x + D$, où $\cdot$ est le produit interne. De nombreux problèmes se modélisent comme un problème de minimisation de demi-produits [5]. Il en est de même pour le problème d'ordonnancement considéré.

Théorème 1.7. *Le problème d'ordonnancement pour la fonction de pénalité $f(t) = \min\{t, d\}$ peut être modélisé comme un problème de minimisation de demi-produits pour les valeurs*

$$\begin{aligned} a_j &= w_j \\ b_j &= p_j \\ c_j &= (d - p_j) w_j \\ D &= d \sum w_j. \end{aligned}$$

Nous héritons alors d'un schéma d'approximation pour notre problème d'ordonnancement par le théorème suivant dû à Erel et Ghosh [23].

Théorème 1.8. *Il existe un schéma d'approximation pour minimiser le demi-produits F en temps $O(n^2/\varepsilon)$, avec la hypothèse que* $\min F > 0$.

Une conséquence de ce théorème est bien sûr que le problème d'ordonnancement ne peut pas être NP-dur au sens fort, sans exclure toute fois la NP-complétude au sens faible.

Finalement, nous présentons une preuve dans un cadre plus global, où les tâches ont différentes fonctions de pénalité.

Théorème 1.9. *Le problème d'ordonnancement pour la fonction de pénalité* $1||\sum w_j \min\{C_j, d_j\}$ *est NP-difficile.*

1.4.3 Propriétés de dominance d'ordre

Dans cette section nous abordons la résolution du problème d'ordonnancement B par une méthode de résolution exacte, en particulier, l'algorithme A*, et montrons des propriétés de dominance d'ordre sur les tâches. La section se termine par une évaluation expérimentale de nos nouvelles règles.

Nous fixons une fonction de pénalité générale sous la forme $f(t) = sign(\beta) \cdot t^\beta$ pour une constante arbitraire $\beta \in \mathbb{R}$ où $sign(\beta) \in \{-1, 0, +1\}$ est le signe de β. Le problème d'ordonnancement associé consiste à trouver un ordre sur les tâches qui minimise $\sum w_j f(C_j)$. En absence d'algorithme polynomial, des résolutions exhaustives ont été proposées. En particulier, il existe une réduction vers un problème de plus court chemin que nous décrivons maintenant.

La réduction utilise le tors suivant. Il s'agit du graphe orienté acyclique (DAG en anglais), dont les sommets sont des sous ensembles de tâches $\{1, \ldots, n\}$, et il existe un arc de S vers S' si $S' = S \cup \{j\}$ pour un sommet $j \notin S$. Un tel arc est pondéré avec $w_j f\left(\sum_{i \in S'} p_i\right)$. Il est clair qu'un chemin de $\{\}$ à $\{1, \ldots, n\}$ correspond à un ordonnancement dont le coût est la somme du coût des arcs. Ainsi le problème d'ordonnancement est réduit à un problème de plus court chemin dans un graphe de 2^n sommets, ce qui représente déjà une légère amélioration par rapport à une recherche exhaustive parmi les $n!$ ordres possibles.

Chercher un plus court chemin dans ce type de graphe peut être résolu avec l'algorithme A*, qui est une variante de l'algorithme de Dijkstra. Ce dernier fonctionne avec une pile de priorité Q contenant les arcs (S, S') vers des sommets S' à explorer, avec une priorité qui est définie comme la distance de $\{\}$ vers S plus le coût de l'arc. L'amélioration apportée par A* est d'ajouter à cette priorité une borne inférieure de la distance de S' vers $\{1, \ldots, n\}$. En pratique, l'amélioration est significative et permet de résoudre de très grandes instances pour le problème considéré.

Pour notre implémentation nous avons choisi la borne inférieure la plus élémentaire, à savoir $\sum_{j \notin S'} w_j f\left(p_j + \sum_{i \in S'} p_i\right)$. Ce choix répond à un souci d'équilibre entre qualité de la borne inférieure et le temps de calcul nécessaire de la borne.

Cette approche a été introduite en 1972 par [28], puis appliquée à cette famille de problèmes d'ordonnancement par [52] et finalement améliorée pour la fonction de pénalité quadratique $f(t) = t^2$, par Höhn et Jacobs [30].

Nous considérons deux variantes de l'algorithme précédent. Dans l'approche *gauche-droite* ou *foward* en anglais, un ordonnancement partiel décrit un préfixe d'une certaine longueur t

d'un ordonnancement complet et se prolonge à sa droite pour l'arbre de recherche, tandis que l'approche *droite-gauche* ou *backward* en anglais, un ordonnancement partiel décrit un suffixe d'un ordonnancement complet et se prolonge à sa gauche.

L'algorithme A* peut être amélioré en incorporant des règles de dominance. Celles-ci sont de deux types, des règles de précédence locale et des règles de précédence globale.

Considérons un ordonnancement où la tâche j suit immédiatement la tâche i, et que i commence au temps t. L'échange de i avec j a une conséquence sur la fonction objective qui ne dépend que des tâches i et j et du temps t. Si l'ordre i, j est préférable nous notons cette propriété par $i \prec_{\ell(t)} j$. Cette propriété permet de supprimer un arc (S, S') du graphe dans le cas où (1) $S' = S \cup \{j\}$, (2) l'ordonnancement optimal de S se termine avec i et (3) que $i \prec_{\ell(t)} j$ pour $t = \sum_{k \in S} p_k$. Cette suppression est valide, car tout chemin de $\{\}$ à $\{1, \ldots, n\}$ qui passe par cet arc serait sous optimal.

Si la priorité $i \prec_{\ell(t)} j$ est vraie pour tout $t \geq 0$ alors on dit que i, j suivent un ordre de précédence local, et on note $i \prec_{\ell} j$. Cet ordre peut être renforcé comme suit.

Définition 1.2. *Si tout ordonnancement S où j est exécuté avant i — sans être nécessairement consécutif — est sous optimal, alors on dit que les tâches i, j suivent un ordre de précédence global noté $i \prec_g j$.*

Cet ordre peut être généralisé à des intervalles $[a, b]$, où $i \prec_{g[a,b]} j$ dit essentiellement que si j, i s'exécutent en $[a, b]$ et dans l'ordre j, i alors l'ordonnancement est sous optimal. La définition précise est omise. Mais ce qui est important est que cet ordre peut aussi être utilisé pour couper des arcs dans le graphe de recherche et ainsi améliorer la performance de A*.

En effet, pour les mêmes que raisons celles citées ci-dessus, on peut supprimer un arc (S, S') du graphe si $S' = S \cup \{j\}$, et qu'il existe une tâche $i \notin S'$ avec $i \prec_{g[a,b]} j$ pour a la durée totale de S et b la durée totale sur toutes les tâches de l'instance.

Des règles impliquant des propriétés d'ordre ont été proposées dans la littérature (voir [3,7,17,53,56,57]) et concernent principalement la fonction de pénalité quadratique, i.e. $\beta = 2$. En 2000, Mondal et Sen [45] ont conjecturé pour $\beta = 2$ que si $(w_i \geq w_j) \wedge (w_i/p_i > w_j/p_j)$, alors $i \prec_g j$. Récemment, Höhn et Jacobs [30] ont montré cette conjecture et donné des règles de précédence locale et globale pour tout constante entière $\beta \geq 2$. Leur travail est complété par une étude expérimentale pour évaluer l'impact sur la performance de ces nouvelles règles en utilisant l'Algorithme A*.

Nous contribuons par de nouvelles règles, illustrées en Figure 1.2.

Théorème 1.10 (nos règles)**.** *Soit i, j deux tâches avec $p_i > p_j$.*

Si $\beta > 1$

Si $(p_i/p_j)^\beta \leq w_i/w_j$	*alors* $i \prec_g j$	
Si $p_i/p_j < w_i/w_j$	*alors* $i \prec_\ell j$	*[ou même* $i \prec_g j$ *si* $\beta = 2$*]*
Si $w_i/w_j < \phi_{ij}(0)$	*alors* $j \prec_g i$	
sinon $\exists t^* : w_i/w_j = \phi_{ij}(t^*)$ *et*	$i \prec_{\ell[0,t^*)} j$ *et* $j \prec_{g[t^*,\infty)} i$	

Si $0 < \beta < 1$

Si $(p_i/p_j)^2 \leq w_i/w_j$	*alors* $i \prec_g j$
Si $\phi_{ij}(0) < w_i/w_j$	*alors* $i \prec_\ell j$
Si $w_i/w_j < p_i/p_j$	*alors* $j \prec_{g[0,t_1]} i$

sinon $\exists t^* : w_i/w_j = \phi_{ij}(t^*)$ *et*	$i \prec_{g[0,t^*)} j$
	et $j \prec_{\ell[t^*,\infty)} i$.
Si $\beta = -1$	
Si $(p_i/p_j)^2 \leq w_i/w_j$	*alors* $i \prec_g j$
Si $(w_i/w_j) < p_i/p_j$	*alors* $i \prec_g j$
sinon $\exists t^* = \frac{w_j p_i^2 - w_i p_j^2}{w_i p_j - w_j p_i} \geq 0$ *et*	$i \prec_{g[0,t^*)} j$
	et $i \prec_{g[0,t^*)} j$

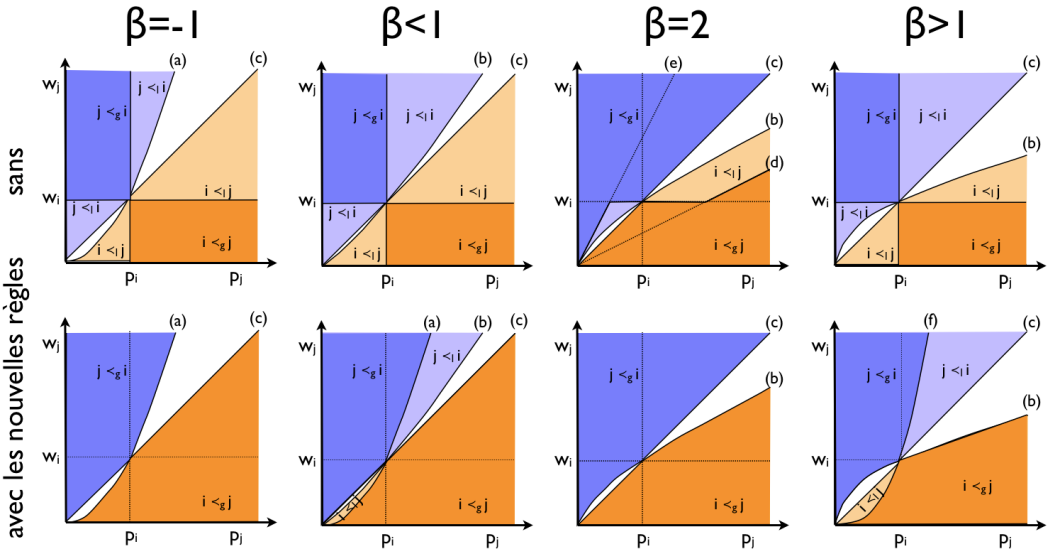


FIGURE 1.2 – Tâche j par rapport à une tâche fixée i. Les fonctions montrées sont : (a) $w_j = w_i(p_j/p_i)^2$, (b) $w_j = w_i((p_i+p_j)^\beta - p_i^\beta)/((p_i+p_j)^\beta - p_j^\beta)$, (c)$w_j = w_i p_j/p_i$, (d) $w_j = w_i p_j/2p_i$, (e) $w_j = 2w_i p_j/p_i$ et (f) $w_j = w_i(p_j/p_i)^\beta$.

Dans ce contexte nous énonçons la conjecture suivante motivée par des résultats expérimentaux.

Conjecture 1.1. *Si la fonction de pénalité est de la forme* $f(t) = t^\beta$ *pour* $\beta > 0$ *arbitraire, alors pour toutes tâches* i, j, $i \prec_\ell j$ *implique* $i \prec_g j$.

La conjecture est triviale pour $\beta = 1$ et nous avons pu la montrer pour $\beta = 2$. Pour $\beta = -1$, le problème consiste à maximiser $\sum w_j/C_j$ et une variante plus forte de la conjecture est aussi vraie, quand les ordres de précédence sont restreints à des intervales $[a, b]$. Pour $\beta = 2$ nous avons par contre un contre-exemple à cette variante de la conjecture.

Pour l'évaluation de l'impact des règles proposées, nous considérons l'algorithme A* et un modèle de génération d'instances aléatoires inspirée du modèle proposé par Höhn et Jacobs [30].

Formellement, les instances de nos principaux ensembles de tests sont générées comme suit. Pour chaque choix de β et σ, nous générons 25 instances de 20 tâches chacun. La durée d'exécution p_i de chaque tâche i est générée de manière uniforme dans $1, 2, \ldots, 100$. Ensuite, le poids est généré selon β. Pour $\beta > 1$, la condition pour $i \prec_g j$ de nos règles peut être approchée à la relation $w_i/p_i \geq \beta w_j/p_j$ quand p_j/p_i tend vers l'infini. Par conséquent, afin d'obtenir

une difficulté similaire des instances aléatoires pour le même paramètre σ pour différentes valeurs de $\beta > 1$, nous choisissons le rapport w_i/p_i — appelé *Smith-ratio* — selon $2^{N(0,\beta^2\sigma^2)}$. De cette façon, le rapport entre les "Smith-ratios" de deux tâches est une variable aléatoire de distribution $2^{2N(0,\beta^2\sigma^2)}$, et la probabilité que cette valeur est au moins β ne dépend que de σ. Cependant, quand β est -1 ou β est compris entre 0 et 1, la condition pour $i \prec_g j$ de nos règles peut être aproximée par la relation $w_i/p_i \geq 2w_j/p_j$ quand p_j/p_i tend vers l'infini et par conséquent, nous choisissons le Smith-ratio des tâches selon β indépendant de la distribution de $2^{N(0,4\sigma^2)}$.

Afin de savoir quelle variante de l'algorithme A* est la plus efficace, nous avons mené une étude expérimentale. Les valeurs sont plus significatives pour les petites valeurs σ, puisque pour les grandes valeurs les instances sont faciles de toute façon et le choix de la variante n'est pas très important. Les résultats indiquent que, sans nos règles de la variante gauche-droite doit être utilisée chaque fois que $\beta = \{-1, 2\}$ ou $0 < \beta < 1$, tandis que la variante gauche-droite doit être utilisée lorsque $\beta > 1$. Pour comparer le comportement de l'algorithme nous utilisons la variante la plus favorable en fonction la valeur de β.

Dans l'exécution de l'algorithme A*, nous avons fixé une limite maximale d'un million de nœuds. A partir de ces expériences, nous mesurons les tailles d'instances qui peuvent être résolues de manière efficace et observons que cette limite est bien sûr plus petite quand σ est petit, dont les cas deviennent plus difficiles. En plus, nous constatons aussi qu'avec l'utilisation de nos règles de bien plus grandes instances peuvent être résolues.

Finalement, nous mesurons l'influence sur le nombre de nœuds générés quand nos règles sont utilisées. Pour la validité de la comparaison, nous avons exclu les cas où la limite n'est pas satisfaite. Les résultats montrent une amélioration significative quand l'algorithme est exécuté avec nos règles. La Figure 1.3 montre le ratio entre le nombre moyen de nœuds générés quand l'algorithme est exécuté avec nos règles, et quand il est exécuté sans nos règles pour certaines valeurs de β.

1.5 Contributions principales

En résumé les contributions principales de cette thèse sont les suivantes :

- Concevoir et étudier des jeux dans le contexte de la minimisation d'énergie et de la qualité de service pour un processeur capable de varier librement la vitesse de calcul où plusieurs agents non-coopératifs soumettent chacun une tâche à exécuter. En particulier, l'analyse des moyens de facturer l'énergie consommée aux utilisateurs pour les inciter à adopter un comportement qui concile la consommation d'énergie de la machine et les temps de calcul assumés ressentis par les utilisateurs. Formellement, nous avons montré l'existence d'équilibres de Nash purs, le temps de convergence vers ces équilibres, la garantie de véracité et le rapport entre l'énergie consommée et le montant des factures pour differents schémas (voir chapitre 3).

- La réduction du problème consistant à produire un ordonnancement minimisant une combinaison linéaire entre la consommation d'énergie et les temps de complétude des tâches pondérés vers un problème d'ordonnancement classique. Il s'agit de trouver un ordre sur les tâches données pour une exécution sur une machine sans variation de vitesse, qui minimiserait la somme sur toutes tâches j de $w_j f(C_j)$, où w_j est le poids de priorité de la tâche et C_j le temps de complétude, et $f(t) = t^{(\alpha-1)/\alpha}$ est une fonction

croissante concave particulière. Ce résultat lie un problème bi-critère à un problème d'ordonnancement, qui est mono-critère, et comporte une structure proche des problèmes d'ordonnancement étudiés dans la littérature (voir chapitre 4).

- L'étude de la complexité selon la fonction de pénalité f. En particulier nous démontrons des propriétés structurelles impliquant qu'une technique de preuve de NP-complétude largement utilisée dans le domaine échoue ici. En plus, nous étudions le problème avec une simple fonction concave $f(t) = \min\{t, d\}$, pour laquelle on ne connaît pas d'algorithme en temps polynomial. Nous proposons une réduction à un problème de *minimisation de demi-produits* ce qui permet d'obtenir un schéma d'approximation fortement polynomial (FPTAS) (voir chapitre 4).

- Pour des problèmes avec la fonction $f(t) = \mathit{sign}(\beta) \cdot t^{\beta}$ pour une constante $\beta \in \mathbb{R}$ et la fonction de signe $\mathit{sign}(\beta) \in \{-1, 0, +1\}$, nous énonçons une conjecture qui informellement peut s'énoncer ainsi. Si $\beta > 0$ et si pour deux tâches i, j, une exécution adjacente optimale privilègie l'ordre i, j indépendant de la position dans l'ordonnancement, alors l'ordre i, j est respecté par tout ordonnancement optimal, que ces tâches soient adjacentes ou non. Une preuve de cette conjecture aura un impact important sur des résolutions exactes, comme démontré par nos expériences, et nous avons été capables de la montrer pour $\beta = 2$, le cas $\beta = 1$ étant connu depuis soixante ans. Dans les autres cas, nous avons néanmoins montré des règles de dominance, certes plus faibles que la conjecture, mais dont la contribution a pu être évaluée favorablement par nos expériences. Pour $\beta = -1$, le problème consiste à maximiser $\sum w_j/C_j$ et nous avons montré une variante plus forte de la conjecture, qui est restreinte à des intervalles. Pour $\beta = 2$ cette variante de la conjecture est fausse (voir chapitre 3).

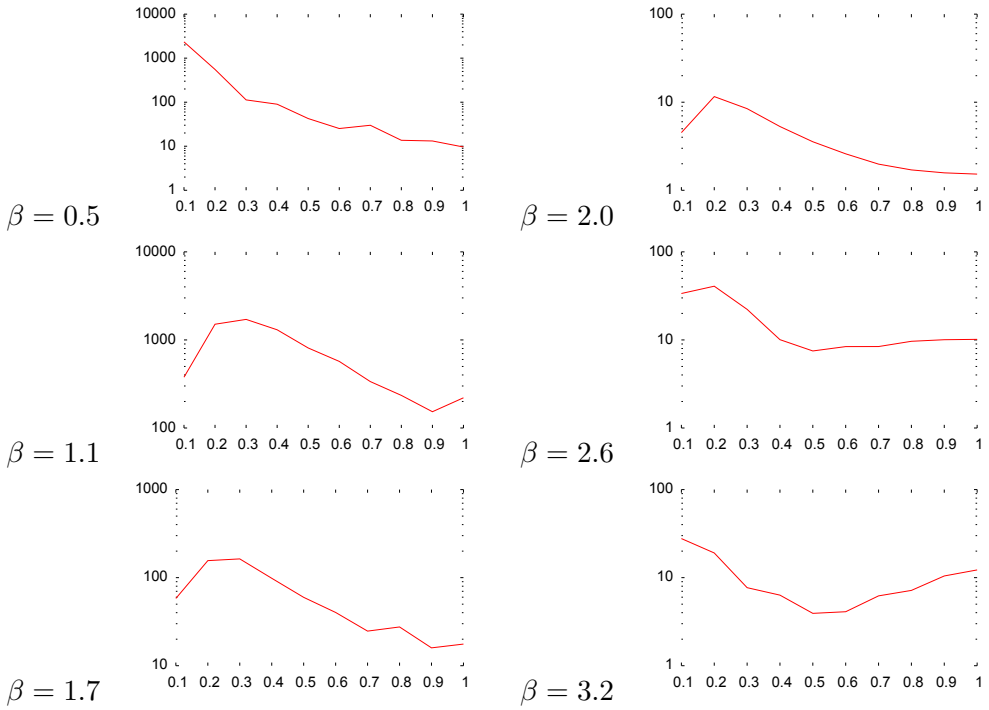


FIGURE 1.3 – Facteur d'amélioration du nombre de nœuds générés pendant la résolution avec et sans nos règles, en fonction de β et σ. Chaque point est une moyenne sur 25 instances à 20 tâches.

Chapter 2
Introduction (english)

2.1 The context

Advances in technologies placed processors in most of the machines used today, increasing in a certain manner the quality of life of developed countries. Tablet PCs, notebooks, iPads, iPods are good examples. However the usage of these technologies implies an increasing amount in computing power, placing the IT sector as an important energy consumer at world scale.

Many effort is made recently to limit the ecological impact of these new technologies. Minimizing energy consumption can be made at quite a few levels of a computing system, and one of them is the job scheduler. The goal is to design scheduling policies, which minimize energy consumption while guaranteeing some level of quality of service experienced by the users. These policies are as important for big computing centers, who need to reduce their electricity bill than they are for embedded systems, who need to augment the energetic autonomy.

The studied environment concerns computing systems with last generation microprocessors (like Intel SpeedStep, AMD PowerNow or IBM EnergyScale) which can change dynamically their clock speed, and hence influence on the energy consumption and quality of service at the same time. We observe a similar situation in production sites, where in order to satisfy an increasing workload, one needs to use additional resources, which have higher production costs. The electricity market is a typical example.

In this thesis we study decentralized computing systems, where non-cooperative users submit their jobs to a unique processor, with speed scaling ability. The goal of the machine is to balance between energy consumption and offered quality of service. This reflects the economic models adopted by an increasing number of software editors, who do not propose programs to be installed locally, but instead offer a web interface to the program running in the *cloud*. This situation arises also in Smartphones and personal computers, where applications are written by different editors, but run in concurrence on a single machine.

2.1.1 The scheduling model for minimizing the energy consumption

Scheduling problems form a large class of optimization problems, which have been studied since over 50 years. Like all research domains, it contains classical results, famous open problems, and regularly new problems generated by new application areas.

One of these new application domains is scheduling with the goal of energy minimization, in the speed scaling context. The simplified model is the following. Rather than to work on a fixed speed, new type of processors can work under variable speed s, where $s(t)$ is the speed at time t. In this context, the work done during an interval I is simply $\int_{t\in I} s(t)\ \mathrm{d}t$, which stands for a number of instructions executed during I. At the same time the energy consumption is proportionally to $\int_{t\in I} s(t)^\alpha dt$ for some given physical constant $2 < \alpha < 3$. This energy consumption is due to the heat dissipation at the surface of the processor, which needs cooling in order to prevent damage of the machine. Other models of energy consumption have been considered, involving several hibernation levels (idle, suspended, hibernated, shutdown), or involving parallel machine. In this thesis we restricted ourself on the case of a single speed scaling processor.

The input to a scheduling problems usually consists of n jobs. Every job i has a workload w_i, representing a number of instructions generated by its execution. But typically the job comes also with a time window $[r_i, d_i)$ in which the job must be executed. The time r_i represents a release time, before which the job cannot be scheduled, and d_i a deadline, representing a guarantee on the completion time of the jobs, which the schedule needs to fulfill.

A schedule is defined by two functions, the speed $s : \mathbb{R}_+ \to \mathbb{R}_+$ and a job assignment job: $\mathbb{R}_+ \to \{idle, 1, \ldots, n\}$. The idea is that the job i is executed during all time moments t such that $job(t) = i$. Then the problems — which is now classic in the speed scaling domain — consists in producing a schedule for n given jobs, respecting all given time windows, and minimizing overall energy consumption.

After a first proposal of a polynomial time algorithm for this problem in 1995 [63], many articles proposed improvements. Today the best known algorithm has complexity $O(n^2 \log n)$ [43] and for the special case where jobs have no release times — in other words $r_i = 0$ for all jobs i — there is an algorithm running in time $O(n \log n)$.

The ingredients to these algorithms are the following. The convexity of the energy consumption function $s \mapsto \int_{t\in I} s(t)^\alpha dt, \alpha \in [2, 3]$ implies that during all time moments where job i is scheduled, the speed is the same. The proof of this claim is subtle because the object of the minimization, meaning the functions s and job describing the schedule, are continuous. Thus the proof is based on the Karush-Kuhn-Tucker (KKT) conditions [9]. However if we assume that these functions are piecewise constant, a simple averaging argument is sufficient.

The second ingredient considers the density of the intervals. The density of an interval I is defined as the total workload over all jobs that need to be scheduled in interval I, divided by the length of I. Clearly the average speed in I of a schedule must be at least this density, otherwise the total work done in I is just insufficient for these jobs. The key to a polynomial time algorithm is the observation that in an optimal schedule, the speed during an maximal density interval I, is simply constantly this density. The idea of a polynomial time algorithm is therefore to identify a highest density interval I, to schedule all included jobs with constant speed equal to the density, and then to ignore both these jobs and the interval from the time line, and repeat until all jobs are scheduled.

The actual ordering of jobs in an interval I follows the *Earliest Deadline First* (EDF) policy, which consists simply in scheduling at every moment of T, the most urgent job i (with the smallest deadline d_i) among all already release but not yet completed jobs. If the speed is chosen as described above, then such a schedule will always respect the deadlines of the jobs. Otherwise a missed deadline d_j could be associated to an interval of the form $[r_i, d_j]$ of strictly larger density than I, contradicting the choice of I. Figure 2.1.1 shows an example of such an energy minimizing schedule.

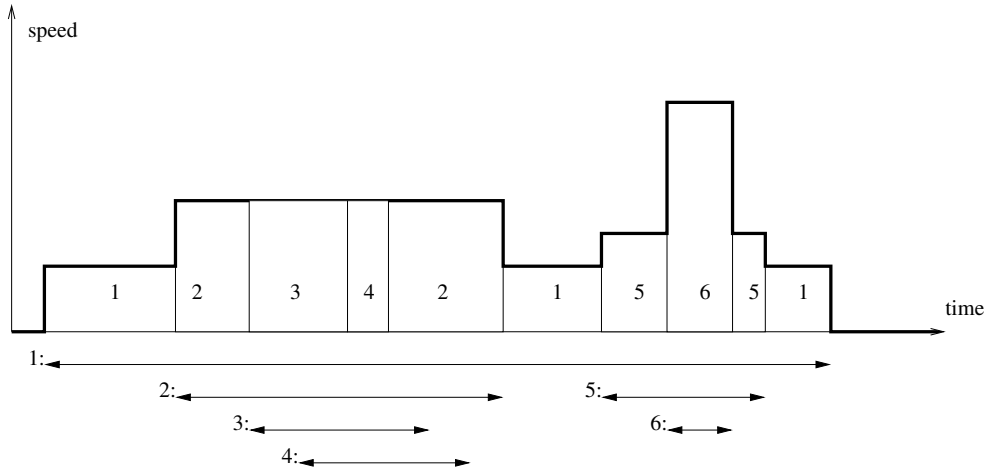


Figure 2.1: Example of a schedule minimizing the energy consumption for 5 jobs. The intervals indicate the intervals $[r_j, d_j]$ for every job j, the bold line the speed $s(t)$ and the labels in the rectangles, the job assignment $job(t)$.

2.2 Overview

Formally we study in this thesis the following bi-criteria scheduling problem. Given a speed scaling processor and a given number of jobs, we want to minimize at the same time the energy consumption and the average flow time of the jobs, which is defined as the difference between the completion time and the release time. Our objective is then to minimize a weighted sum of these two quantities, and we weight the flow time with given penalty factors. We call this quantity the social cost, and approach it from different perspectives, a decentralized one and a centralized one.

For the decentralized point of view, we approach the problem in form of a strategic game, where every player has a single job, which she submits with some additional information to the machine. This machine produces a schedule with the given jobs. All jobs are available from time 0 on. We are facing here an interesting conflict of interest. The players are interested in one thing, make their job complete as early as possible, which sounds quite natural, who likes to wait? However the machine follows a different goal, it wants that the produced schedule consumes as little energy as possible. This is also a natural goal, for economic and ecological reasons.

In this context we look for ways to charge the consumed energy to the users, to make them sensible as well to this aspect of the schedule. These ways are called *charging schemes* or *mechanism designs*.

The situation from the users point of view is the following. Every player wants to minimize a weighted sum of the completion time of his job, and of the cost charged by the machine. This is the key to a good charging scheme, in order to incite the players to adopt a good behavior one needs to relate individual penalties with the social cost, through a well defined charging scheme. Thus when player minimize their penalties, at the same time in some manner they also minimize the social cost. In order to make a sum possible between these two quantities of different nature, time and energy, we convert them into monetary units, and the conversion factors are hidden in the coefficients of the linear combination of those costs. In essence every

player i wants to minimize the sum $p_iC_i + b_i$, where p_i is a priority factor of player i, C_i is the completion time of his job and b_i the amount of her cost share (b for *bill*).

We face a very simple scheduling problem, where the individual penalty of each player is linear in the completion time of her job. In the scheduling literature quite a few other objective functions have been considered, including threshold functions, penalizing whenever the job completes after some deadline. For this thesis we restricted ourself to this linear penalty, called flow time, which is one of the most studied objective, as shown by the query log of the The Scheduling Zoo webpage [19].

This type of game is called a strategic game. It contrasts with repeated games, like chess for example, in the sense, that it consists of a single round. Players submit their jobs with chosen values. These values are called strategies. All announced values and the job parameters are public information and known to the other players, even though private information could exist in the game, which players won't reveal.

The machine computes a schedule for all jobs, and charges the consumed energy to the users. The algorithm used to produce the schedule and to compute the cost shares are known to all players. Hence a player who knows the choice of all other players has all necessary information to decide which is the best strategy for her. We are concerned with several desirable properties of such a game.

- First of all it is desirable that *pure Nash equilibria* exist. This means that for a given set of players, there is a strategy for each one (the vector of strategies is called a *strategy profile*) such that no player has an incentive to deviate from her strategy. In other words, the strategy of each player i minimizes her penalty when the strategies of all other players are fixed. Such a strategy is called a *best response*. Note that in this definition, we don't allow simultaneous changes in the strategies by a group of players, which would be called a *coalition*. The term *pure* is chosen in opposition with *mixed Nash equilibria*, where users choose a probability distribution over strategies. The famous Nash's Theorem states that mixed Nash equilibria always exist in finite strategic games, which does not hold for pure Nash equilibria.

- In the case where pure Nash equilibria exist, we would like that it is possible to find them in a time polynomial in the number of players. Also we would like that the computation of the best response for a single player is a polynomial time computable problem. All these requirements are clearly important for a realistic game.

- Different Nash equilibria can have different qualities. For a fixed social cost function, we would like that the social cost of any Nash equilibria is as close as possible from the social optimum. Note that the optimum does not have to be an equilibrium. A guaranty on the ratio between the two quantities is called the *price of anarchy*. Another quantity connected to the price of anarchy is the *price of stability*. It is an upper bound on the ratio between the best Nash equilibrium and the social optimum. This quantity concerns equilibria which are suggested or imposed by the game regulator to the players, rather than to let them find one by themselves. These two measures have been introduced by the seminal work of Koutsoupias and Papadimitriou and have been studied for a large number of games since, see the books [21, 49]. In the context of charging schemes, sometimes another term is used, we say that the game is v-efficient, if every Nash equilibria has a social cost at most v times the optimum.

- From the point of view of the game regulator, we want the amount charged to users to match the consumed energy. But when this is not the case, we want the amount to be close as possible. We say that the mechanism is γ-*budget balanced* if the total charged amount is at least the energy consumption and at most γ times this quantity.

- In case there are private information involved in the game, and that the values announced by the players can deviate from the true private values, we would like that announcing the true value is a best response. We also say that this strategy is dominant. In this case the game is called *truthful* or *strategy proof.* This concept will become more clear later, when we will make precise definitions. But the important aspect of it is that such a property makes life easy for the players, computing the best response is a trivial computational problem. And it is an important property for the regulator, who can therefore optimize the social cost, which is impossible if he is missing information.

For the centralized context, we consider the problem of computing the social cost, meaning the problem of computing a schedule minimizing a linear combination between the consumed energy and the weighted completion times. It turns out that this problem has the shape of a classical scheduling problem, but with an objective function only rarely studied. To understand its complexity, we study the complexity for related objective functions, in order to detect NP-hardness proof techniques that could be applied, or reductions leading to approximation schemes. In the absence of a polynomial time algorithm for our problem, we analyze dominance properties on the job orders. To complete this part we conduct an experimental study to measure the impact of the newly obtained rules.

2.3 Organization of the manuscript

The thesis is composed of 6 chapters. Chapters 1 and 2 are the introduction and presentation at a glance of the results both in French and English. The chapters 3 to 5 present in detail the contributions of this thesis. The last chapter concludes with future research directions.

In the chapter 3 we consider two types of games for the decentralized problem of minimizing the social cost. In the first game, players control the completion time of their job, simply by announcing a deadline d_i, which has to be respected by the machine. However in the second game, the players do not decide directly the completion time of their jobs, but announce a penalty factor p_i. This value indicates the importance of the completion time for the player. Intuitively if this value is large then the player is ready to pay a lot to make sure that his job completes early. For both games we study the existence of pure Nash equilibria, the convergence time to equilibria, and the ratio between the consumed energy and the total charged amount.

Chapter 4 is devoted to the centralized problem of minimizing the social cost. Here we show that the problem can be stated as a scheduling problem denoted by $1||\sum w_i f(C_i)$ [26] where $f(t) = t^{(\alpha-1)/\alpha}$ is a particular increasing concave penalty function. The complexity of this problem is open and we establish a state of the art of this scheduling problem for a large class of penalty functions. In particular we are interested in concave functions, and study a particular NP-hardness proof, which is widely used for scheduling problems.

Chapter 5 concerns the function $f(t) = sign(\beta) \cdot t^{\beta}$ for an arbitrary constant $\beta \in \mathbb{R}$ where $sign(\beta) \in \{-1, 0+1\}$ is the sign of β. The objective function $\sum w_i f(C_i)$ of this form of f has

attracted the attention of the community in particular for $\beta = 2$. In this chapter we study dominance properties which are useful for exact and exhaustive solvers.

In chapter 7 we present our conclusion and mention questions that we would like to answer in future work.

2.4 The results at a glance

2.4.1 Optimizing social cost: two decentralized approaches

We define two games that differ by the strategy set available to the players. In both cases we aim the following desirable properties of a game:

- existence of pure Nash equilibria,
- bounded overcharge,
- truthfulness.

2.4.1.1 deadline game

We study a game where the players control the completion time of their job. To illustrate this possibility, we refer to the situation where we buy in a shop the service of printing our photos, and typically have the possibility to get the job done within 2 hours or say 2 days at different prices.

To be precise, the player i has a job i, with a workload w_i, a release date r_i and penalty factor p_i. With the submission of his job, she announces a deadline d_i, which the regulator has to respect. This last value represents the strategy the player i has to choose, while w_i, r_i are given public values, out of control of the player. The deadline penalty p_i is a private value.

The game regulator computes a schedule which respects all execution time windows of the form $[r_i, d_i)$, and minimizes total energy consumption. This problem can be solved in polynomial time, as explained in section 2.1.1. Afterwards this energy is charged to the players. The individual goal of each player is to minimize her cost charge plus her weighted deadline penalty $p_i(d_i - r_i)$.

Note that we could have defined the weighted deadline penalty as the weighted flow time $p_i(C_i - r_i)$, without really changing the nature of the game. Indeed if for a player $C_i < d_i$, then reducing the announced deadline for i, preserves her individual cost. For simplicity we decided for the cost $p_i(d_i - r_i)$. Also since $p_i r_i$ is independent of the schedule, we will ignore from now on this quantity. In summary the penalty of player i is

$$p_i d_i + b_i,$$

where b_i is the amount of her cost share.

How to charge the users? The first charging scheme that comes into mind is to charge to player i precisely the energy consumed by her job i in the produced schedule. We call it the *proportional cost sharing mechanism*. It seems rather fair, since every player pays her contribution in the energy consumption. Also there is no overcharge, since the total

cost shares equal the energy cost by definition. So it seems a rather perfect cost sharing mechanism, if there were not a serious drawback.

The game does not guarantee existence of pure Nash equilibria. The proof of this claim is a very simple example consisting of two identical jobs. The hard part is an analysis of the best response functions, showing that there is no fix-point in the best response dynamics. The idea is that since jobs have to be scheduled in some order, the produced schedule generates an asymmetry between the otherwise identical jobs. Therefore there is always a player that can improve her individual penalty by changing her strategy, and in certain way take the role of the other player.

Theorem 2.1. *For the proportional cost sharing mechanism, the existence of pure Nash equilibria is not guaranteed.*

We propose another cost sharing mechanism, where each player pays the difference between the optimal schedules for all player and for all player but her. We call it the *marginal cost sharing mechanism*. Intuitively we imagine the situation where player i joins the game. Then the cost of the energy minimal schedule increases by her presence, and it is this increase which is charged to player i.

By construction we obtain an exact potential game. This means that there exists a potential function which depends solely on the players strategies, and when a player changes her strategy generating a change of Δ in her individual cost, then the potential changes by the same amount Δ. Exact potential games are very well understood, and admit a lot of nice properties. The potential function happens to equal the social cost function. Therefore a pure Nash equilibria which minimizes the potential is a social optimum.

For finite strategic games, meaning when the strategy space is finite, exact potential games always admit pure Nash equilibria [47]. In our game we encounter the difficulty that the announced deadlines by the players are positive real numbers, and therefore a simple local search argument is not enough to prove the existence of a pure Nash equilibrium. It could be that such a local search is an infinite process, and it turns out that this is indeed the case with our game.

However we are lucky and the strategy space is continuous and closed in some sense, which permits to conclude the existence of minimum for the potential function, which is therefore also a pure Nash equilibrium.

Theorem 2.2. *For the marginal cost sharing mechanism, the game is an exact potential game, which admits always a pure Nash equilibria. However the best response dynamics can have infinite convergence time.*

In comparison with the proportional cost sharing mechanism, the existence of equilibria is a nice feature, but it comes with the price of giving some slack in the overcharge. However using upper bounds techniques, we were able to bound the overcharge by a small factor.

Theorem 2.3. *For the marginal cost sharing mechanism, the game has overcharge α at most. This is ratio is tight as shown by a some family of game instances.*

However with this mechanism the game does not have a constant factor guarantee for the price of anarchy. We don't talk about truthfulness here, as there is no notion of private value in this game.

2.4.1.2 penalty game

We consider a game which differs from the one of the previous section. Here the player i does not decide directly the completion time of her job, but announces a penalty factor p_i. This penalty indicates how much it is worth for player i that her job completes early. Intuitively this value is large if the player accepts to pay a lot to make sure her job completes early.

In this game each player i submits her job, and announces the workload w_i and a penalty factor $\hat{p}_i$, which could differ from p_i if it advantages the player. However without loss of generality we can assume that the player cannot lie on the workload w_i, since this value can be verified during the execution of the schedule, and not truthful players can be punished in case of a difference.

In this context, jobs have no release times and are therefore available from time 0 on. Also jobs do not come with deadlines, it is the task of the regulator to decide freely for a schedule, which considers both the weighted completion time of the jobs and the energy consumption.

As always the social cost is defined as the total weighted completion time plus the consumed energy. Minimizing the social cost, is a scheduling problem to which section 2.4.2 is devoted. We mention here that only an approximation scheme (PTAS) is known for the problem, even though it admits interesting structural properties, which we describe now.

Since all jobs are available from time 0 on, the optimal schedule executes all jobs without idle time, nor preemption, until the completion of the last job. Therefore there exists a permutation π which describes the order in which the jobs are scheduled, where $\pi(i)$ is the rank of job i in the schedule. In addition since every job is scheduled at constant speed, it is enough to describe in a vector ℓ the execution durations of the jobs. The pair π, ℓ specifies completely a schedule.

Two costs are associated to the schedule. First there is the energy consumption

$$E(\ell, w) := \sum_i w_i^\alpha \ell_i^{1-\alpha},$$

which has this form since each job i is scheduled during time ℓ_i with a speed w_i/ℓ_i. The second cost is the total weighted completion time, also called weighted flow time. It is

$$F(\pi, \ell, p) := \sum_j p_j C_j$$

where C_j is the completion time of the job j, which is defined as the sum of ℓ_i over all jobs i with rank $\pi(i) \leq \pi(j)$. The difficulty which we encounter in this game, is that is seems impossible to achieve all three properties at the same time: truthfulness, efficiency and budget balancedness. We propose therefore a mechanism that gives up by a constant factor the budget balancedness to guarantee the truthfulness of the players.

This mechanism however has a serious drawback, namely that it does not guarantee constant price of anarchy. We were unable to correct this problem, and the main reason is that we have to fix an order on the jobs, which is independent of the players strategies, in order to avoid their influence. This property is crucial for truthfulness.

Our charging scheme uses a fixed execution order π on the jobs. This order could be arbitrary, or random, it does not matter. The important property is that players cannot influence this order. Let ℓ be the vector of execution durations, which minimizes the social cost $E(\ell, w) + F(\pi, \ell, \hat{p})$. This vector can be computed in polynomial time. (Remember, π is fixed). Let $E(OPT_\pi(\hat{p}))$ be energy consumption of this schedule, which is $E(\ell, w)$. In

addition let $E(OPT_\pi(\hat{p}_{-i}))$ be the optimal energy consumption when job i is not part of the schedule. The difference represents the increase of the consumed energy by the presence of i in the game. Player i will be charged the following value

$$b_i := \alpha(E(OPT_\pi(\hat{p})) - E(OPT_\pi(\hat{p}_{-i}))) - \hat{p}_i C_i,$$

where C_i is the completion time of job i in the schedule defined by π, ℓ. The idea behind this charging scheme is that the penalty of player i becomes

$$\alpha(E(OPT_\pi(\hat{p})) - E(OPT_\pi(\hat{p}_{-i}))) + (p_i - \hat{p}_i)C_i,$$

which for the $\widehat{p_i}$ dependent part is quite close to the social cost. Relating the individual penalty with the social cost is the key to an efficient charging scheme: since the players minimize in the same direction as the social cost. If a player tries to minimize $(p_i - \widehat{p_i})C_i$, then clearly he has to announce the true value $\widehat{p_i} = p_i$. Our contribution is to show that if he tries to minimize his total penalty, then the best response is also the true value p_i. The proof uses a first and second analysis of the penalty in $\widehat{p_i}$.

Theorem 2.4. *The game where the execution order is fixed, is truthful. It overcharges with a factor $\alpha + 1$ at most.*

Concerning the price of anarchy, by construction the mechanism computes the schedule which minimizes the social cost $E(\ell, w) + F(\pi, \ell, \hat{p})$, and is therefore optimal for the fixed execution order π. However it can be arbitrarily large with respect to the social optimum, where minimization is also done over all execution orders.

2.4.2 Optimizing social cost: a centralized approach

2.4.2.1 Reduction to a scheduling problem

We study the optimization problem of minimizing the social cost, which consists of the sum between the energy consumption and the total weighted completion time.

- In section 2.1.1 we mention that the problem of minimizing the energy consumption admits a polynomial time algorithm when all job deadlines are given [43].
- On the other hand, the complexity of the problem of minimizing total weighted completion time subject to a given energy budget is open. Only for the special case of unweighted jobs ($w_i = 1$) a polynomial time algorithm has been found [50].
- The complexity of the general problem of minimizing energy consumption plus total weighted completion time is open.

Formally, our problem can be stated as follows.

Problem A Given n jobs with workloads w and penalty factors p, find a schedule defined by π, ℓ which minimizes $A_{w,p}(\pi, \ell) = E(\ell, w) + F(\pi, \ell, p)$.

This problem is equivalent to another scheduling problem. Here the machine runs at uniform speed, and the role of the speed is encoded in the objective function. Note that the two values of the input w, p play opposite roles in both problems.

Problem B Given n jobs with priority weights w and processing times p, find a schedule defined by an order σ which minimizes $B_{w,p}(\sigma) = \sum w_j C_j^{(\alpha-1)/\alpha}$, where the completion time C_j is defined as $\sum_i p_i$ over all jobs i with $\sigma(i) \leq \sigma(j)$.

We show equivalence between the two above problems. Our proof uses the Hessian conditions. Independently, a similar reduction has been discovered in [44] using the Karush-Kuhn-Tucker conditions, relating Problem B and a variant of Problem A, with the goal of minimizing total weighted completion time under a given energy budget.

Theorem 2.5. *Let π be a job order, and denote by σ its reverse, i.e. $\sigma(i) = n+1-\pi(i)$. Let ℓ be the execution length vector minimizing $A_{w,p}(\pi, \ell)$. Then $A_{w,p}(\pi, \ell) = \alpha(\alpha-1)^{(\alpha-1)/\alpha} \cdot B_{w,p}(\sigma)$.*

Note that the result shows that a bi-criterion problem can be reduced to a mono-criterion scheduling problem, which has a structure close to scheduling problems studied in the literature. Unfortunately this problem is open in the sense that a polynomial algorithm or a proof of NP-completeness is not known. The absence of polynomial algorithm can be explained by the lack of dominance structures that normally lead to dynamic programming solutions, linear programming formulations or greedy algorithms. On the other hand, the continuous nature of the objective function is a difficulty to encode an NP-hard problem, since a reduction needs binary decisions. The only result known for the problem B is the polynomial approximation schema proposed by Megow and Verschae [44]. On the other hand, the problem B belongs to a class of scheduling problems that have been studied in the past.

2.4.2.2 On complexity of the scheduling problem

Motivated by the study of the scheduling problem B, we establish a survey and study the complexity of penalty functions with the following form. Consider n jobs, each job j has some processing time p_j and priority weight w_j. The jobs have to be scheduled on a single machine in order to minimize a particular objective function of the form $\sum_j w_j f(C_j)$, where C_j the completion time of job j and f is a problem specific penalty function. Extending the three-field notation [26] our problem could be denoted as $1||\sum w_j f(C_j)$.

The study of different penalty functions has several motivations. The first one is due to Theorem 2.5 for $f(t) = t^{\frac{\alpha-1}{\alpha}}$ with $2 \leq \alpha \leq 3$. The second motivation concerns machines with speed varying over time. If we denote by s the speed function, then after t_0 time units, the machine processed a workload of $\int_0^{t_0} s(t)dt$ units. In this setting, minimizing the weighted sum of the completion times reduces to minimizing the value $\sum w_j f(C_j)$ for a constant speed machine, where f is the inverse of the integral of s. Varying speed can be the consequence of a learning effect, upgrades, or even varying available energy or tear and wear of the machine.

In our survey, we study penalty functions with several elementary properties such as the following:

Monotonicity of f This natural property represents the fact that it is usually better to finish a job early than late. Except for just-in-time problems, where the goal is to finish around to a common deadline d, this property holds for the usual penalty functions.

Convexity or concavity of f Imagine a situation where the completion time of a job is increased by some value δ. The corresponding increase in the penalty clearly depends on δ but could also depend on the actual completion time of the job. In the speed

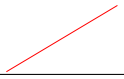
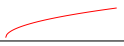
plot	function	complexity status	plot	function	complexity status
	t	$O(n \log n)$ [54]		e^{rt}	$O(n \log n)$ [51]
	$\max\{0, t-d\}$	Binary NP-hard [64] DP: $O(n^2 d)$ [42] FPTAS : $O((n^6 \log \sum_j w_j)/\varepsilon^3)$ [39], $O(n^2/\varepsilon)$ [37]		$1-e^{-rt}$	$O(n \log n)$ [51]
	$t+\max\{0, t-d\}$	DP:$O(n^2 d^2)$ [60] FPTAS: $O(n^3 \log d/\varepsilon^2)$ [60]		t^2	Open
	$\max\{0, d-t\}$	Binary NP-hard [14, 18] DP: $O(n^2 \sum_j p_j)$ [14] FPTAS :$O(n^4/\varepsilon)$ [14]		$\sqrt{t}$	Open
	$\min\{t, d\}$	DP: $O(nd)$ FPTAS:$O(n^2/\epsilon)$		$\lvert d-t\rvert$	Binary NP-hard [27, 33] DP: $O(n^2 d)$ [33], $O(n \sum_j p_j)$ [27] FPTAS: $O(n^6/\varepsilon^3)$ [40]
	piecewise linear	Unary NP-hard [32] PTAS: $2^{O(\log(1/\varepsilon)/\varepsilon^2)} \log(\sum_j w_j) n$ [44]		$(d-t)^2$	Results only for equals weight DP: $O(n \sum_j p_j)$ [61], $O(n^2(d+\max p_j))$ [46], $O(nd)$ [55]

Table 2.1: Results for several penalties functions

scaling context, the convexity implies a constant deceleration over time, whereas the concavity represents a constant acceleration.

Piecewise linearity of f, models constant speed with speed speed changes of the machine at specific times, which could be the effect of a machine update at some moment during the schedule. In a production environment, for example, penalties for a late delivery change when a certain critical date is exceeded.

Table 2.1 shows the known complexity results for different penalty functions f. We consider f as a function of time t and d, w constant parameters.

NP-hardness proofs for scheduling problems of this form, often consist in a reduction from PARTITIONING [25] and involve instances, where all jobs j have the same Smith-ratio w_j/p_j, with the exception of a single job. We illustrate the technique, using the scheduling problem with a penalty function $f(t) = \max\{0, t-d\}$ (see [64])

An instance of the PARTITION problem is defined as follows: Given $a_1, \ldots, a_n$ integers, the problem is to decide whether it is possible to partition $\{1, \ldots, n\}$ into two sets $S, \bar{S}$ such that

$$\sum_{i \in S} a_i = \sum_{i \in \bar{S}} a_i. \tag{2.1}$$

In order to encode a *Partition* instance as an instance to the scheduling problem, we will use the fact that a job has only a contribution to the objective function if it is completed not later than d. Thus, the deadline d allows naturally a partition the jobs into two sets — those completing after or before d. In equation (2.1), there is no notion of an order among

the elements and therefore we need also this property in the encoding. To this purpose, we generate an instance as follows: consider n jobs, each job i has a processing time $p_i := a_i$ and priority weight $w_i := a_i$. Here, we use Smith's result [54] i.e., stating that for equal Smith-ratio instances, the value of objective function is independent of jobs order after d (and obviously, also before d). The reduction is completed by the $(n+1)$-th job, which must be completed in time d in an optimal schedule. This job has a small processing time and a density w_{n+1}/p_{n+1} greater than 1. To complete the proof, we suppose that all values are integers.

The previous reduction shows that the scheduling problem for the penalty function $f(t) = \max\{0, t-d\}$ is NP-hard already for the almost equal Smith-ratio instance. These are instances where all jobs i, except possibly one, have the same density w_i/p_i.

We show that this class of instances is easy for any increasing concave penalty function.

Theorem 2.6. *Almost equal Smith-ratio instances can be solved optimally in time $O(n \log n)$ for any increasing concave penalty function.*

Clearly, this result does not exclude the possibility that the problem with an increasing concave penalty functions is NP-hard, but it shows that fundamentally new NP-hardness proof techniques would be needed for such a result.

Given the above result, we study resolution methods for this problem class. In particular, we are interested in the increasing concave penalty function $f(t) = \min\{t, d\}$. It corresponds to the scheduling problem for minimizing the total weighted completion time on a single machine, which has a speed 1 from time 0 to time d and infinite speed afterwards. The complexity status of this problem is open.

In this work, we have proposed a reduction to the problem of *minimizing half-products*, which gives rise to a strongly polynomial approximation scheme (FPTAS).

Definition 2.1. *A half-product is a pseudo-boolean function of the form:*

$$F(x) = \sum_{j=1}^{n} a_j x_j \sum_{i=1}^{j-1} b_i x_i - \sum_{j=1}^{n} c_j x_j + D$$

where $x \in \{0,1\}^n$, $a = (a_1, ..., a_n), b = (b_1, ..., b_{n-1})$ are non-negative integer vectors, $c = (c_1 ..., c_n)$ is an arbitrary integer vector, and D an arbitrary integer.

The name derives from the fact that is contains only half of the elements from the complete product $(a \cdot x)(b \cdot x) + c \cdot x + D$, where $\cdot$ is the inner product. Many problems can be modeled as a problem of minimization of half-products [5]. This is also the case for our scheduling problem.

Theorem 2.7. *The scheduling problem with a penalty function $f(t) = \min\{t, d\}$ can be modeled as the minimizing problem of half-products, considering*

$$\begin{aligned} a_j &= w_j \\ b_j &= p_j \\ c_j &= (d - p_j) w_j \\ D &= d \sum w_j. \end{aligned}$$

Therefore, an approximation scheme for our scheduling problem is implied by the Theorem of Erel and Ghosh.

Theorem 2.8 ([23]). *There is an approximation scheme for minimizing half-products F in time $O(n^2/\varepsilon)$ subject to* $\min F > 0$.

A consequence of the above theorem is that the scheduling problem cannot be NP-hard in the strong sense, however it could be NP-complete in the weak sense.

Finally, we present results for a more comprehensive framework, where each job has a different penalty function.

Theorem 2.9. *The scheduling problem* $1||\sum w_j \min\{C_j, d_j\}$ *is NP-hard.*

2.4.3 Order constraints properties

This section addresses the exact resolution of the scheduling problem B. In particular we consider the algorithm A* and show dominance properties for optimal job orders. The section ends with an experimental evaluation of our new rules. We fix a general penalty function of the form $f(t) = sign(\beta) \cdot t^\beta$ for an arbitrary constant $\beta \in \mathbb{R}$ where $sign(\beta) \in \{-1, 0, +1\}$ is the sign of β. The scheduling problem consists in finding an order on the given jobs, which minimizes $\sum w_j f(C_j)$. In the absence of a polynomial algorithm, exact resolution methods have been proposed. In particular there is a reduction to a shortest path problem. The reduction follows the following lines: the search space is the directed acyclic graph consisting of all subsets $S \subseteq \{1, \ldots, n\}$. In this graph for every vertex S there is an arc to $S\backslash\{j\}$ for any $j \in S$. It is labeled with j, and has cost $w_j t^\beta$ for $t = \sum_{i \in S} p_i$. Every directed path from the root $\{1, \ldots, n\}$ to $\{\}$ corresponds to a schedule of an objective value being the total arc cost. Thus, the scheduling problem is reduced to a shortest path problem, where the potential search space has size 2^n, which is already less than the space of the $n!$ different schedules. To compute shortest path in this setting the algorithm A* seems well suited. It is a refined variant of Dijkstra algorithm, which explores the graph using a priority queue Q containing arcs pointing to vertices that still need to be visited. An arc (S, S') has a weight corresponding to the distance from the root to S' through this arc. The improvement of algorithm A* with respect to Dijkstra's is to add to this weight a lower bound for the distance from S' to $\{1, \ldots, n\}$. In practice, this improvement is important, allowing to resolve very large instances. For our implementation, we chose the basic lower bound $\sum_{j \notin S'} w_j f\left(p_j + \sum_{i \in S'} p_i\right)$, which balances the quality of the lower bound and the computation time required for it. This approach has been introduced in 1972 by [28], and has been applied to this class of scheduling problem by [52]. Recently it was improved for the quadratic penalty function $f(t) = t^2$ by Höhn and Jacobs [30].

We consider two variants of the above mentioned algorithm. In the *forward* approach, a partial schedule describes a prefix of length t of a complete schedule and is extended to its right along an edge of the search tree, and in this variant the basic lower bound is $\sum_{i \in S} w_i (t + p_i)^\beta$. However in the *backward* approach, a partial schedule S describes a suffix of a complete schedule and is extended to its left. We conducted an experimental study in order to determine which variant gives better results in which case.

The Algorithm A* can be improved by incorporating dominance rules. These are of two types, local precedence rules and global precedence rules. Consider a schedule where i, j are scheduled one after the other, with job i starting at time t. Then the exchange of jobs i, j

has some effect to the objective value of the schedule, which depends on jobs i, j and on time t. We denote by $i \prec_{\ell(t)} j$ the property that the order i, j generates a strictly smaller cost than the order j, i. If it holds for all time points $t \in [a, b]$, then we denote this property by $i \prec_{\ell[a,b]} j$, and if it holds everywhere we denote it simply by $i \prec_{\ell} j$.

Global precedence is defined similarly. If every schedule scheduling job j before i is sub-optimal, no matter if i, j are adjacent or not, then we say that the job i, j satisfy a global precedence order denoted by $i \prec_g j$.

This order can be generalized to intervals $[a, b]$, where $i \prec_{g[a,b]} j$ basically means that every schedule executing the jobs i, j in $[a, b]$ and in the order j, i is sub-optimal. The exact definition is omitted, in this summary. Here the important point is that this property permits to cut arcs in the search graph and improve the performance of algorithm A*. Indeed the arc (S, S') for $S' = S \cup \{j\}$ can savely be removed from the graph without affecting the shortest path, if there is a job $i \notin S'$ with $i \prec_{g[a,b]} j$ for a being the total processing time of S and b the total processing time over all jobs of the instance. Several experimental studies for pruning rules based on order properties have been proposed in the literature ([3, 7, 17, 53, 56, 57]), and were mainly focused on the quadratic penalty function, i.e. $\beta = 2$. In 2000, Mondal and Sen [45] conjectured that $\beta = 2, (w_i \geq w_j) \wedge (w_i/p_i > w_j/p_j)$ implies the global order property $i \prec_g j$, and provided experimental evidence that this constraint would significantly improve the runtime of a branch-and-prune search. Recently, Höhn and Jacobs [30] succeeded to prove this conjecture. In addition they improved local and global order conditions and generalized them to integer constants $\beta \geq 2$. An extensive experimental study analyzed the effect of these rules to the performance of the algorithm A*.

In summary, we obtain the following rules, illustrated in Figure 2.2.

Theorem 2.10 (our rules)**.** *Let i, j be two jobs with $p_i > p_j$.*

If $\beta > 1$		
If $(p_i/p_j)^\beta \leq w_i/w_j$	*then* $i \prec_g j$	
If $p_i/p_j < w_i/w_j$	*then* $i \prec_\ell j$	*[in case* $\beta = 2$ *is enforced by the implication* $i \prec_g j$*]*
If $w_i/w_j < \phi_{ij}(0)$	*then* $j \prec_g i$	
else $\exists t^* : w_i/w_j = \phi_{ij}(t^*)$ *and*	$i \prec_{\ell[0,t^*)} j$	
	and $j \prec_{g[t^*,\infty)} i$	
If $0 < \beta < 1$		
If $(p_i/p_j)^2 \leq w_i/w_j$	*then* $i \prec_g j$	
If $\phi_{ij}(0) < w_i/w_j$	*then* $i \prec_\ell j$	
If $w_i/w_j < p_i/p_j$	*then* $j \prec_{g[0,t_1]} i$	
else $\exists t^* : w_i/w_j = \phi_{ij}(t^*)$ *and*	$i \prec_{g[0,t^*)} j$	
	and $j \prec_{\ell[t^*,\infty)} i$.	
If $\beta = -1$		
If $(p_i/p_j)^2 \leq w_i/w_j$	*then* $i \prec_g j$	
If $(w_i/w_j) < p_i/p_j$	*then* $i \prec_g j$	
else $\exists t^* = \frac{w_j p_i^2 - w_i p_j^2}{w_i p_j - w_j p_i} \geq 0$ *et*	$i \prec_{g[0,t^*)} j$	
	and $i \prec_{g[0,t^*)} j$	

We state the following conjecture, motivated by partial results and experiments.

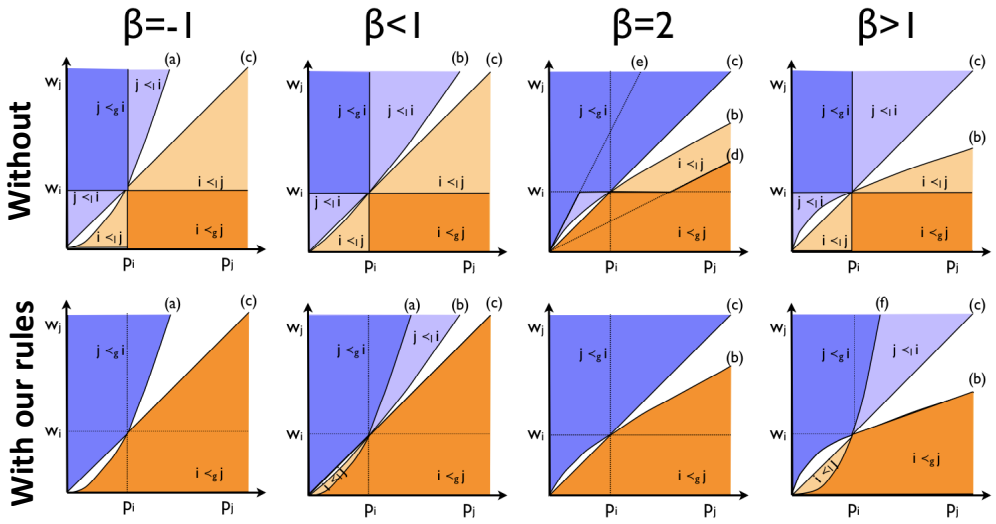


Figure 2.2: Job j compared to a fixed job i. Labels of particular functions: (a) $w_j = w_i(p_j/p_i)^2$, (b) $w_j = w_i((p_i+p_j)^\beta - p_i^\beta)/((p_i+p_j)^\beta - p_j^\beta)$, (c)$w_j = w_i p_j/p_i$, (d) $w_j = w_i p_j/2p_i$, (e) $w_j = 2w_i p_j/p_i$ et (f) $w_j = w_i(p_j/p_i)^\beta$.

Conjecture 2.1. *Consider the penalty function $f(t) = t^\beta$ for an arbitrary constant $\beta > 0$. Then for all jobs i, j, $i \prec_\ell j$ implies $i \prec_g j$.*

The conjecture for the case $\beta = 1$ is trivial, whereas our contribution is a proof for the case $\beta = 2$. For $\beta = -1$, the problem is to maximize $\sum w_j/C_j$ and we showed a stronger variant of the conjecture, which is $i \prec_{\ell[a,b]} j \Rightarrow i \prec_{g[a,b]} j$. For $\beta = 2$ this claim does not hold, as shown by a counterexample. In order to measure the impact of our rules, we implemented the algorithm A* and tested it against randomly generated instances, following a random model described by Höhn and Jacobs [30].

Formally, the instances of our main test sets are generated as follows: for each choice of σ and β, we generated 25 instances of 20 jobs each. The processing time of every job is uniformly generated in $\{1, 2, \ldots, 100\}$. When $\beta > 1$, the condition for $i \prec_g j$ of our rules can be approximated, when p_j/p_i tends to infinity, by the relation $w_i/p_i \geq \beta w_j/p_j$. Therefore in order to obtain a similar "hardness" of the random instances for the same parameter σ for different values of $\beta > 1$, we choose the Smith-ratio according to $2^{N(0,\beta^2\sigma^2)}$. This way the ratio between the Smith-ratios of two jobs is a random variable from the distribution $2^{2N(0,\beta^2\sigma^2)}$, and the probability that this value is at least β depends only on σ. However when $\beta = -1$ or $0 < \beta < 1$, the condition for $i \prec_g j$ of our rule can be approximated when p_j/p_i tends to infinity by the relation $w_i/p_i \geq 2w_j/p_j$, and therefore we choose the Smith-ratio of the jobs according to the β-independent distribution $2^{N(0,4\sigma^2)}$. We conducted an experimental study in order to find out which variant is most likely to be more efficient. The impact is more important for small σ values, since for large values the instances are easy anyway. The results indicate that without our rules the forward variant should be used whenever $0 < \beta < 1$ or $\beta \in \{-1, 2\}$, while with our rules the forward variant should be used whenever $\beta > 1$. During the execution of the algorithm A* a timeout was set, aborting executions that needed more than a million nodes. From our experiments we measure the instance sizes that can

be efficiently solved, and observe that this limit is of course smaller when σ is small, as the instances become harder. But we also observe that with the usage of our rules much larger instances can be solved. Finally, we measure the influence on the number of nodes generated during a resolution when our rules are used. For fairness we excluded instances where the timeout was reached without the use of our rules. Figure 2.3 shows the average improvement factor for some β and σ values.

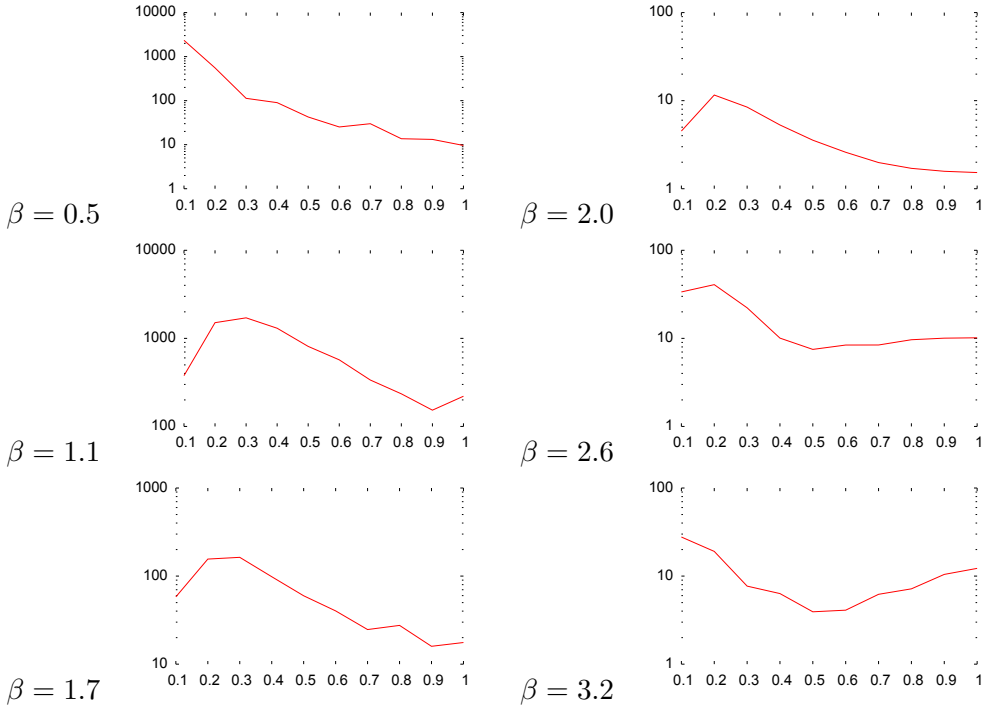


Figure 2.3: Improvement factor on the average number of nodes as function of β and σ. Every point represents an average over 25 instances of 20 jobs each.

2.5 Main Contributions

In summary, the main contributions of this thesis are:

- The design and study of games aiming the minimization of energy consumption and maximization of quality of service on a single processor in the speed scaling model, where non-cooperative users submit jobs to be executed. In particular, we studied the mechanism design of the game, and compared different methods to charge the energy consumption to the users, with the goal of inciting them to reach equilibria with social cost close to optimality. Formally, we have analyzed the existence of pure Nash equilibria, the convergence time to these equilibria, truthfulness and bounded by how much we overcharge the players. (see Chapter 3).

- Reduction of minimizing a linear combination between energy consumption and total weighted completion time to a classical scheduling problem. Formally, the problem is to order the jobs on a single machine with constant speed, minimizing $\sum_j w_j f(C_j)$,

where for each job j, w_j is the given priority weight and C_j is its completion time, and $f(t) = t^{(\alpha-1)/\alpha}, 2 \leq \alpha \leq 3$ is a fixed increasing concave penalty function. This result links a bi-criterion problem with a mono-criterion scheduling problem, which has a structure closed to scheduling problems studied in the past (see Chapter 4).

- The study of the complexity for several penalty functions f. In particular, we explained why a classical NP-completeness proof technique fails for the concave increasing penalty functions. In addition, we study the problem with the simple concave function $f(t) = \min\{t, d\}$, for which no polynomial time algorithm is known. We proposed a reduction to the problem of *minimization of half-products* which results in a strongly polynomial approximation scheme (FPTAS) (see Chapter 4).

- For the penalty function of the form $f(t) = \mathit{sign}(\beta) \cdot t^{\beta}$ where $\beta \in \mathbb{R}$ is a constant and $\mathit{sign}(\beta) \in \{-1, 0, +1\}$ is the sign of β, we stated a conjecture, which can be roughly stated as follows. If for two jobs i, j, any adjacent scheduling of these jobs privileges the order i, j, independently of their position in the schedule, then the order i, j is respected in any optimal schedule, whether these jobs are adjacent or not. A proof of this conjecture would have a significant impact on exact resolutions, as shown by our experiments, and we were able to prove it for the cases $\beta = 2$, whereas the case $\beta = 1$ has been settled sixty years ago. For the remaining cases, we were nevertheless able to provide dominance rules, which are weaker than the conjecture, but their contributions have been positively evaluated in experiments. For $\beta = -1$, the problem is to maximize $\sum w_j/C_j$ and corresponds to the so-called Airplane Refueling Problem and we showed even a stronger variant of the conjecture, restricting the order properties to intervals. In contrast this stronger variant is false for $\beta = 2$ (see Chapter 5).

Chapter 3
Optimizing social cost : two decentralized approaches

This chapter is based on the following articles.

- Mechanism Design Aggregating Energy Consumption and Quality of Service in Speed Scaling Scheduling, with C. Dürr and Ł. Jeż In *Proceeding of 9th Conference on Web and Internet Economics (WINE)*, 2013.

- Scheduling under dynamic speed-scaling for minimizing weighted completion time and energy consumption, with C. Dürr and Ł. Jeż, submitted to a journal.

3.1 Introduction

In many computing systems, minimizing energy consumption and maximizing quality of service are opposed goals. This is also the case for the speed scaling scheduling model considered in this chapter. Higher speed means that jobs finish earlier at the price of a higher energy consumption. It has been introduced in [35], and triggered a lot of work on offline and online algorithms; see [1] for an overview.

The online and offline optimization problem for minimizing flow time while respecting a maximum energy consumption has been studied for the single machine setting in [2, 9, 13, 50] and for the parallel machines setting in [4]. For the variant where an aggregation of energy and flow time is considered, polynomial approximation algorithms have been presented in [8,12,44].

In this chapter we propose to study this problem from a different perspective, namely as a strategic game. In society many ecological problems are either addressed in a centralized manner, like forcing citizens to sort household waste, or in a decentralized manner, like tax incentives to enforce ecological behavior. We propose incentives for a scheduling game, in form of an energy cost charging scheme.

Consider a scheduling problem for a common single processor, that can run at variable speed, such as the modern microprocessors Intel SpeedStep, AMD PowerNow! or IBM EnergyScale. Each job has some workload, representing a number of instructions to execute, and a release time before which it cannot be scheduled. Every user submits a single job to processor, declaring the jobs parameters. The processor will schedule the submitted jobs preemptively, so that all release times and the overall energy usage is minimized. The energy consumed by the schedule needs to be charged to the users. The individual goal of each user is to minimize the sum of the energy cost share and the completion time weighted by the user's priority, which represents a quality of service coefficient. This individual priority weight implies a conversion factor that allows of aggregation of deadline and energy.

3.2 The model

Formally, we consider a non-cooperative game with n players and a regulator. The regulator manages the machine where the jobs are executed. Each player has a job i with a workload w_i, a release time r_i and a priority p_i, representing a quality of service coefficient.

The regulator implements some *cost sharing mechanism*, which is known to all users. This mechanism defines a cost share function b_i specifying how much player i is charged. The penalty of player i is the sum of two values: his *energy cost share* $b_i(w, r, d)$ defined by the mechanism, where $w = (w_1, \ldots, w_n), r = (r_1, \ldots, r_n)$ and $d = (d_1, \ldots, d_n)$, and his *waiting cost*, which can be either $p_i d_i$ or $p_i(d_i - r_i)$; we use the former waiting cost throughout the article but all our results apply to both. The sum of all player's penalties, i.e., energy cost shares and waiting costs will be called the *utilitarian social cost*.

The regulator computes a minimum energy schedule for a single machine in the speed scaling model, which stipulates that at any point in time t the processor can run at arbitrary speed $s(t) \geq 0$; for a time interval I, the workload executed in I is $\int_{t \in I} s(t)\mathrm{d}t$, while the energy consumed is $\int_{t \in I} s(t)^\alpha \mathrm{d}t$ for some fixed physical constant $\alpha \in [2, 3]$ characteristic for a device [11].

The sum of the energy used by this optimum schedule and of all the players' waiting costs will be called the *effective social cost*.

The minimum energy schedule can be computed in time $O(n^2 \log n)$ [43] and has (among others) the following properties [63]. The jobs in the schedule are executed by preemptive earliest deadline first order (EDF), and the speed $s(t)$ at which they are processed is piecewise linear. Preemptive EDF means that at every time point among all jobs which are already released and not yet completed, the job with the smallest deadline is executed, using job indices to break ties.

Here we define two game in function the information announced by the player to regulator.

We call *deadline game*, the game where The player submits its job together with a deadline $d_i > r_i$ to the regulator. Workloads, release times and deadlines are public information known to all players, while quality of service coefficients can be private.

On the other hand, we say *penalty game* when the player submits his job, he announces the workload and some delay penalty $\hat{p}_i$ and all jobs are available from time 0 on. The announced value might differ from the real value, in order to influence the game towards his advantage, while the workload has to be the true value. The latter assumption makes sense, since it is a quantity observable by the operator, who could punish players in case they lied on the workload.

In latter situation, we note that a schedule is defined by an execution order π and an execution speed. Following [63] it is assumed that every job i is scheduled at constant speed, and for the purpose of simplifying notation we rather specify the execution length ℓ_i of every job i, rather than its speed, which is w_i/ℓ_i. Two costs are associated with a schedule: the energy cost

$$E(\ell, w) := \sum_i w_i^\alpha \ell_i^{1-\alpha}$$

and the weighted flow time, representing the quality of service delivered to the users,

$$F(\pi, \ell, p) := \sum_j p_j C_j,$$

where C_j is the completion time of job j, defined as $\sum_{i:\pi(i)\leq\pi(j)} \ell_i$, with $\pi(i)$ being the rank of job i in the schedule.

In both case, the cost sharing mechanism defines the game completely.

3.2.1 Desirable properties

Ideally, we would like the game and the mechanism to have the following properties.

existence of pure Nash equilibria This means that there is a strategy profile vector such that no player can unilaterally deviate from their strategy while strictly decreasing their penalty.

budget balance The mechanism is γ-budged balanced, when the sum of the cost shares is no smaller than the total energy consumption and no larger than γ times the energy consumption.

truthful Every player minimizes his penalty by announcing his true value. This property applies to the penalty game only, since in the deadline game there is no notion of true private value. Clearly the operator can optimize the social cost only if the players announce the true penalty factors, otherwise the operator optimizes with false values, and we have no guarantee on the outcome. Note that if $\hat{p} = p$ then that strategy profile p is the unique pure Nash equilibrium.

3.3 Mechanism design for the deadline game

In the sequel we introduce and study two different cost sharing mechanisms, namely Proportional Cost Sharing where every player pays exactly the cost generated during the execution of his job, and Marginal Cost Sharing where every player pays the increase of energy cost generated by adding this player to the game.

3.3.1 Proportional cost sharing

The proportional cost sharing is the simplest budget balanced cost sharing scheme one can think of. Every player i is charged exactly the energy consumed during the execution of his job. Unfortunately this mechanism does not behave well as we show in Theorem 3.1.

Fact 3.1. *In a single player game, the player's penalty is minimized by the deadline*

$$r_1 + w_1(\alpha - 1)^{1/\alpha} p_1^{-1/\alpha}.$$

Proof. If player 1 chooses deadline $d_1 = r_1 + x$ then the schedule is active between time r_1 and $r_1 + x$ at speed w_1/x. Therefore his penalty is

$$p_1(r_1 + x) + x^{1-\alpha} w_1^{\alpha}.$$

Deriving this expression in x, and using the fact that the penalty is concave in t for any $x > 0$ and $\alpha > 0$, we have that the optimal x for the player will set to zero the derivative. This implies the claimed deadline. □

argument	value	applicable range
$d_1^{(1)} = (\alpha-1)^{1/\alpha}$	$g_1(d_2) = \alpha(\alpha-1)^{1/\alpha-1}$	$d_2 \geq 2(\alpha-1)^{1/\alpha}$
$d_1^{(2)} = \frac{d_2}{2}$	$g_2(d_2) = d_2/2 + (d_2/2)^{1-\alpha}$	$d_2 \leq 2(\alpha-1)^{1/\alpha}$
$d_1^{(3)} = 2\left(\frac{\alpha-1}{2}\right)^{1/\alpha}$	$g_3(d_2) = \alpha\left(\frac{\alpha-1}{2}\right)^{1/\alpha-1}$	$\left(\frac{\alpha-1}{2}\right)^{1/\alpha} \leq d_2 \leq 2\left(\frac{\alpha-1}{2}\right)^{1/\alpha}$
$d_1^{(4)} = d_2 + (\alpha-1)^{1/\alpha}$	$g_4(d_2) = d_2 + \alpha\,(\alpha-1)^{1/\alpha-1}$	$d_2 \leq (\alpha-1)^{1/\alpha-1}$

Table 3.1: The local minimum in the range of f corresponding to f_i is a function of α and d_2, which we denote by $d_1^{(i)}$. The value at such local minimum is again a function of α and d_2, which we denote by $g_i(d_2)$. These are only potential minima: they exist if and only if the condition given in the last column is satisfied.

If there are at least two players however, the game does not have nice properties as we show now.

Theorem 3.1. *The* Proportional Cost Sharing *does not always admit a pure Nash equilibrium.*

The proof consists of a very simple example: there are 2 identical players with identical jobs, say $w_1 = w_2 = 1$, $r_1 = r_2 = 0$ and $p_1 = p_2 = 1$. First we determine the best response of player 1 as a function of player 2, then we conclude that there is no pure Nash equilibrium.

Lemma 3.1. *Given the second player's choice d_2, the penalty of the first player as a function of his choice d_1 is given by*

$$f(d_1) = \begin{cases} f_1(d_1) = d_1 + d_1^{1-\alpha} & \text{if } d_1 \leq \frac{d_2}{2} \\ f_2(d_1) = d_1 + (\frac{d_2}{2})^{1-\alpha} & \text{if } \frac{d_2}{2} \leq d_1 \leq d_2 \\ f_3(d_1) = d_1 + (\frac{d_1}{2})^{1-\alpha} & \text{if } d_2 \leq d_1 \leq 2d_2 \\ f_4(d_1) = d_1 + (d_1 - d_2)^{1-\alpha} & \text{if } d_1 \geq 2d_2 \end{cases} \tag{3.1}$$

The local minima of $f(d_1)$ are summarized in Table 3.1, and the penalties corresponding to player 1 picking these minima are illustrated in Figure 3.1.

Proof. Formula (3.1) follows by a straightforward case inspection. Then, to find all the local minima of f, we first look at the behavior of each of f_i, finding their local minima in their respective intervals, and afterwards we inspect the border points of these intervals.

Range of f_1: The derivative of f_1 is

$$f_1'(d_1) = 1 - (\alpha-1)d_1^{-\alpha} \ ,$$

whose derivative in turn is positive for $\alpha > 1$. Therefore, f_1 has a local minimum at $d_1^{(1)}$ as specified. Since we require that this local minimum is within the range where f coincides with f_1, the necessary and sufficient condition is $d_1^{(1)} \leq \frac{d_2}{2}$.

Range of f_2: f_2 is an increasing function, and therefore it attains a minimum value only at the lower end of its range, $d_1^{(3)}$. However, if $d_1^{(2)}$ is to be a local minimum of f, there can be no local minimum of f in the range of f_1 (immediately to the left), so the applicable range of $d_1^{(2)}$ is the complement of that of $d_1^{(1)}$.

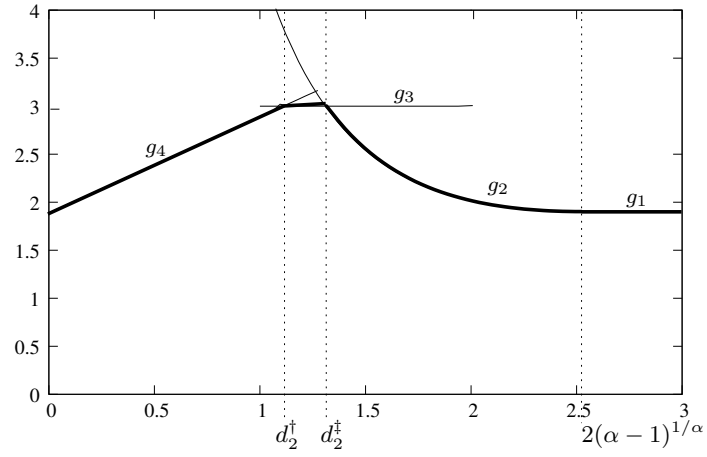


Figure 3.1: First player's penalty (in bold) when choosing his best response as a function of second player's strategy d_2, here for $\alpha = 3$.

Range of f_3: The derivative of f_3 is

$$f_3'(d_1) = 1 - \frac{\alpha - 1}{2}(d_1/2)^{-\alpha} \ , \tag{3.2}$$

whose derivative in turn is positive for $\alpha > 1$. Hence, f_3 has a local minimum at $d_1^{(3)}$ as specified. The existence of this local minimum requires $d_2 \leq d_1^{(3)} \leq 2d_2$, which is equivalent to $\frac{d_1^{(3)}}{2} \leq d_2 \leq d_1^{(3)}$.

Range of f_4: The derivative of f_4 is

$$f_4'(d_1) = 1 - (\alpha - 1)(d_1 - d_2)^{-\alpha} \ , \tag{3.3}$$

whose derivative in turn is positive for $\alpha > 1$. Hence, f_4 has a local minimum at $d_1^{(4)}$ as specified. The existence of this local minimum requires $d_1^{(4)} \geq 2d_2$.

Now let us consider the border points of the ranges of each f_i. Since f_2 is strictly increasing, the border point of the ranges of f_2 and f_3 is not a local minimum of f. This leaves only the border point $d_1^{(2)} = 2d_2$ of the ranges of f_3 and f_4 to consider. Clearly, $d_1^{(2)}$ is a local minimum of f if and only if $f_3'(d_1^{(2)}) \leq 0$ and $f_4'(d_1^{(2)}) \geq 0$. However, by (3.2), $f_3'(d_1^{(2)}) = 2 - (\alpha - 1)d_2^{-\alpha}$, and by (3.3), $f_4'(d_1^{(2)}) = 2 - 2(\alpha - 1)d_2^{-\alpha} < f_3'(d_1^{(2)})$, so $d_1^{(2)}$ is not a local minimum of f either. □

Note that the range of g_1 is disjoint with the ranges of g_3 and g_4, and with the exception of the shared border value $2(\alpha - 1)^{1/\alpha}$, also with the range of g_2. However, the ranges of g_2, g_3 and g_4 are not disjoint. Therefore, we now focus on their shared range, and determine which of the functions gives rise to the true local minimum (the proof is omitted due to space constraints).

Lemma 3.2. *The function $g_3(d_2)$ is constant, the function $g_4(d_2)$ is an increasing linear function, and the function $g_2(d_2)$ is decreasing for $d_2 < d_1^{(3)}$. Moreover, there exist two unique values*

$$d_2^\dagger = \alpha(\alpha-1)^{1/\alpha-1}(2^{1-1/\alpha}-1) \quad \textit{such that} \quad g_4(d_2^\dagger) = g_3(d_2^\dagger) \ , \tag{3.4}$$

$$d_2^\ddagger \in \left[d_2^\dagger,\ d_1^{(3)}\right) \quad \textit{such that} \quad g_2(d_2^\ddagger) = g_3(d_2^\ddagger) \ . \tag{3.5}$$

Proof. It follows from their definitions in Table 3.1 that g_3 is constant and g_4 strictly increasing, and $d_2^\dagger$ is defined as the unique root of $g_4(d_2) = g_3(d_2)$.

It also follows from Table 3.1 that g_2 is decreasing for $d_2 < 2\left(\frac{\alpha-1}{2}\right)^{1/\alpha}$ when the derivative of g_2 is inspected. Thus, to show that there is a root of $g_2(d_2) = g_3(d_2)$ in $\left(d_2^\dagger,\ d_1^{(3)}\right)$, it suffices to show the following

$$g_2\left(d_1^{(3)}\right) < \alpha\left(\frac{\alpha-1}{2}\right)^{1/\alpha-1} < g_2\left(d_2^\dagger\right) \ .$$

We start with the left inequality:

$$\begin{aligned} g_2\left(d_1^{(3)}\right) &= g_2\left(2\left(\frac{\alpha-1}{2}\right)^{1/\alpha}\right) = \left(\frac{\alpha-1}{2}\right)^{1/\alpha}\left(1+\frac{2}{\alpha-1}\right) \\ &= \left(\frac{\alpha-1}{2}\right)^{1/\alpha-1}\frac{\alpha+1}{2} < \left(\frac{\alpha-1}{2}\right)^{1/\alpha-1}\alpha \ . \end{aligned}$$

For the right inequality, we first note that

$$\begin{aligned} g_2\left(d_2^\dagger\right) &= d_2^\dagger/2 + (d_2^\dagger/2)^{1-\alpha} = d_2^\dagger(1/2 + (d_2^\dagger)^{-\alpha}2^{\alpha-1}) \\ &= \alpha(\alpha-1)^{1/\alpha-1}(2^{1-1/\alpha}-1)\left(\frac{2^{\alpha-1}}{(\alpha(\alpha-1)^{1/\alpha-1}(2^{1-1/\alpha}-1))^\alpha}+\frac{1}{2}\right) \ , \end{aligned}$$

hence $g_2\left(d_2^\dagger\right) > \alpha\left(\frac{\alpha-1}{2}\right)^{1/\alpha-1}$ is equivalent to

$$\begin{aligned} (2^{1-1/\alpha}-1)\left(\frac{2^{\alpha-1}}{\alpha^\alpha(\alpha-1)^{1-\alpha}(2^{1-1/\alpha}-1)^\alpha}+\frac{1}{2}\right) &\geq 2^{1-1/\alpha} \\ (2^{1-1/\alpha}-1)\left(\frac{2^{\alpha-1}}{\alpha\alpha^{\alpha-1}(\alpha-1)^{1-\alpha}(2^{1-1/\alpha}-1)^\alpha}-\frac{1}{2}\right) &\geq 1 \\ \frac{2^{\alpha-1}(\alpha-1)^{\alpha-1}\alpha^{1-\alpha}}{\alpha(2^{1-1/\alpha}-1)^\alpha}-\frac{1}{2} &\geq \frac{1}{2^{1-1/\alpha}-1} \\ \frac{1}{\alpha}\frac{\left(2-\frac{2}{\alpha}\right)^{\alpha-1}}{(2^{1-1/\alpha}-1)^\alpha} &\geq \frac{1}{2^{1-1/\alpha}-1}+\frac{1}{2} \\ \frac{1}{\alpha}\frac{\left(2-\frac{2}{\alpha}\right)^{\alpha-1}}{\left(\frac{2-2^{1/\alpha}}{2^{1/\alpha}}\right)^\alpha} &\geq \frac{2^{1-1/\alpha}+1}{2(2^{1-1/\alpha}-1)} \\ \frac{1}{\alpha}\frac{\left(2-\frac{2}{\alpha}\right)^{\alpha-1}}{(2-2^{1/\alpha})^{\alpha-1}(2-2^{1/\alpha})}2 &\geq \frac{(2+2^{1/\alpha})2^{-1/\alpha}}{2(2-2^{1/\alpha})2^{-1/\alpha}} \\ \frac{1}{\alpha}\left(\frac{2-\frac{2}{\alpha}}{2-2^{1/\alpha}}\right)^{\alpha-1} &\geq \frac{2+2^{1/\alpha}}{4}. \end{aligned}$$

We claim that for $\alpha \geq 2$ we have

$$\frac{2-\frac{2}{\alpha}}{2-2^{1/\alpha}} \geq \frac{1}{2-\sqrt{2}}. \tag{3.6}$$

Since we have equality at $\alpha = 2$, it suffices to prove that the left hand side is increasing. To this end, we consider its derivative

$$\frac{2\left(2-2^{\frac{1}{\alpha}}-\ln 2\cdot(1-\frac{1}{\alpha})\cdot 2^{\frac{1}{\alpha}}\right)}{\left(\alpha(2-2^{\frac{1}{\alpha}})\right)^2} = \frac{4-2^{1+\frac{1}{\alpha}}\cdot\left(1+(1-\frac{1}{\alpha})\ln 2\right)}{\left(\alpha(2-2^{\frac{1}{\alpha}})\right)^2} \ ,$$

and note that its enumerator is increasing with α and equals 0 for $\alpha = 1$. Thus (3.6) holds.

This permits us to define

$$z(\alpha) := (2-\sqrt{2})^{1-\alpha}/\alpha,$$

and to upper bound

$$\frac{1}{\alpha}\left(\frac{2-\frac{2}{\alpha}}{2-2^{1/\alpha}}\right)^{\alpha-1} \geq z(\alpha).$$

In order to lower bound

$$z(\alpha) \geq \frac{2+2^{1/\alpha}}{4}$$

for $\alpha \geq 2$, it suffices to show that z is increasing with α, since the right hand side is decreasing with α. Its derivative is

$$z'(\alpha) = -\frac{(2-\sqrt{2})^{1-\alpha}(1+\alpha\ln(2-\sqrt{2}))}{\alpha^2}.$$

Observe that $\alpha\ln(2-\sqrt{2}) < -1$ for every $\alpha \geq 2$, and therefore z' is positive and z is increasing as required. This concludes the proof of the lemma. □

With Lemma 3.1 and Lemma 3.2, whose statements are summarized in Table 3.1 and Figure 3.1, we can finally determine what is the best response of the first player as a function of d_2.

Lemma 3.3. *The best response for player 1 as function of d_2 is*

$$\begin{aligned}
d_1^{(4)} &= d_2 + (\alpha-1)^{1/\alpha} && \text{if} && 0 < d_2 \leq d_2^{\dagger} \ , \\
d_1^{(3)} &= 2\left(\frac{\alpha-1}{2}\right)^{1/\alpha} && \text{if} && d_2^{\dagger} < d_2 \leq d_2^{\ddagger} \ , \\
d_1^{(2)} &= \frac{d_2}{2} && \text{if} && d_2^{\ddagger} < d_2 \leq 2(\alpha-1)^{1/\alpha} \ , \\
d_1^{(1)} &= (\alpha-1)^{1/\alpha} && \text{if} && 2(\alpha-1)^{1/\alpha} < d_2 \ .
\end{aligned}$$

Proof. The proof consists in determining which of the applicable local minima of f is the global minimum for each range of d_2. Again, the cases are depicted in Figure 3.1.

case (i) $0 < d_2 \leq d_2^\dagger$**:** In this case, we claim that the best response of player 1 is

$$d_1^{(4)} = d_2 + (\alpha - 1)^{1/\alpha} \ .$$

First we prove that

$$d_2^\dagger \in \left(\left(\frac{\alpha - 1}{2} \right)^{1/\alpha}, \ (\alpha - 1)^{1/\alpha - 1} \right) \ .$$

The upper bound hold since

$$\begin{aligned} \alpha(\alpha - 1)^{1/\alpha - 1}(2^{1 - 1/\alpha} - 1) &< (\alpha - 1)^{1/\alpha - 1} \\ \alpha(2^{1 - 1/\alpha} - 1) &< 1, \end{aligned}$$

holds for $\alpha \geq 2$.

The lower bound holds since,

$$\begin{aligned} \alpha(\alpha - 1)^{1/\alpha - 1}(2^{1 - 1/\alpha} - 1) &> \left(\frac{\alpha - 1}{2} \right)^{1/\alpha} \\ \frac{\alpha}{\alpha - 1}(2^{1 - 1/\alpha} - 1) &> 2^{-1/\alpha} \\ \frac{\alpha}{\alpha - 1}(2 - 2^{1/\alpha}) &> 1 \end{aligned}$$

holds for $\alpha \geq 2$.

In fact, both inequalities are true even for $\alpha > 1$, but as we require $\alpha \geq 2$ due to Lemma 3.2, we settle for simpler proofs.

These bounds imply that in case (i) player 1 chooses the minimum among the 3 local minima $d_1^{(2)}$, $d_1^{(3)}$, and $d_1^{(4)}$, where the middle one is only an option for $\left(\frac{\alpha-1}{2}\right)^{1/\alpha} \leq d_2 \leq d_2^\dagger$. It follows from Lemma 3.2 that the last option always dominates: by (3.5), for every $\left(\frac{\alpha-1}{2}\right)^{1/\alpha} \leq d_2 < d_2^\ddagger$, we have $g_3(d_2) < g_2(d_2)$, and by (3.4), for every $\left(\frac{\alpha-1}{2}\right)^{1/\alpha} \leq d_2 \leq d_2^\dagger$, we have $g_4(d_2) < g_3(d_2)$. This concludes the analysis for case (i).

case (ii) $d_2^\dagger < d_2 \leq d_2^\ddagger$**:** In this case, we claim that the best response of player 1 is

$$d_1^{(3)} = 2 \left(\frac{\alpha - 1}{2} \right)^{1/\alpha} \ .$$

First we observe that by Lemma 3.2 (3.5),

$$d_2^\ddagger < d_1^{(4)} \ ,$$

which rules out $d_1^{(1)}$ as a choice for player 1, leaving only $d_1^{(2)}$, $d_1^{(3)}$, and $d_1^{(4)}$.

Again, Lemma 3.2 implies that $d_1^{(4)}$ dominates other choices: by (3.5), we have $g_3(d_2) < g_2(d_2)$ for all $\left(\frac{\alpha-1}{2}\right)^{1/\alpha} \leq d_2 < d_2^\ddagger$, and by (3.4), we have $g_3(d_2) < g_4(d_2)$ for all $d_2 > d_2^\dagger$. Note that for $\alpha = 2$, the range of this case is empty.

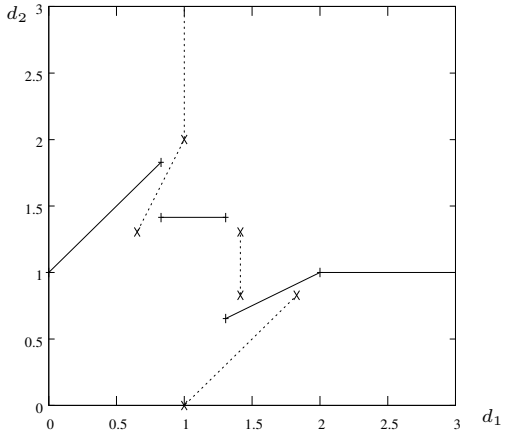


Figure 3.2: Best response of player 1 as function of d_2, and best response of player 2 as function of d_1. Here for $\alpha = 3$.

case (iii) $d_2^{\ddagger} < d_2 \leq 2(\alpha-1)^{1/\alpha}$**:** For this range, only $d_1^{(2)}$ and $d_1^{(3)}$ are viable choices for player 1, and Lemma 3.2 (3.5) implies that $d_1^{(2)}$ dominates. Therefore first player's best response is

$$d_1^{(2)} = \frac{d_2}{2} \ .$$

case (iv) $2(\alpha-1)^{1/\alpha} < d_2$**:** For this range, the only viable choice for player 1 is

$$d_1^{(1)} = (\alpha-1)^{1/\alpha} \ ,$$

which is therefore his best response.

This concludes the proof of the lemma. □

By the symmetry of the players, the second player's best response is in fact an identical function of d_1 as the one stated in Lemma 3.3. By straightforward inspection it follows that there is no fix point (d_1, d_2) to this game, which implies the following theorem, see figure 3.2 for illustration.

3.3.2 Marginal cost sharing

In this section we propose a different cost sharing scheme, that improves on the previous one in the sense that it admits pure Nash equilibria, however for the price of overcharging by at most a constant factor.

Before we give the formal definition we need to introduce some notations. Let $\mathrm{OPT}(d)$ be the energy minimizing schedule for the given instance, and $\mathrm{OPT}(d_{-i})$ be the energy minimizing schedule for the instance where job i is removed. We denote by $E(S)$ the energy cost of schedule S.

In the marginal cost sharing scheme, player i pays the penalty function

$$p_i d_i + E(\mathrm{OPT}(d)) - E(\mathrm{OPT}(d_{-i})).$$

This scheme defines an exact potential game by construction [47]. Formally, let n be the number of players, $D = \{d|\forall j : d_j > r_j\}$ be the set of action profiles (deadlines) over the action sets D_i of each player.

Let us denote the effective social cost corresponding to a strategy profile d by $\Phi(d)$. Then

$$\Phi(d) = \sum_{i=1}^{n} p_i d_i + E(\mathrm{OPT}(d)).$$

Clearly, if a player i changes its strategy d_i and his penalty decreases by some amount Δ, then the effective social cost decreases by the same amount Δ, because $E(\mathrm{OPT}(d_{-i}))$ remains unchanged.

3.3.2.1 Existence of Equilibria

While the best response function is not continuous in the strategy profile, precluding the use of Brouwer's fixed-point theorem, existence of pure Nash equilibria can nevertheless be easily established.

To this end, note that the global minimum of the effective social cost, if it exists, is a pure Nash equilibrium. Its existence follows from (1) compactness of a non-empty sub-space of strategies with bounded social cost and (2) continuity of Φ.

For (2), note that $\sum_i p_i d_i$ is clearly continuous in d, and hence $\Phi(d)$ is continuous if $E(\mathrm{OPT}(d))$ is. The continuity of the latter is clear once considers all possible relations of the deadlines chosen by the players.

For (1), let d' be any (feasible) strategy profile such that $d_i > r_i$ for each player i. The subspace of strategy profiles d such that $\Phi(d) \leq \Phi(d')$ is clearly closed, and bounded due to the $p_i d_i$ terms. Thus it is a compact subspace that contains the global minimum of Φ.

3.3.2.2 Convergence can take forever

In this game the strategy set is infinite. Moreover, the convergence time can be infinite as we demonstrate below in Theorem 3.2. Notice that this also proves that in general there are no dominant strategies in this game.

Theorem 3.2. *For the game with the marginal cost sharing mechanism, the convergence time to reach a pure Nash equilibrium can be unbounded.*

Proof. The proof is by exhibiting again the same small example, with 2 players, release times 0, unit weights, unit penalty factors, and $\alpha > 2$.

For this game there are two pure Nash equilibria, the first one is

$$d_1 = \left(\frac{\alpha - 1}{2}\right)^{1/\alpha}, \; d_2 = d_1 + (\alpha - 1)^{1/\alpha},$$

while the second one is symmetric for players 1 and 2.

In the reminder of the proof, we assume that player 1 chooses a deadline which is close to the pure Nash equilibrium above. By analyzing the best responses of the players, we conclude that after a best response of player 2, and then of player 1 again, he chooses a deadline which is even closer to the pure Nash equilibrium above but different from it, leading to an infinite convergence sequence of best responses. The proofs of the following two lemmas are omitted.

Now suppose $d_1 = \delta \left(\frac{\alpha-1}{2}\right)^{1/\alpha}$ for some $1 < \delta < 2^{1/\alpha}$. What is the best response for player 2?

Lemma 3.4. *Given the first player's choice d_1, the penalty of the second player as a function of his choice d_2 is given by*

$$h(d_2, d_1) = \begin{cases} h_1(d_2, d_1) = d_2 + d_2^{1-\alpha} + (d_1 - d_2)^{1-\alpha} - d_1^{1-\alpha} & \text{if } d_2 \leq \frac{d_1}{2} \\ h_2(d_2, d_1) = d_2 + (2^\alpha - 1)d_1^{1-\alpha} & \text{if } \frac{d_1}{2} \leq d_2 \leq d_1 \\ h_3(d_2, d_1) = d_2 + 2^\alpha d_2^{1-\alpha} - d_1^{1-\alpha} & \text{if } d_1 \leq d_2 \leq 2d_1 \\ h_4(d_2, d_1) = d_2 + (d_2 - d_1)^{1-\alpha} & \text{if } d_2 \geq 2d_1, \end{cases}$$

and the best response for player 2 as function of d_1 is

$$d_1 + (\alpha - 1)^{1/\alpha} = (\alpha - 1)^{1/\alpha}(1 + 2^{-1/\alpha}\delta) \tag{3.7}$$

Proof. We first analyze the behavior of each of h_i, finding their local minima in their respective intervals, and afterwards we show the dominance of (3.7). For convenience we omit parameter d_1 in each function h_i. Figure 3.3 illustrate the best response for player 2 when player 1 chooses $d_1 = (\frac{\alpha-1}{2})^{1/\alpha}\delta$ for some $\delta > 1$.

Range of h_4: The derivative of h_4 in d_2 is

$$1 - (\alpha - 1)(d_2 - d_1)^{-\alpha},$$

which is zero exactly for the value

$$d_1 + (\alpha - 1)^{1/\alpha}.$$

As the second derivative $h_1''(d_2) = \alpha(\alpha - 1)(d_2 - d_1)^{-\alpha-1}$ is positive, the choice $d_2 = d_1 + (\alpha - 1)^{1/\alpha}$ minimizes the penalty among $d_2 \geq 2d_1$. Therefore, h_1 has a local minimum if

$$(\alpha - 1)^{1/\alpha} \geq d_1 = \delta \left(\frac{\alpha - 1}{2}\right)^{1/\alpha},$$

which holds, since $\delta < 2^{1/\alpha}$

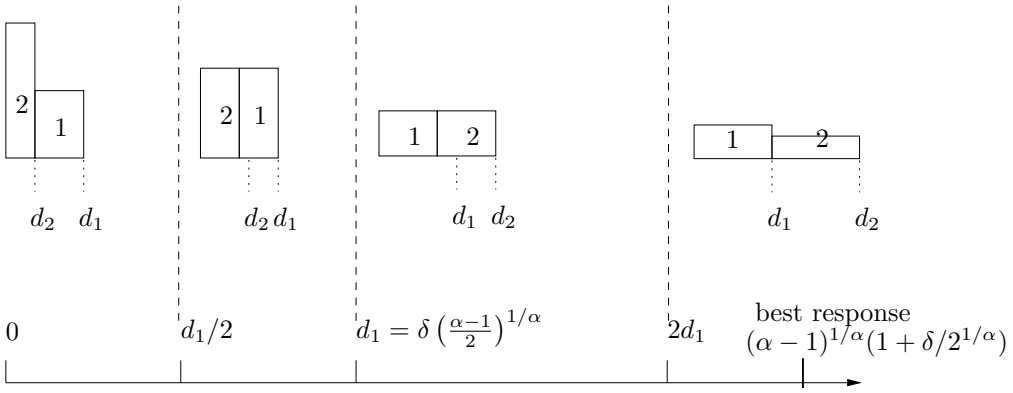


Figure 3.3: Best response for player 2.

In that case the penalty would be

$$d_1 + (\alpha-1)^{\frac{1-\alpha}{\alpha}} + (\alpha-1)^{1/\alpha} = (\alpha-1)^{1/\alpha}(1+\delta/2^{1/\alpha}+1/(\alpha-1)). \tag{3.8}$$

By comparing (3.8) with the remaining case, we show that it is indeed the best choice of player 2.

Range of h_3: The derivative of the h_3 in d_2 is

$$1 - 2^\alpha(\alpha-1)d_2^{-\alpha},$$

which is zero for $d_2 = 2(\alpha-1)^{1/\alpha}$ and negative for $d_2 < 2(\alpha-1)^{1/\alpha}$. As

$$2(\alpha-1)^{1/\alpha} > 2\delta\left(\frac{\alpha-1}{2}\right)^{1/\alpha} = 2d_1 \ ,$$

the minimum penalty in this range is attained at the right boundary of the interval, i.e., for $d_2 = 2d_1 = 2\delta\left(\frac{\alpha-1}{2}\right)^{1/\alpha}$. But as for $d_2 > 2d_1$ the function h, which is continuous, coincides with h_4, which is decreasing in $\left(2d_1, d_1+(\alpha-1)^{1/\alpha}\right)$, $2d_1$ is not a local minimum of h.

Range of h_2: h_2 is an increasing function, and therefore it attains a minimum value only at the lower end of its range, which is $d_1/2$ in this case. Clearly, $d_1/2$ is a local minimum of h if and only if $h_1'(d_1/2) \le 0$ and $h_2'(d_1/2) \ge 0$. However, we have $h_2'(d_1/2) = h_1'(d_1/2) = 1$, so $d_2 = d_1/2$ is not a local minimum of h either.

Range of h_1: We will show that h_1 is strictly larger than (3.8).

Since $\delta < 2^{1/\alpha}$, (3.8) is at most

$$(\alpha-1)^{1/\alpha}(2+1/(\alpha-1)) = (\alpha-1)^{1/\alpha-1}(2\alpha-1).$$

To lower bound h_1 we use the strict convexity of the function $x \mapsto x^{1-\alpha}$, which implies

$$2\left(\frac{d_2^{1-\alpha}}{2} + \frac{(d_1-d_2)^{1-\alpha}}{2}\right) > 2\left(\frac{d_1}{2}+\frac{d_1-d_2}{2}\right)^{1-\alpha} = 2^\alpha d_1^{1-\alpha}$$

Note that $d_1 < (\alpha-1)^{1/\alpha}$ implies $d_1^{1-\alpha} > (\alpha-1)^{1/\alpha-1}$. Combining these, we can finally strictly lower bound the difference h_1-(3.8) by

$$\begin{aligned} & d_2 + (2^\alpha-1)d_1^{1-\alpha} - (\alpha-1)^{1/\alpha-1}(2\alpha-1) \\ > \ & d_2 + (2^\alpha-1)(\alpha-1)^{1/\alpha-1} - (\alpha-1)^{1/\alpha-1}(2\alpha-1) \\ = \ & d_2 + (\alpha-1)^{1/\alpha-1}(2^\alpha-2\alpha), \end{aligned}$$

which is non-negative since $2^\alpha \ge 2\alpha$ whenever $\alpha \ge 2$.

□

From now on we assume that player 2 chooses $d_2 = d_1+(\alpha-1)^{1/\alpha} = (\alpha-1)^{1/\alpha}(1+2^{-1/\alpha}\delta)$. What is the best response for player 1?

Lemma 3.5. *Given the second player's choice d_2, the penalty of the first player as a function of his choice d_1 is given by $h(d_1, d_2)$ and the best response for player 1 is*

$$d_1 = \delta' \left(\frac{\alpha - 1}{2}\right)^{1/\alpha},$$

for some $\delta' \in (1, \delta)$.

Proof. Again player 1 best response is analyzed through a case analysis, similar to the previous one. Figure 3.4 illustrate the best response for player 1 when he chooses $d_2 = (\alpha - 1)^{1/\alpha}(1 + 2^{-1/\alpha}\delta)$ for some $\delta > 1$. As in the previous proof, for convenience we omit parameter d_2 in each function h_i.

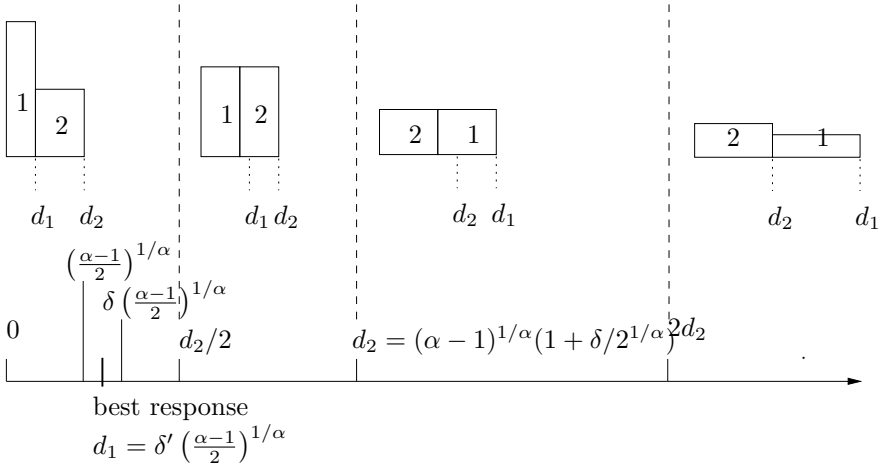


Figure 3.4: Best response for player 1.

Range of h_1: The first derivative of h_1 in d_1 is

$$h_1'(d_1) = 1 + (\alpha - 1)((d_2 - d_1)^{-\alpha} - d_1^{-\alpha})$$

And the second derivative is

$$h_1''(d_1) = \alpha(\alpha - 1)((d_2 - d_1)^{-\alpha-1} + d_1^{-\alpha-1})$$

which is positive, implying that the penalty is convex in d_1.

Now, we show that we have a local minimum for some $1 < \delta' < \delta$ at some value

$$\delta' \left(\frac{\alpha - 1}{2}\right)^{1/\alpha}.$$

For this purpose we analyze the interval

$$\left(\frac{\alpha - 1}{2}\right)^{1/\alpha} \leq d_1 \leq \delta \left(\frac{\alpha - 1}{2}\right)^{1/\alpha}.$$

First we evaluate h_1' at the lower end $d_1 = \left(\frac{\alpha-1}{2}\right)^{1/\alpha}$. In this case

$$d_2 - d_1 = \left(\frac{\alpha-1}{2}\right)^{1/\alpha} (2^{1/\alpha} + \delta - 1).$$

This means that

$$\begin{aligned} h_1'(d_1) &= 1 + (\alpha-1)\frac{2}{\alpha-1}(2^{1/\alpha} + \delta - 1)^{-\alpha} - (\alpha-1)\frac{2}{\alpha-1} \\ &= 2(2^{1/\alpha} + \delta - 1)^{-\alpha} - 1, \end{aligned}$$

which is negative for $\delta > 1$ and $\alpha > 1$.

Secondly, we evaluate the h_1' at the upper end

$$d_1 = \delta \left(\frac{\alpha-1}{2}\right)^{1/\alpha}, \tag{3.9}$$

then by $d_2 - d_1 = (\alpha-1)^{1/\alpha}$ we obtain

$$1 + (\alpha-1)\frac{1}{\alpha-1} - (\alpha-1)\frac{2}{\alpha-1}\delta^{-\alpha} = 2 - 2/\delta^{\alpha} \ ,$$

which is positive by $\delta < 2^{1/\alpha}$ and $\alpha > 1$. Together with the continuity of the penalty function, it implies that there is a value $1 < \delta' < \delta$ such that

$$d_1 = \delta' \left(\frac{\alpha-1}{2}\right)^{1/\alpha}$$

is a local minimum.

To conclude we compare this local minimum with the remaining cases, showing the dominance of this value.

Range of h_2: h_2 is an increasing function and therefore the minimum value is at $d_1 = d_2/2$. Again, $d_1/2$ is a local minimum of h if and only if $h_1'(d_2/2) \leq 0$ and $h_2'(d_2/2) \geq 0$. However, we have $h_2'(d_2/2) = h_1'(d_2/2) = 1$, so $d_1 = d_2/2$ is not a local minimum of h either.

Range of h_3: In this case, the derivative of penalty function is

$$1 - (\alpha-1)2^{\alpha} d_1^{-\alpha}$$

which is zero at

$$d_1 = 2(\alpha-1)^{1/\alpha}. \tag{3.10}$$

Note that the second derivative

$$h_3''(d_1) = \alpha(\alpha-1)2^{\alpha} d_1^{-\alpha-1}$$

is positive, so the penalty is convex, and (3.10) is a local minimum. It is greater than d_2 by $\delta < 2^{1/\alpha}$ and smaller than $2d_2$ by $\delta > 0$. Therefore the local minimum belongs to the range considered in this case.

The penalty at $d_1 = 2(\alpha-1)^{1/\alpha}$ is

$$\begin{aligned} & 2(\alpha-1)^{1/\alpha} + 2^{\alpha} \cdot 2^{1-\alpha}(\alpha-1)^{1/\alpha-1} - (\alpha-1)^{1/\alpha-1}(1+\delta/2^{1/\alpha})^{1-\alpha} \\ = \; & (\alpha-1)^{1/\alpha-1}\left(2\alpha - (1-\delta/2^{1/\alpha})^{1-\alpha}\right). \end{aligned}$$

We claim that this penalty is larger than h_1 evaluated at (3.9), eliminating it for a best choice option. For this purpose we need to show

$$\begin{aligned} h_1\left(\delta\left(\frac{\alpha-1}{2}\right)^{1/\alpha}\right) &< h_3(2(\alpha-1)^{1/\alpha}) \\ \frac{\delta(\alpha-1)}{2^{1/\alpha}} + \frac{2^{1-1/\alpha}}{\delta^{\alpha-1}} + 1 &< 2\alpha \\ \delta^{\alpha}(\alpha-1) + 2 &< (2\alpha-1)2^{1/\alpha}\delta^{\alpha-1}. \end{aligned}$$

This holds because

$$\begin{aligned} \delta^{\alpha-1}(2\alpha-1)2^{1/\alpha} - \delta^{\alpha}(\alpha-1) &= \delta^{\alpha-1}\left(2^{1/\alpha}(2\alpha-1) - \delta(\alpha-1)\right) \\ &> 2^{1/\alpha}(2\alpha-1) - \delta(\alpha-1) > 2^{1/\alpha}\alpha > 2 \;, \end{aligned}$$

for any $\alpha > 1$ because $2^{1/\alpha}\alpha$ increases for $\alpha > \ln 2$ and equals 2 for $\alpha = 1$.

Range of h_4: The derivative of h_4 in d_2 is

$$1 - (\alpha-1)(d_1-d_2)^{-\alpha},$$

which is zero exactly for the value

$$d_2 + (\alpha-1)^{1/\alpha}.$$

As the second derivative $h_1''(d_1) = \alpha(\alpha-1)(d_1-d_2)^{-\alpha-1}$ is positive, the choice $d_1 = d_2+(\alpha-1)^{1/\alpha}$ minimizes the penalty among $d_2 \geq 2d_1$. However, h_1 has a local minimum if

$$(\alpha-1)^{1/\alpha} \geq d_2 = (\alpha-1)^{1/\alpha}(1+2^{-1/\alpha}\delta),$$

which is a contradiction, since $\delta 2^{-1/\alpha} > 0$

□

This concludes the proof of Theorem 3.2. □

3.3.2.3 Bounding total charge

In this section we bound the total cost share for the Marginal Cost Sharing Scheme, by showing that it is at least $E(\mathrm{OPT}(d))$ and at most α times this value. In fact we show a stronger claim for individual cost shares.

Theorem 3.3. *For every player i, its marginal costshare is at least its proportional costshare and at most α times the proportional costshare.*

Proof. Fix a player i, and denote by S_{-i} the schedule obtained from OPT(d) when all executions of i are replaced by idle times. Clearly we have the following inequalities.

$$E(\text{OPT}(d_{-i})) \leq E(S_{-i}) \leq E(\text{OPT}(d))$$

Then the marginal cost share of player i can be lower bounded by

$$E(\text{OPT}(d)) - E(\text{OPT}(d_{-i})) \geq E(\text{OPT}(d)) - E(S_{-i}).$$

According to [63] the schedule OPT can be obtained by the following iterative procedure. Let S be the support of a partial schedule. For every interval $[t, t')$ we define its domain $I_{t,t'} := [t, t') \backslash S$, the set of included jobs $J_{t,t'} := \{j : [r_j, d_j) \subseteq [t, t')\}$, and the density $\sigma_{t,t'} := \sum_{j \in J_{t,t'}} w_j / |I_{t,t'}|$. The procedure starts with $S = \emptyset$, and while not all jobs are scheduled, it selects an interval $[t, t')$ with maximal density, and schedules all jobs from $J_{t,t'}$ in earliest deadline order in $I_{t,t'}$ at speed $\sigma_{t,t'}$ adding $I_{t,t'}$ to S.

For the upper bound, let $t_1 < t_2 < \ldots < t_\ell$ be the sequence of all release times and deadlines for some $\ell \leq 2n$. Clearly both schedules S run at uniform speed in every interval $[t_{k-1}, t_k)$. For every $1 \leq k \leq n$ let s_k be the speed of S in $[t_{k-1}, t_k)$, and s'_k the speed of S' in the same interval.

From the algorithm above it follows that every job is scheduled at constant speed, so let s_a be the speed at which job i is scheduled in OPT(d). It also follows that if $s_k > s_a$, then $s'_k = s_k$, and if $s_k \leq s_a$, then $s'_k \leq s_k$.

We establish the following upper bound.

$$\begin{aligned}
E(\text{OPT}(d)) - E(\text{OPT}(d_{-i})) &= \sum_{k=1}^{\ell} s_k^\alpha (t_k - t_{k-1}) - s_k'^\alpha (t_k - t_{k-1}) \\
&= \sum (t_k - t_{k-1})(s_k^\alpha - (s_k - (s_k - s'_k))^\alpha) \\
&= \sum (t_k - t_{k-1}) s_k^\alpha \left(1 - \left(1 - \frac{s_k - s'_k}{s_k}\right)^\alpha\right) \\
&\leq \sum (t_k - t_{k-1}) s_k^\alpha \left(1 - \left(1 - \alpha \frac{s_k - s'_k}{s_k}\right)\right) \\
&= \sum (t_k - t_{k-1}) \alpha s_k^{\alpha-1} (s_k - s'_k) \\
&\leq \alpha s_a^{\alpha-1} \sum (t_k - t_{k-1})(s_k - s'_k) \\
&= \alpha s_a^{\alpha-1} w_i \\
&= \alpha (E(\text{OPT}(d)) - E(S_{-i})).
\end{aligned}$$

The first inequality uses the generalized Bernoulli inequality, and the last one the fact that for all k with $s_k \neq s'_k$ we have $s_k \leq s_a$.

The theorem follows from the fact that $s_a^{\alpha-1} w_i$ is precisely the proportional cost share of job i in OPT(d). □

A tight example is given by n jobs, each with workload $1/n$, release time 0 and deadline 1. Clearly the optimal energy consumption is 1 for this instance. The marginal cost share for each player is $1 - (1 - 1/n)^\alpha$. Finally we observe that the total marginal cost share tends to α, i.e.

$$\lim_{n \to +\infty} n - n(1 - 1/n)^\alpha = \alpha.$$

3.3.3 A note on cross-monotonicity

We conclude by a short note on *cross-monotonicity*. This is a property of cost sharing games, stating that whenever new players enter the game, the cost share of any fixed player does not increase. This property is useful for stability in the game, and is the key to the Moulin carving algorithm [48], which selects a set of players to be served for specific games.

In the game that we consider, the minimum energy of an optimal schedule for a set S of jobs contrasts with many studied games, where serving more players becomes more cost effective, because the used equipment is better used.

Consider a very simple example of two identical players, submitting their respective jobs with the same deadline 1. Suppose the workload of each job is w, then the minimum energy necessary to schedule one job is w^α, while the cost to serve both jobs is $(2w)^\alpha$, meaning that the cost share increase whenever a second player enters the game. Therefore the marginal cost sharing scheme is not cross-monotonic.

3.4 Mechanism design for the penalty game

In this section we study the *penalty game*, where player announce with the submitted job penalty factors $\hat{p}$. Several problems arise in this situation.

First, the game operator knows only the announced penalty factors $\hat{p}$, so the game has to be *truthful*. This means that every player optimizes his cost by announcing the true value $\hat{p}_i = p_i$.

Second, no polynomial algorithm is known for finding a schedule π, ℓ that minimizes $E(\ell, w) + F(\pi, \ell, p)$ for arbitrary value α. In fact, it is also not known whether this problem is NP-hard. However, a PTAS is known [44]. Dominance properties for this problem have been established in [20] (see Chapter 5.)

We note that the solution boils down to finding the right order of scheduling, since once π is fixed, the optimum durations (speeds) for processing each job can be easily determined, cf. [44] or the Section 4.2

In order to define the mechanism, we need to introduce some notations. In this section we fix an arbitrary job order π. The actual mechanism could choose any order, for example uniformly at random. The important property is that the order does not depend on the players strategies, to cut any possible influence.

Let $E(OPT_\pi(\hat{p}))$ denote the value $E(\ell, w)$ with ℓ being the minimizer for $E(\ell, w) + F(\pi, \ell, \hat{p})$, which is the energy component of the social optimum. In addition we denote by $E(OPT_\pi(\hat{p}_{-i}))$ the similar value when player i is excluded from the game.

The operator defines a cost sharing scheme where player i pays the penalty function

$$b_i := \alpha \left(E(OPT_\pi(\hat{p})) - E(OPT_\pi(\hat{p}_{-i})) \right) - \hat{p}_i C_i.$$

We now analyze properties of the mechanism.

3.4.1 Truthfulness

In this section, we prove that the mechanism admits a unique Nash equilibrium, which is truthful. Formally, we claim that for every player i the strictly dominant strategy is $\hat{p}_i = p_i$.

To show the above claim, we will need the following technical lemmas.

Lemma 3.6. *Consider $\alpha > 1$, an order fixed π and an arbitrary player $i = \pi(j)$. For all $k \leq \pi(j)$, we have:*

$$\alpha \frac{\delta s_k^\alpha \ell_k}{\delta \hat{p}_i} = \ell_k \tag{3.11}$$

$$\frac{\delta \ell_k}{\delta \hat{p}_i} < 0. \tag{3.12}$$

Proof. Fix a permutation π, a value $\alpha > 1$ and an arbitrary player i. Without loss of generality, we assume that $i = \pi(i)$ for all $i = 1, \ldots, n$.

For every $k \leq i$, we have:

$$\begin{aligned} s_k^\alpha \ell_k = w_k^\alpha \ell_k^{1-\alpha} &= w_k^\alpha \left(\left(\frac{w_k^\alpha (\alpha - 1)}{\sum_{j=k}^n \hat{p}_j} \right)^{1/\alpha} \right)^{1-\alpha} \\ &= \frac{w_k}{(\alpha - 1)^{1-1/\alpha}} \left(\sum_{j=k}^n \hat{p}_j \right)^{1-1/\alpha} \end{aligned}$$

We now partial derive the above expression in $\hat{p}_i$ and have

$$\begin{aligned} \frac{\delta s_k^\alpha \ell_k}{\delta \hat{p}_i} = \frac{w_k}{(\alpha - 1)^{1-1/\alpha}} \frac{\delta \left(\sum_{j=k}^n \hat{p}_j \right)^{1-1/\alpha}}{\delta \hat{p}_i} &= \frac{w_k}{(\alpha - 1)^{1-1/\alpha}} \frac{\alpha - 1}{\alpha} \left(\sum_{j=k}^n \hat{p}_j \right)^{-1/\alpha} \\ &= \frac{w_k}{\alpha (\alpha - 1)^{-1/\alpha}} \left(\sum_{j=k}^n \hat{p}_j \right)^{-1/\alpha} \\ &= \frac{w_k}{\alpha} \left(\frac{\sum_{j=k}^n \hat{p}_j}{\alpha - 1} \right)^{-1/\alpha} \\ &= \frac{w_k}{\alpha} \left(\frac{\alpha - 1}{\sum_{j=k}^n \hat{p}_j} \right)^{1/\alpha} \\ &= \frac{\ell_k}{\alpha} \end{aligned}$$

and thus, we obtain

$$\alpha \frac{\delta s_k^\alpha \ell_k}{\delta \hat{p}_i} = \ell_k,$$

which concludes the proof of (3.11).

For (3.12), we have that for every $k \leq i$.

$$\frac{\delta \ell_k}{\delta \hat{p}_i} = w_k (\alpha - 1)^{1/\alpha} \frac{\delta \left(\sum_{j=k}^n \hat{p}_i \right)^{-1/\alpha}}{\delta \hat{p}_i} = \frac{-w_k (\alpha - 1)^{1/\alpha}}{\alpha} \left(\sum_{j=k}^n \hat{p}_j \right)^{-1/\alpha - 1} < 0,$$

which completes the proof of the lemma. □

We now show the main result.

Theorem 3.4. *The mechanism is truthful.*

Proof. We need to show that every player i minimizes his penalty when choosing the strategy $\hat{p}_i = p_i$. For the ease of notation, without loss of generality we assume $\pi(i) = i$. The total penalty of player i is:

$$\begin{aligned} & p_i C_i + \alpha \left(E(OPT_\pi(\hat{p})) - E(OPT_\pi(\hat{p}_{-i})) \right) - \hat{p}_i C_i \\ =& (p_i - \hat{p}_i) \sum_{k=1}^{i} \ell_k + \alpha \left(E(OPT_\pi(\hat{p})) - E(OPT_\pi(\hat{p}_{-i})) \right). \end{aligned}$$

We derive partially in p_i and have:

$$\begin{aligned} & -\sum_{k=1}^{i} \ell_k + (p_i - \hat{p}_i) \sum_{k=1}^{i} \frac{\delta \ell_k}{\delta \hat{p}_i} + \alpha \left(\sum_{k=1}^{i} \frac{\delta s_k^\alpha \ell_k}{\delta \hat{p}_i} \right) \\ = & -\sum_{k=1}^{i} \ell_k + (p_i - \hat{p}_i) \sum_{k=1}^{i} \frac{\delta \ell_k}{\delta \hat{p}_i} + \sum_{k=1}^{i} \ell_k \\ = & (p_i - \hat{p}_i) \sum_{k=1}^{i} \frac{\delta \ell_k}{\delta \hat{p}_i}. \end{aligned}$$

by (3.11). By (3.12), the value is zero and minimized at $\hat{p}_i = p_i$. □

3.4.2 Bounding total charge

In this section, we estimate the overcharging of total cost share when at least 2 players participate in the game.

Theorem 3.5. *The sum of the cost shares is at least $OPT(\hat{p})$ and at most $(\alpha + 1)OPT(\hat{p})$.*

Proof. Without loss of generality, we assume $i = \pi(i)$. To define a lower bound, we claim

that the share cost b_i is at least its cost energy consumption. We have

$$
\begin{aligned}
b_i =& \alpha \left(E(OPT_\pi(\hat{p})) - E(OPT_\pi(\hat{p}_{-i}))\right) - \hat{p}_i C_i \\
=& \alpha \left(\sum_{k=1}^{i} \ell_k s_k^\alpha - \sum_{k=1}^{i-1} w_k \left(\frac{\sum_{j=k}^{n} \hat{p}_j - \hat{p}_i}{\alpha - 1} \right)^{1-1/\alpha} \right) - \hat{p}_i \sum_{k=1}^{i} \ell_k \\
=& \alpha \ell_i s_i^\alpha - \ell_i \hat{p}_i + \sum_{k=1}^{i-1} \left(\alpha \ell_k s_k^\alpha - \ell_k \hat{p}_i - \alpha w_k \left(\frac{\sum_{j=k}^{n} \hat{p}_j - \hat{p}_i}{\alpha - 1} \right)^{1-1/\alpha} \right) \\
=& \alpha \ell_i s_i^\alpha - \ell_i \hat{p}_i + \sum_{k=1}^{i-1} \left(\alpha w_k \left(\frac{\sum_{j=k}^{n} \hat{p}_j}{\alpha - 1} \right)^{1-1/\alpha} - \alpha w_k \left(\frac{\sum_{j=k}^{n} \hat{p}_j - \hat{p}_i}{\alpha - 1} \right)^{1-1/\alpha} - \ell_k \hat{p}_i \right) \\
=& \alpha \ell_i s_i^\alpha - \ell_i \hat{p}_i + \sum_{k=1}^{i-1} \left(\frac{\alpha}{(\alpha-1)^{1-1/\alpha}} w_k \left(\left(\sum_{j=k}^{n} \hat{p}_j \right)^{1-1/\alpha} - \left(\sum_{j=k}^{n} \hat{p}_j - \hat{p}_i \right)^{1-1/\alpha} \right) - \ell_k \hat{p}_i \right) \\
>& \alpha \ell_i s_i^\alpha - \ell_i \hat{p}_i + \sum_{k=1}^{i-1} \left(\frac{\alpha}{(\alpha-1)^{1-1/\alpha}} w_k p_i \frac{\alpha - 1}{\alpha} \left(\sum_{j=k}^{n} \hat{p}_j \right)^{-1/\alpha} - \ell_k \hat{p}_i \right) \qquad (3.13) \\
=& \alpha \ell_i s_i^\alpha - \ell_i \hat{p}_i + \sum_{k=1}^{i-1} \left(\frac{w_k}{(\alpha-1)^{-1/\alpha}} \hat{p}_i \left(\sum_{j=k}^{n} \hat{p}_j \right)^{-1/\alpha} - \ell_k \hat{p}_i \right) \\
=& \alpha \ell_i s_i^\alpha - \ell_i \hat{p}_i + \sum_{k=1}^{i-1} \left(\ell_k \hat{p}_i - \ell_k \hat{p}_i \right) \\
=& \alpha \ell_i s_i^\alpha - \ell_i \hat{p}_i \\
=& \ell_i s_i^\alpha + (\alpha - 1) \ell_i s_i^\alpha - \ell_i \hat{p}_i \\
=& \ell_i s_i^\alpha + \ell_i \sum_{k=i}^{n} \hat{p}_k - \ell_i \hat{p}_i \\
=& \ell_i s_i^\alpha + \ell_i \left(\sum_{k=i+1}^{n} \hat{p}_k \right) \\
>& \ell_i s_i^\alpha
\end{aligned}
$$

The inequality (3.13) follows from the strict concavity of function $f : t \mapsto t^{1-1/\alpha}$ for $\alpha > 1$, which implies

$$f(t_1) - f(t_1 - a) = t_1^{1-1/\alpha} - (t_1 - a)^{1-1/\alpha} > a f'(t_1) = a(1 - 1/\alpha) t_1^{-1/\alpha}.$$

Thus, we have that the energy consumption of user i is $\ell_i s_i^\alpha$ and then, we have that the share cost b_i is at least its cost energy consumption, which concludes the first part of the proof.

For the upper bound, we denote by ℓ' the executing lengths vector which is the minimizer for $OPT_\pi(\hat{p}_{-i})$. For convenience we denote the corresponding speed $s'_k := w_k / \ell'_k$.

$$
\begin{aligned}
E(OPT_\pi(\hat{p})) - E(OPT_\pi(\hat{p}_{-i})) &= \sum_{k=1}^{n}(\ell_k s_k^\alpha - \ell'_k s'^\alpha_k) \\
&= \sum_{k=1}^{i-1}(\ell_k s_k^\alpha - \ell'_k s'^\alpha_k) + \ell_i s_i^\alpha \\
&= \sum_{k=1}^{i-1}\left(\ell_k \frac{\sum_{j=i}^{n}\hat{p}_j}{\alpha-1} - \ell'_k \frac{\sum_{j=i}^{n}\hat{p}_j - \hat{p}_i}{\alpha-1}\right) + \ell_i s_i^\alpha \\
&= \sum_{k=1}^{i-1}\left((\ell_k - \ell'_k)\frac{\sum_{j=i}^{n}\hat{p}_j - \hat{p}_i}{\alpha-1}\right) + \frac{\hat{p}_i}{\alpha-1}\sum_{k=1}^{i-1}\ell_k + \ell_i s_i^\alpha \\
&< \frac{\hat{p}_i}{\alpha-1}\sum_{k=1}^{i-1}\ell_k + \ell_i s_i^\alpha \\
&< \frac{\hat{p}_i}{\alpha-1}\sum_{k=1}^{i}\ell_k + \ell_i s_i^\alpha.
\end{aligned}
$$

The inequality follows from $\ell_k < \ell'_k$ for every $1 \leq k < i$.

We have the following bound on the cost share on player i,

$$
\ell_i s_i^\alpha \leq b_i \leq \alpha\left(\frac{\hat{p}_i}{\alpha-1}\sum_{k=1}^{i}\ell_k + \ell_i s_i^\alpha\right) - \hat{p}_i\sum_{k=1}^{i}\ell_k,
$$

which summed up over all players leads to

$$
\begin{aligned}
E(OPT_\pi(\hat{p})) \leq \sum_{i=1}^{n} b_i \leq& \alpha E(OPT_\pi(\hat{p})) + \frac{1}{\alpha-1}\sum_{i=1}^{n}\hat{p}_i\sum_{k=1}^{i}\ell_k \\
=& \alpha OPT_\pi(\hat{p}) + \frac{1}{\alpha-1}\sum_{i=1}^{n}\ell_i\sum_{k=i}^{n}\hat{p}_k \\
=& \alpha OPT_\pi(\hat{p}) + \frac{1}{\alpha-1}(\alpha-1)\sum_{i=1}^{n}\ell_i s_i^\alpha \\
=& \alpha OPT_\pi(\hat{p}) + OPT_\pi(\hat{p}) \\
=& (\alpha+1)OPT_\pi(\hat{p}),
\end{aligned}
$$

concluding the proof. □

3.4.3 Final remark

The standard quality measure of the outcome of a game, is the ratio between the social cost of the Nash equilibria and the optimal social cost. From Theorems 3.4 and 3.5 it follows that when the job order is fixed, this ratio is between 1 and $\alpha + 1$. However this constant upper bound does not hold when considering the optimal social cost for the best possible order, which might differ from the fixed order in the game.

This observation motivates future work: the study of a different game, where the game regulator organizes an auction over the rank positions of the schedule.

3.5 Our contribution at a glance

We show our main contributions for the decentralized approach in Table 3.2.

Desired properties	**Deadline game**		**Penalty game**
Mechanism	*Proportional*	*Marginal*	*Our mechanism*
Overcharging	1 (exact)	α	$\alpha + 1$
Existence of Pure Nash Equilibrium (PNE)	Existence is not guaranteed (even for 2 identical players)	Always	Always (unique for any fixed job order)
Truthfulness for p_i		Not needed	Yes
Computation of PNE		By the best response dynamics	By the regulator
Price de Stability (PoS)		α	$\alpha + 1$
Price de Anarchy (PoA)		Unbounded	Unbounded
Computation time		Infinite convergence time (even when PNE is unique)	constant

Table 3.2: our main contribution for the decentralized approach

Chapter 4
Optimizing social cost: a centralized approach

This chapter is based on the following articles.

- Scheduling under dynamic speed-scaling for minimizing weighted completion time and energy consumption, with C. Dürr and Ł. Jeż, submitted to a journal.
- On the complexity of the single machine scheduling problem minimizing total weighted delay penalty, submitted to a journal.

4.1 Introduction

In this chapter, we study the problem of optimization social cost from a centralized perspective.

Specifically, we consider a simplified computing system with a single shared machine using dynamic speed-scaling, meaning it can run at a variable continuous speed to influence the completion times of the jobs. Each job i has a workload w_i, a release and priority p_i, representing a quality of service coefficient. All jobs are available from time 0 on.

Here, a schedule is defined by an execution order π and an execution speed. Following [63] it is assumed that every job i is scheduled at constant speed, and for the purpose of simplifying notation we rather specify the execution length ℓ_i of every job i, rather than its speed, which is w_i/ℓ_i. Two costs are associated with a schedule: the energy cost

$$E(\ell, w) := \sum_i w_i^\alpha \ell_i^{1-\alpha}$$

defined for some fixed physical constant $2 \leq \alpha \leq 3$ [11], and the weighted flow time, representing the quality of service delivered to the users,

$$F(\pi, \ell, p) := \sum_j p_j C_j,$$

where C_j is the completion time of job j, defined as $\sum_{i:\pi(i)\leq\pi(j)} \ell_i$, with $\pi(i)$ being the rank of job i in the schedule.

The operator goal is to minimize the total cost $E(\ell, w)+F(\pi, \ell, p)$, which we call the *social cost*. When adding the two costs, we consider the conversion of energy and completion time into monetary values, and assume for simplification that the conversion factors are hidden in the penalty factors.

In setting game, this optimization problem consisting in minimizing the sum of the energy consumption and of the total weighted completion times, under the assumption that regulator knows the true penalty factors p of all players. Formally it can be stated as follows.

Problem A Given n jobs with workloads w and penalty factors p, find a schedule defined by π, ℓ which minimizes $A_{w,p}(\pi, \ell) = E(\ell, w) + F(\pi, \ell, p)$.

This problem is equivalent to another scheduling problem. Here the machine runs at uniform speed, and the role of the speed is encoded in the objective function. Note that the two values of the input w, p play opposite roles in both problems.

Problem B Given n jobs with priority weights w and processing times p, find a schedule defined by an order σ which minimizes $B_{w,p}(\sigma) = \sum w_j C_j^{(\alpha-1)/\alpha}$, where the completion time C_j is defined as $\sum_i p_i$ over all jobs i with $\sigma(i) \leq \sigma(j)$.

4.2 Reduction to a scheduling problem

In this section we show equivalence between the above two problems. Our proof uses the Hessian conditions. Independently, a similar reduction has been discovered in [44] using Karush-Kuhn-Tucker conditions, relating Problem B and a variant of Problem A, with the goal of minimizing total weighted completion time under a given energy budget.

Theorem 4.1. *Let π be a job order, and denote by σ its reverse, i.e. $\sigma(i) = n+1-\pi(i)$. Let ℓ be the execution length vector minimizing $A_{w,p}(\pi, \ell)$. Then $A_{w,p}(\pi, \ell) = \alpha(\alpha-1)^{(\alpha-1)/\alpha} \cdot B_{w,p}(\sigma)$.*

Proof. Fix permutations π, σ and vector ℓ with the required condition. Without loss of generality, we assume that $\pi(i) = i$ (and $\sigma(i) = n + 1 - i$). This can always be achieved by renaming the jobs, since the problems are independent on the actual job indices.

The objective value of problem A is

$$A_{p,w}(\pi, \ell) = \sum_{j=1}^{n} w_j^{\alpha} \ell_j^{1-\alpha} + \sum_{j=1}^{n} p_j \sum_{k=1}^{j} \ell_k. \tag{4.1}$$

For any job j, the value above must be a local minimum with respect to ℓ_j, meaning that its derivative is zero. In other words

$$(1-\alpha) w_j^{\alpha} \ell_j^{-\alpha} + \sum_{k=j}^{n} p_k = 0,$$

or equivalently

$$w_j^{\alpha} \ell_j^{-\alpha} = \frac{\sum_{k=j}^{n} p_k}{\alpha - 1} \quad , \tag{4.2}$$

$$\ell_j = w_j \cdot \left(\frac{(\alpha-1)}{\sum_{k=j}^{n} p_k} \right)^{\frac{1}{\alpha}} \quad . \tag{4.3}$$

This condition is sufficient, since the Hessian is positive definite. In fact, the first derivative in ℓ_j is independent of ℓ_i for any $i \neq j$, and so the Hessian of A has zero non-diagonal terms, whereas the second derivative of A in ℓ_j is

$$w_j^\alpha(\alpha-1)\alpha/\ell_j^{\alpha+1},$$

which is positive for positive ℓ_j and $\alpha > 1$. Thus, we have that the diagonal terms of the Hessian of A are positive, the Hessian is positive definite and then, A is minimized by a vector ℓ which sets to zero the first derivative for every j.

Now, substituting (4.2) for $w_j^\alpha \ell_j^{-\alpha}$ and then (4.3) for ℓ_j in (4.1) yields

$$\begin{aligned}
A_{p,w}(\pi,\ell) &= \sum_{j=1}^{n} \ell_j \left(\frac{\sum_{k=j}^{n} p_k}{\alpha-1} + \sum_{k=j}^{n} p_k \right) \\
&= \frac{\alpha}{\alpha-1} \sum_{j=1}^{n} \ell_j \sum_{k=j}^{n} p_k \\
&= \frac{\alpha}{\alpha-1} \sum_{j=1}^{n} \left(\frac{w_j^\alpha(\alpha-1)}{\sum_{k=j}^{n} p_k} \right)^{\frac{1}{\alpha}} \sum_{k=j}^{n} p_k \\
&= \alpha(\alpha-1)^{\frac{1-\alpha}{\alpha}} \sum_{j=1}^{n} w_j \left(\sum_{k=j}^{n} p_k \right)^{\frac{\alpha-1}{\alpha}} \\
&= \alpha(\alpha-1)^{\frac{1-\alpha}{\alpha}} \cdot B_{w,p}(\sigma) \ ,
\end{aligned}$$

since scheduling jobs with processing times p in order $n, n-1, \ldots, 1$ on a uniform speed machine, produces completion time

$$C_j = \sum_{k=j}^{n} p_k$$

for every job j. □

4.3 On complexity of the scheduling problem

Typically in a scheduling problem, we are given a number of jobs, each which has some processing time and some priority weight, which have to be scheduled on a single machine in order to minimize a particular objective function. In this chapter we consider objective functions of the form $\sum_j w_j f(C_j)$, where w_j is the priority weight of job j, C_j the completion time of job j and f is a problem specific penalty function. Extending the three-field notation [26] our problem could be denoted as $1||\sum w_j f(C_j)$.

Most of the penalty functions considered in literature are monotone increasing, which forces optimal schedules to be *left-shifted*, meaning that all executions happen without idle time between time 0 and the total job processing time. In this case the problem is said to have the *left-shifted property*. The study of different penalty functions has several motivations. The first one concerns machines with speed varying over time. If we denote by s the speed function, then after t_0 time units, the machine processed workload $\int_0^{t_0} s(t)dt$. Minimizing

the weighted sum of the completion times in this setting reduces to minimizing the value $\sum w_j f(C_j)$ for a constant speed machine, where f is the inverse of the integral of s. Varying speed can be the consequence of a learning effect, upgrades, or even varying available energy or tear and wear of the machine. The second motivation was mentioned in [44, 58], where the authors considered a particular dynamic voltage scheduling problem with the goal of optimizing a linear combination between total weighted completion time and overall energy consumption. It was shown that this problem reduces to constant speed scheduling problem with the objective function $\sum w_j C_j^\beta$ for some constant $0 < \beta < 1$.

In this survey we study penalty functions with several elementary properties. We consider convex and concave functions, increasing and decreasing functions, strictly monotone functions and non monotone functions. Among those classes we consider piecewise linear functions with two slopes for simplicity and restrict ourself to left-shifted schedules, even when the penalty functions are not monotone increasing.

Most of the previous work focused on convex functions. In this class of functions, NP-hardness proofs consist generally in reductions from PARTITIONING and involve instances, where all jobs j have the same Smith-ratio w_j/p_j, with the exception of a single job. Our contribution is to show that for any increasing concave penalty function these so-called *almost equal ratio instances* are easy, ruling out a trivial adaption of the NP-hardness proofs. In addition we model the scheduling problem for a particular concave penalty function as the minimization of *half-products*, inheriting therefore of pseudopolynomial time algorithms and FPTAS for this problem.

4.3.1 Literature Review

We list a few scheduling problems for several penalties functions, whose complexity has been studied in the literature.

Weighted completion time For the identity function $f : t \mapsto t$, the problem reduces essentially to sorting, as has been found out more than 60 years ago [54]. It is known by a simple exchange argument that a schedule is optimal if and only if it sequences jobs in order of non-increasing *Smith-ratio*, where the Smith ratio of job j is defined by w_j/p_j.

Weighted exponential completion time: The concave penalty function $f : t \mapsto 1 - e^{-rt}$ for a constant r is considered by Rothkopk [51] in the context of minimizing total flow time, with continuously discounted linear waiting costs. He reduces the problem to sorting using a similar exchange argument. Here a schedule is optimal if and only if it sequences jobs in order of non-increasing order of $\frac{rw_j e^{-rp_j}}{1-e^{-rp_j}}$. Clearly, the same exchange argument applies to the convex penalty function $f : t \mapsto e^{rt}$, where the a schedule is optimal if and only if it sequences jobs in order of non-increasing order of $\frac{rw_j e^{-rp_j}}{e^{-rp_j}-1}$.

Weighted quadratic completion time: The square penalty function $f : t \mapsto t^2$ attracted a lot of attention [30, 45, 53, 57] as a compromise between minimizing makespan and minimizing average weighted completion time. Chen and Liu [15] showed the unary NP-hardness of the problem for parallel machines by reduction from NUMERICAL THREE-DIMENSIONAL MATCHING PROBLEM. However the complexity status of the single machine problem still remains open for any non-linear monomial penalty function.

Weighted total tardiness: In [64] the penalty function $f : t \mapsto \max\{0, t-d\}$ was considered for a given common due date d. This corresponds to minimizing the total weighted tardiness, where *tardiness* is defined as the time by which a job completes late with respect to the due date. It was shown that the problem is binary NP-hard, by reduction from the BiPartition problem. The paper [42] provides a pseudopolynomial algorithm in time $O(n^2 d)$ and an $O(n^2)$-time algorithm for the special case, when all jobs have equal weights. A fully polynomial time approximation scheme (FPTAS) is proposed in [39], which runs in time $O((n^6 \log \sum_j w_j)/\varepsilon^3)$. The above result is improved in [37], with an FPTAS running in time $O(n^2/\varepsilon)$.

Weighted total tardiness-completion time: A related function to the previous one is the linear combination $f : t \mapsto t + \delta \max\{t - d, 0\}$ for some fixed constant $\delta > 0$. Binary NP-hardness has been shown by Megow and Verschae [44], in the case δ is part of the input. The same authors propose for a convex piecewise linear penalty function an algorithm in pseudopolynomial time $O(n^2 d^2)$ as well as an FPTAS with time complexity $O(n^3 \log d/\varepsilon^2)$ [60].

Weighted total earliness: A penalty function complementary to weighted total tardiness is $f : t \mapsto \max\{0, d - t\}$, which defines the objective of minimizing the total weighted earliness. Here *earliness* of a job is defined as the time by which a job completes early with respect to the due date d. Note that for this objective function the restriction to left-shifted schedules is important. In [14] a pseudo-polynomial algorithm was proposed with time complexity $O(n^2 \sum_j p_j)$ as well as an FPTAS with time complexity $O(n^4/\varepsilon)$. In addition a relation between the earliness and the tardiness penalties has been established in [18], which transfers the NP-hardness proof from [14] to this penalty function.

Weighted total earliness - tardiness: The combination of the two previous penalty functions is $f : t \mapsto |t-d|$, which corresponds to the goal of scheduling jobs as close as possible to a common due date. This problem is motivated by just-in-time manufacturing and presents two versions: *constrained* when $d < \sum_j p_j$ and *unconstrained* otherwise. Only the first case has the left-shifted property and is already NP-hard when all jobs have unit weights as has been shown in [27, 33]. A dynamic programming algorithm to solve the problem in time $O(n^2 d)$ is proposed in [33], while [27] admits an algorithm with time complexity $O(n \sum_j p_j)$. Finally, an FPTAS was proposed in [40], running in time $O(n^6/\varepsilon^3)$.

Mean squared deviation: The squared variant of the previous function is $f : t \mapsto (t-d)^2$, which defines a scheduling problem with the goal of minimizing the weighted mean squared deviation of completion times with respect to a common due date. Previous work considered solely the unweighted version. Here, we have three cases: *unrestricted* when $d \geq d^*$, *restricted* when $d^{**} < d < d^*$ and *tightly restricted* otherwise; where d^* and d^{**} are the minimum and maximum value of the average completion times over all possible sequences of an instance. The left-shifted property is only guaranteed for the *tightly restricted* case, which presents a dynamic programming algorithm proposed in [61], which runs in time $O(n \sum_j p_j)$. In an unified problem context, Mondal [46] proposes a dynamic programming algorithm with complexity $O(n^2(d + \max p_j))$, which was improved to $O(nd)$ by Srirangacharyulua and Srinivasanb [55]. A relation between

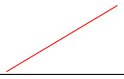
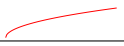
plot	function	complexity status	plot	function	complexity status
	t	$O(n \log n)$ [54]		e^{rt}	$O(n \log n)$ [51]
	$\max\{0, t-d\}$	Binary NP-hard [64] DP: $O(n^2 d)$ [42] FPTAS : $O((n^6 \log \sum_j w_j)/\varepsilon^3)$ [39], $O(n^2/\varepsilon)$ [37]		$1-e^{-rt}$	$O(n \log n)$ [51]
	$t+\max\{0, t-d\}$	DP:$O(n^2 d^2)$ [60] FPTAS: $O(n^3 \log d/\varepsilon^2)$ [60]		t^2	Open
	$\max\{0, d-t\}$	Binary NP-hard [14, 18] DP: $O(n^2 \sum_j p_j)$ [14] FPTAS :$O(n^4/\varepsilon)$ [14]		$\sqrt{t}$	Open
	$\min\{t, d\}$	DP: $O(nd)$ FPTAS:$O(n^2/\epsilon)$		$\lvert d-t \rvert$	Binary NP-hard [27, 33] DP: $O(n^2 d)$ [33], $O(n \sum_j p_j)$ [27] FPTAS: $O(n^6/\varepsilon^3)$ [40]
	piecewise linear	Unary NP-hard [32] PTAS: $2^{O(\log(1/\varepsilon)/\varepsilon^2)} \log(\sum_j w_j) n$ [44]		$(d-t)^2$	Results only for equals weight DP: $O(n \sum_j p_j)$ [61], $O(n^2(d+\max p_j))$ [46], $O(nd)$ [55]

Table 4.1: Results for several penalties functions

the problem of *unrestricted* case and the minimization of completion times variance has been established in [6], which transfers the NP-hardness proof from [41] to this problem. However, in the *tightly restricted* case, to the best of our knowledge no NP-hardness proof has been proposed.

Piecewise linear: Unary NP-hardness has been shown in [32] for a specific piecewise linear strictly increasing penalty function with two alternating slopes, which is therefore neither convex nor concave. For the general penalty function $f : t \mapsto \int_0^t s(x)dx$, where $s(x)$ is a given speed function, Megow and Verschae [44] proposed a PTAS, which runs in time $2^{O(\log(1/\varepsilon)/\varepsilon^2)} \log(\sum_j w_j) n$.

4.3.2 Piecewise linear concave penalty functions

We study a particular piecewise linear concave penalty function and relate it to the problem of minimizing a *half-product*. We refer to [5] for a detailed introduction to the later problem.

4.3.2.1 Half-product minimization

A half-product is a pseudo-Boolean function of the form:

$$F(x) = \sum_{j=1}^{n} a_j x_j \sum_{i=1}^{j-1} b_i x_i - \sum_{j=1}^{n} c_j x_j + D \tag{4.4}$$

where $x \in \{0,1\}^n$, $a = (a_1, ..., a_n), b = (b_1, ..., b_{n-1})$ are non-negative integer vectors, $c = (c_1 ..., c_n)$ is an arbitrary integer vector, and D an arbitrary integer. Denote by $x^* = (x_1^*, ..., x_n^*)$ a 0-1 vector minimizing F.

We know that for any quadratic pseudo-Boolean function of the form (4.4), and for any local minimum with respect to the flip of a single bit and in particular for the optimal x^*, we have for all $j = 1, .., n$

$$x_j^* \left(a_j \sum_{i=1}^{j-1} b_i x_i^* + b_j \sum_{i=j+1}^{n} a_i x_i^* - c_j \right) \leq 0. \tag{4.5}$$

Minimizing a half-product is NP-hard in the ordinary sense and admits two pseudo-polynomial algorithms, running in time $O(n \sum_j a_j)$ and $O(n \sum_j b_j)$ [23], respectively. These algorithms use the so called "state-space thinning" technique, which is a standard technique that has been applied to the SUBSET SUM problem [34] and the MINCLIQUE problem [36]. As for approximations, [5] present a scheme with time complexity $O(n \sum b_j/\varepsilon)$, which was improved to $O(n^2/\varepsilon)$ in [23].

Now we state our results for a particular piecewise linear concave penalty function.

Theorem 4.2. *The scheduling problem* $1||\sum_j w_j \min\{C_j, d\}$ *is polynomial time reducible to the problem of minimizing a half-product.*

Proof. For convenience we assume that the jobs are indexed such that $\frac{w_1}{p_1} \geq ... \geq \frac{w_n}{p_n}$. The problem generated by the penalty function $f : t \mapsto \min\{t, d\}$ reduces to minimizing (4.4), subject to

$$\sum_{i=1}^{j} b_i x_i \leq d \tag{4.6}$$

where $x_j \in \{0,1\}$, $a_j = w_j$, $b_j = p_j$, $c_j = (d - p_j)w_j$ and $D = d \sum_j w_j$ for all $j = 1, ..., n$. Denote by $x^* = (x_1^*, ..., x_n^*)$ a 0-1 vector minimizing F, such that $x_j^* = 1$ means the job j completes no later than deadline d.

We now show that the constraint (4.6) is satisfied by any solution minimizing (4.4), i.e. the constraint is not active for the objective function. From the optimality condition (4.5), we have:

$$x_j^* \left(w_j \sum_{i=1}^{j-1} p_i x_i^* + p_j \sum_{i=j+1}^{n} w_i x_i^* - w_j(d - p_j) \right) \leq 0.$$

In particular. For any $x_j^* = 1$, we have:

$$w_j \sum_{i=1}^{j} p_i x_i^* + p_j \sum_{i=j+1}^{n} w_i x_i^* - d w_j \leq 0$$

or equivalently

$$p_j \sum_{i=j+1}^{n} w_i x_i^* \leq w_j \left(d - \sum_{i=1}^{j} p_i x_i^* \right).$$

We choose k to be the largest index j with $x_j^* = 1$, implying $d \geq \sum_{i=1}^{k} p_i x_i^* = \sum_{i=1}^{n} p_i x_i^*$, since $p_k \sum_{i=k+1}^{n} w_i x_i^* \geq 0$. Therefore, the constraint (4.6) is satisfied by any optimal solution. □

From above theorem we immediately derive the following result.

Corollary 4.1.

The scheduling problem $1||\sum w_j \min\{C_j, d\}$ *admits an FPTAS in time* $O(n^2/\varepsilon)$ *and a pseudo-polynomial dynamic programming algorithm in time* $O(nd)$.

Proof. The first and second statement follow by the equivalence with half-product minimization. For the third statement, we adopt the pseudo-polynomial dynamic programming algorithm from [23] for the half-product minimization. It computes for every parameter t the best solution with $\sum x_j p_j = t$, and the claim follows from the observation that it suffices to range t from 0 to d. □

4.3.3 Almost equal Smith-ratio instances

Previous NP-hardness proofs for the scheduling problem with some convex penalty function involved almost equal ratio instances, i.e. instances where all but a single job have unit Smith-ratio. In this section we show that these instances are easy for any increasing concave penalty functions, ruling out a trivial adaption of the reductions used in the NP-hardness proofs.

Theorem 4.3. *Almost equal Smith-ratio instances are easy for any increasing concave penalty function.*

Proof. Fix an almost equal Smith-ratio instance, consisting of n jobs with $p_1 \leqslant \ldots \leqslant p_n$ and $w_i = p_i$ for all $i = 1, \ldots, n-1$. We show that without loss of generality the optimal schedule is of the form $1, 2, \ldots, k, n, k+1, k+2, \ldots, n-1$ for some $k = 0, \ldots, n-1$. This implies that the instance can be solved in time $O(n \log n)$, by first sorting the jobs and then evaluating the cost of each schedule of the form above, with a constant update time at each increment of k.

We claim that in an optimal schedule, without loss of generality for any two adjacent jobs i, j with $i, j \neq n$ we have that the smaller indexed job is scheduled first.

To show this claim, suppose that i, j are adjacent in an optimal schedule and let t be the largest completion time among the jobs. The claim states that the order i, j generates a cost not larger than the order j, i, i.e.

$$p_i f(t - p_j) + p_j f(t) \leqslant p_j f(t - p_i) + p_i f(t)$$

or equivalently

$$\frac{f(t) - f(t - p_j)}{p_j} \leqslant \frac{f(t - p_i) - f(t - p_j)}{p_j - p_i},$$

which holds by concavity of f.

To conclude the proof, we show that there is an optimal schedule such that there are no jobs $i < j < n$ where i is scheduled after j.

To this end, consider an optimal schedule, that minimizes the number of job pairs i, j with $i < j < n$ and i is scheduled after j. For a proof by contradiction suppose that this number is at least 1. By the previous claim all jobs before n are scheduled in order of their indices and so are all jobs scheduled after n. This means that job n is preceeded by a job j and followed by a job i with $i < j$.

Let t be the completion time of i. We denote by $F(S)$ the cost of the schedule, where jobs i, j, n follow a particular order described by the sequence S. By the choice of the schedule

$F(jni)$ is strictly upper bounded by $F(jin), F(nji)$ and $F(inj)$. By the claim we know that $F(ijn) \leqslant F(jin)$. Hence it is sufficient to show

$$F(jin) - F(jni) \leqslant F(ijn) - F(inj)$$

to obtain a contradiction. We develop the left hand side as

$$F(jin) - F(jni) = w_n(f(t) - f(t - p_i)) - p_i(f(t) - f(t - p_n)). \tag{4.7}$$

Since the left hand side is positive, the difference $f(t) - f(t - p_i)$ is non-zero

The claim also implies $F(nij) \leqslant F(nji)$ or equivalently

$$p_j \leqslant p_i \frac{f(t) - f(t - p_j)}{f(t) - f(t - p_i)}.$$

By monotonicity of f, the fraction above is at least 1. Multiplying the right hand side of (4.7) with this fraction, leads to

$$F(jin) - F(jni) \leqslant w_n(f(t) - f(t - p_j)) - p_j(f(t) - f(t - p_n)) = F(ijn) - F(inj).$$

□

4.3.4 NP-hardness

We were unable to show NP-hardness of the scheduling problem for some concave penalty function. For completeness we present a proof for the more general framework, where individual jobs have different penalty functions.

Theorem 4.4. *The scheduling problem* $1||\sum w_j \min\{C_j, d_j\}$ *is NP-hard.*

Proof. As usual we reduce from Subset Sum, which is NP-complete [25]. An instance to Subset Sum consists of positive integers $a_1, ..., a_n$ and B and the goal is to find binary values $x_1, \ldots, x_n$ such that $\sum_j a_j x_j = B$. Given an instance of Subset Sum, we define an instance to our problem as $p_j = w_j = 2a_j$, $d_j = 2B + a_j \leq 0$ for all $i = 1, ..., n$. A schedule is defined by a binary vector x, indicating for every job j whether it completes by its deadline d_j or not. Without loss of generality we suppose that $d_1 \leq \ldots \leq d_n$ and assume that early jobs are scheduled in order of indices while tardy jobs are also scheduled in order of indices. In this setting the objective value can be stated as

$$F(x) = \sum_{1 \leq i \leq j \leq n} w_j p_i x_j x_i + \sum_{j=1}^{n} d_j w_j (1 - x_j) \tag{4.8}$$

$$\text{subject to} \sum_{i=1}^{j} p_i x_i \leq d_j \qquad \forall j. \tag{4.9}$$

Now, we use the equivalence between the problem defined by the equation (4.8) and the constraints (4.9) and the problem of minimizing half-product $F(x)$. In fact, from the optimality condition (4.5), for all $x_j^* = 1$ we have

$$p_j \sum_{i=j+1}^{n} w_i x_i^* \leq w_j \left(d_j - \sum_{i=1}^{j} p_i x_i^* \right).$$

Since the left hand side is non-negative we obtain $\sum_{i=1}^{j} p_i x_i^* \leq d_j$. This means that (4.5) is satisfied for all indices j with $x_j^* = 1$, and by the assumption $d_1 \leq \ldots \leq d_n$, the condition is satisfied as well for all remaining indices.

To conclude the proof it suffices to show for any binary vector x, that the inequality $F(x) \leq \sum_{j=1}^{n}(4Ba_j + 2a_j^2) - 2B^2$ holds if and only if $\sum_j^n a_j x_j^* = B$. The following equalities show this claim, and use $x_j^2 = x_j$. We have

$$\begin{aligned}
F(x) =& 4 \sum_{1 \leq i \leq j \leq n} a_j a_i x_j x_i - \sum_{j=1}^{n} (x_j - 1)(4B + 2a_j)a_j \\
=& \sum_{j=1}^{n}(4Ba_j + 2a_j^2) + 2\left(\sum_{j=1}^{n} a_j x_j\right)^2 + 2\sum_{j=1}^{n} a_j^2 x_j^2 - \sum_{j=1}^{n} x_j(4Ba_j + 2a_j^2) \\
=& \sum_{j=1}^{n}(4Ba_j + 2a_j^2) + 2\left(\sum_{j=1}^{n} a_j x_j\right)^2 - 4B\sum_{j=1}^{n} a_j x_j \\
=& \sum_{j=1}^{n}(4Ba_j + 2a_j^2) + 2\left(\sum_{j=1}^{n} a_j x_j - B\right)^2 - 2B^2,
\end{aligned}$$

which means that if x minimizes $F(x)$ then $\sum a_j x_j = B$. □

Chapter 5
Order constraints

This chapter is based on the following articles.

- Order constraints for single machine scheduling with non-linear cost, with C. Dürr, in *Proceeding of 16th Workshop on Algorithm Engineering and Experiments (ALENEX)*, 2014.

- For the airplane refueling problem local precedence implies global precedence, submitted to a journal.

5.1 Introduction

In a scheduling problem the goal is to produce a schedule for the given jobs, that minimizes some objective function, usually depending on the job completion times, under problem specific constraints. One natural goal is to minimize the maximum completion time among all jobs, aka *makespan*. Another natural goal is to minimize the average completion time. A common way to combine these two objectives is to minimize the L_β-norm of the completion times, for some constant $\beta > 1$. As an example, the TEX-typesetting system uses the ℓ_3 metric to compute optimal line breaks.

In this chapter we consider the more general objective function $\sum_j w_j C_j^\beta$ where w_j is the given priority weight of job j, C_j its completion time and β some fixed positive constant. For convenience we introduce the function $f : t \mapsto t^\beta$, and call it the *penalty function*. We consider the most simple setting, where we are given n jobs, each job j has a processing time p_j and a priority weight w_j, and a schedule consists of an ordering of these jobs on a single machine. Here the completion time of job j is simply p_j plus the sum of the processing times over all jobs that are scheduled before j. There are several motivations to this scheduling problem. Some machines have a learning effect, and their efficiency is increasing during the execution, while some other machines have a wear and tear effect and their efficiency is decreasing during the execution. This can be expressed with values of β respectively smaller or greater than one. Moreover in [44, 58] a particular dynamic voltage scheduling problem optimizing quality of service and energy consumption is reduced to this same objective function for some $0 < \beta < 1$.

Embarrassingly very little is known about the computational complexity of this problem, except for the special case $\beta = 1$, where scheduling jobs in order of decreasing Smith ratio w_j/p_j leads to the optimal schedule, as has been found out 60 years ago [54]. Most research focused on the quadratic objective function, i.e. for $\beta = 2$, and exact algorithms have been proposed [7, 17, 45, 53, 57]

Two research directions were applied to this problem, approximation algorithms and branch and bound algorithms. Approximation algorithms have been proposed for the more general problem $1||\sum f_j(C_j)$, where every job j is given an increasing penalty function $f_j(t)$, that does not need to be of the form $w_j t^\beta$. Bansal and Pruhs [10] provided a constant ratio approximation algorithm based on a geometric interpretation. This factor has been improved from 16 to $2+\epsilon$ via a primal-dual approach by Cheung and Shmoys [16]. Epstein et al. [22] considered the problem $1||\sum w_j f(C_j)$, and provided a $4+\epsilon$ approximation algorithm for the setting where f is an arbitrary increasing differentiable penalty function chosen by the adversary *after* the schedule has been produced. A polynomial time approximation scheme has been provided by Megow and Verschae [44] for general monotone penalty functions f.

Finally, Höhn and Jacobs [32] derived a method to compute the tight approximation factor of the Smith-ratio-schedule for any particular monotone increasing convex or concave cost function. In particular for $f(t) = t^\beta$ they obtained for example the ratio 1.07 when $\beta = 0.5$ and the ratio 1.31 when $\beta = 2$.

For branch-and-bound algorithms pruning rules were proposed which reduce the number of nodes in the search graph, having a direct effect on the running time. For example, if we knew, that without loss of generality in an optimal schedule, job i is never scheduled after job j — which we denote by $i \prec_g j$ — then we could eliminate roughly half of the potential orderings, and reduce the number of explored nodes.

So an extensive literature was devoted to finding stronger rules, which are weaker conditions on the job characteristics that would still imply $i \prec_g j$. In this section we provide new rules and generalize existing ones to arbitrary values of $\beta > 0$. We conclude this paper with an experimental study of the impact of our proposed pruning rules to the performance of an exhaustive search procedure.

5.2 Pruning rules

Consider a schedule where i, j are scheduled one after the other, with job i starting at time t. Then the exchange of jobs i, j has some effect to the objective value of the schedule, which depends on jobs i, j and on time t. We denote by $i \prec_{\ell(t)} j$ the property that the order i, j generates a strictly smaller cost than the order j, i. If it holds for all time points $t \in [a, b]$, then we denote this property by $i \prec_{\ell[a,b]} j$, and if it holds everywhere we denote it simply by $i \prec_\ell j$.

The previously introduced properties generalize to situations where i and j are not scheduled adjacently. We say that i, j satisfy the *global order property* for interval $[a.b]$ if whenever in some schedule S, the completion times of jobs i, j satisfy

$$a \leq C_j - p_j \leq C_i - p_i - p_j \leq b,$$

then S is sub-optimal.[1]

Again if this property holds for all intervals, then we use simply the notation $i \prec_g j$.

We state the following conjecture, motivated by partial results and experiments.

Conjecture 5.1. *Fix an arbitrary constant $\beta > 0$. Then for all jobs i, j, $i \prec_\ell j$ implies $i \prec_g j$.*

[1] From the proof of Theorem 5.1 it will become clear why we require this lower bound on C_j.

A proof for this conjecture is given in this paper, for the case $\beta = 2$. Also during the experimental study, described in the last section of this paper, we successfully verified the conjecture on all the instances of our tests cases.

Interestingly the stronger implication $i \prec_{\ell[a,b]} j \Rightarrow i \prec_{g[a,b]} j$ does not hold. A counter example consists of the instance for $\beta = 2$ with $p_i = 13, w_i = 7, p_j = 8, w_j = 5, p_k = 1, w_k = 1$. For $t = 19/18$, we have $i \prec_{\ell[0,t)} j$ and $j \prec_{\ell(t,\infty)} i$. But the unique optimal solution is the sequence jki, meaning that we don't have $i \prec_{g[0,t)} j$. This contrasts with Theorem 5.1, which states that the implication holds whenever $p_i \leq p_j$.

5.3 Related work

Previous research focused mainly on the quadratic penalty function, i.e. $\beta = 2$. Branch-and-bound approaches with pruning rules implying order properties have been proposed, see [3,7,17,53,56,57]) In 2000, Mondal and Sen [45] conjectured that $\beta = 2, (w_i \geq w_j) \wedge (w_i/p_i > w_j/p_j)$ implies the global order property $i \prec_g j$, and provided experimental evidence that this constraint would significantly improve the runtime of a branch-and-prune search. Recently, Höhn and Jacobs [30] succeeded to prove this conjecture. In addition they improved local and global order conditions and generalized them to integer constants $\beta \geq 2$. An extensive experimental study analyzed the effect of these rules to the performance of the branch-and-prune search.

We distinguish the following known rules.

Sen-Dileepan-Ruparel [53] for any $\beta > 0$, if $w_i > w_j$ and $p_i \leq p_j$, then $i \prec_g j$.

Höhn-Jacobs-1 [30] for $\beta \in \mathbb{N}$, $\beta \geq 3$, if $w_i/p_i \geq \beta w_j/p_j$ then $i \prec_\ell j$.

Höhn-Jacobs-2 [30] for $\beta \in \mathbb{N}$, $\beta \geq 3$, if $w_i \geq w_j$ and $w_i/p_i > w_j/p_j$ then $i \prec_\ell j$.

Mondal-Sen-Höhn-Jacobs Conjectured in [45], proved in [30]. For $\beta = 2$ the two previous rules are enforced by the stronger implication $i \prec_g j$.

In this paper we characterize the condition $i \prec_\ell j$, and provide new sufficient conditions for the property $i \prec_g j$. For the special case $\beta = 2$ actually $i \prec_\ell j$ implies $i \prec_g j$.

5.4 Preliminaries

To simplify notation, throughout the paper we assume that no two jobs have the same processing time, weight or Smith-ratio (weight over processing time). For convenience we extend the notation of the penalty function f to the makespan of schedule S as $f(S) := f(\sum_{i \in S} p_i)$. Also we denote by $F(S)$ the cost of schedule S.

We define the following function on $t \geq 0$

$$\phi_{ij}(t) := \frac{f(t + p_i + p_j) - f(t + p_j)}{f(t + p_i + p_j) - f(t + p_i)},$$

and

$$\Delta_{ij}(t) := \; w_j f(t + p_j) + w_i f(t + p_i + p_j) - w_i f(t + p_i) - w_j f(t + p_i + p_j),$$

Note that $\phi_{ij}(t)$ is well defined since f is strictly increasing by assumption and the durations p_i, p_j are non-zero. It is this function ϕ_{ij} that permits us to analyze algebraically the local order property, since

$$i \prec_{\ell(t)} j \;\Leftrightarrow\; \phi_{ij}(t) < \frac{w_i}{w_j} \;\Leftrightarrow\; \Delta_{ij}(t) > 0.$$

The following technical lemmas show a connection between the properties of function $f : t \mapsto t^\beta$ and properties of function ϕ_{ij}, and show properties of f.

Lemma 5.1. *If $p_i \neq p_j$ then ϕ_{ij} is strictly monotone, in particular:*

- *If $p_i > p_j$ and $\beta > 1$, then ϕ_{ij} is strictly increasing.*
- *If $p_i < p_j$ and $\beta > 1$, then ϕ_{ij} is strictly decreasing.*
- *If $p_i > p_j$ and $\beta < 1$, then ϕ_{ij} is strictly decreasing.*
- *If $p_i < p_j$ and $\beta < 1$, then ϕ_{ij} is strictly increasing.*

Proof. See Figure 5.1 for an illustration of the claimed properties. Since $\phi_{ij}(t) = 1/\phi_{ji}(t)$, it suffices to consider the case $p_i > p_j$.

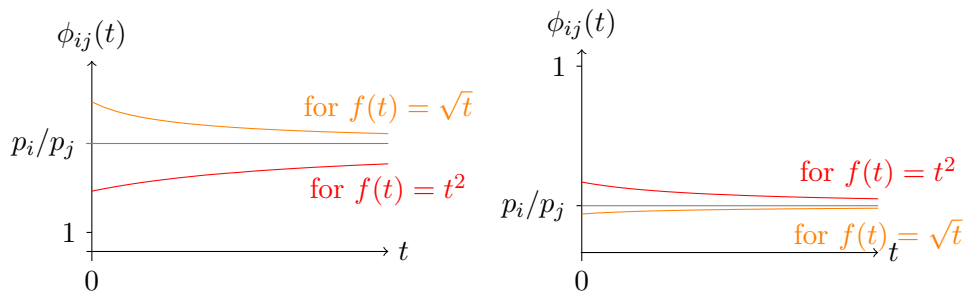


Figure 5.1: Examples of the function $\phi_{ij}(t)$ for $\beta = 0.5$ and $\beta = 2$, as well as for the cases $p_i > p_j$ and $p_i < p_j$.

Fix an arbitrary $t \geq 0$. For convenience let T be a fictive job of processing time t. We will show that

$$\phi'_{ij}(t) = \frac{f'(Tij) - f'(Tj)][f(Tij) - f(Ti)] - [f'(Tij) - f'(Ti)][f(Tij) - f(Tj)]}{[f(Tij) - f(Ti)]^2} > 0. \quad (5.1)$$

Since the denominator of this fraction is positive, we can focus on the numerator:

$$\begin{aligned}
& [f'(Tij) - f'(Tj)][f(Tij) - f(Ti)] - [f'(Tij) - f'(Ti)][f(Tij) - f(Tj)] \\
= \;& f'(Tij)f(Tij) - f'(Tj)f(Tij) - f'(Tij)f(Ti) + f'(Tj)f(Ti) \\
& - f'(Tij)f(Tij) + f'(Ti)f(Tij) + f'(Tij)f(Tj) - f'(Ti)f(Ti) \\
= \;& f'(Tij)f(Tj) - f'(Tj)f(Tij) + f'(Ti)f(Tij) \\
& - f'(Tij)f(Ti) + f'(Tj)f(Ti) - f'(Ti)f(Tj).
\end{aligned}$$

Up to factor β inequality (5.1) is equivalent to

$$\begin{aligned}
&(t+p_j+p_i)^{\beta-1}(t+p_j)^{\beta-1}((t+p_j)-(t+p_j+p_i))\\
+&(t+p_i+p_j)^{\beta-1}(t+p_i)^{\beta-1}((t+p_i+p_j)-(t+p_i))\\
+&(t+p_j)^{\beta-1}(t+p_i)^{\beta-1}((t+p_i)-(t+p_j))>0\\
(t+p_j+p_i)^{\beta-1}&(t+p_i)^{\beta-1}p_j+(t+p_j)^{\beta-1}(t+p_i)^{\beta-1}(p_i-p_j)>(t+p_i+p_j)^{\beta-1}(t+p_j)^{\beta-1}p_i && \Leftrightarrow\\
(p_j/p_i)&(1/(t+p_j))^{\beta-1}+(1-p_j/p_i)(1/(t+p_i+p_j))^{\beta-1}>(1/(t+p_i))^{\beta-1} && \Leftrightarrow
\end{aligned}$$

Using a function $h : x \mapsto (1/x)^{\beta-1}$ we reformulate this inequality as

$$(p_j/p_i)h(t+p_j)+(1-p_j/p_i)h(t+p_i+p_j)>h(t+p_i).$$

Note that $h(x)$ is a strictly convex function for positive x and $\beta > 1$, which implies

$$\lambda h(x_1)+(1-\lambda)h(x_2)>h(\lambda x_1+(1-\lambda)x_2)$$

for any $0 < \lambda < 1$ and $x_1, x_2 > 0$.

We choose $\lambda = p_j/p_i$, $x_1 = t+p_j, x_2 = t+p_i+p_j$ and obtain

$$\begin{aligned}
\lambda x_1+(1-\lambda)x_2 &= (p_j/p_i)(t+p_j)+(1-p_j/p_i)(t+p_i+p_j)\\
&= (p_j/p_i)(t+p_j)+t+p_j+p_i-(p_j/p_i)(t+p_j)-p_j\\
&= t+p_i.
\end{aligned}$$

In summary we obtain the inequalities

$$\begin{aligned}
(p_j/p_i)h(t+p_j)+(1-p_j/p_i)h(t+p_i+p_j) &> h((p_j/p_i)(t+p_j)+(1-p_j/p_i)(t+p_i+p_j))\\
&= h(t+p_i),
\end{aligned}$$

completing the proof. □

Lemma 5.2. *For any jobs i, j, we have*

$$\lim_{t\to\infty}\phi_{ij}(t)=p_i/p_j.$$

Proof. We prove this claim using the generalized binomial theorem. Here $\binom{\beta}{k}$ is defined for any real positive valued β as $\beta(\beta-1)\cdots(\beta-k+1)/k!$. By definition its value is zero iff β is integral and $0 \le \beta \le k-1$. Then we have

$$(t+p_i+p_j)^{\beta}=(t+p_j)^{\beta}+\beta(t+p_j)^{\beta-1}p_i+\sum_{k=2}^{\infty}\binom{\beta}{k}(t+p_j)^{\beta-k}(p_i)^k. \tag{5.2}$$

Therefore the limit can be expressed as (where the sum ranges for k from 2 to infinity)

$$\begin{aligned}
\lim_{t\to\infty}\phi_{ij}(t) &= \lim_{t\to\infty}\frac{(t+p_i+p_j)^\beta-(t+p_j)^\beta}{(t+p_i+p_j)^\beta-(t+p_i)^\beta}\\
&= \lim_{t\to\infty}\frac{(t+p_j)^\beta+\beta(t+p_j)^{\beta-1}p_i+\sum\binom{\beta}{k}(t+p_j)^{\beta-k}(p_i)^k-(t+p_j)^\beta}{(t+p_i)^\beta+\beta(t+p_i)^{\beta-1}p_j+\sum\binom{\beta}{k}(t+p_i)^{\beta-k}(p_j)^k-(t+p_i)^\beta}\\
&= \lim_{t\to\infty}\frac{\beta(t+p_j)^{\beta-1}p_i+\sum\binom{\beta}{k}(t+p_j)^{\beta-k}(p_i)^k}{\beta(t+p_i)^{\beta-1}p_j+\sum\binom{\beta}{k}(t+p_i)^{\beta-k}(p_j)^k}\\
&= \lim_{t\to\infty}\frac{\beta(1+p_j/t)^{\beta-1}p_i+\sum\binom{\beta}{k}\frac{1}{t^{k-1}}(1+p_j/t)^{\beta-k}(p_i)^k}{\beta(1+p_i/t)^{\beta-1}p_j+\sum\binom{\beta}{k}\frac{1}{t^{k-1}}(1+p_i/t)^{\beta-k}(p_j)^k}\\
&= \frac{p_i}{p_j}.
\end{aligned}$$

□

Lemma 5.3. *For $a<b$ and $p_i>p_j$,*

$$\frac{p_i}{p_j}\cdot\frac{f(b+p_i)-f(a+p_i)}{f(b+p_j)-f(a+p_j)}\geq 1.$$

Proof. When f is convex, the second fraction is clearly greater than 1. So we focus on the concave case. For this purpose we define the function

$$g(x) := x(f(b+x)-f(a+x))$$

and show that g is increasing, implying $g(p_i)/g(p_j)\geq 1$ as required. So we have to show $g'(x)>0$ in other words

$$f(b+x)-f(a+x)+x(f'(b+x)-f'(a+x))\geq 0$$

or

$$(b+x)^\beta+x\beta(b+x)^{\beta-1}\geq(a+x)^\beta+x\beta(a+x)^{\beta-1}.$$

To establish the last inequality, we introduce another function

$$r(z) := (z+x)^\beta+x\beta(z+x)^{\beta-1}$$

and show that r is increasing, implying $r(b)\geq r(a)$. By analyzing its derivative we obtain

$$\begin{aligned}
r'(z) &= \beta(z+x)^{\beta-1}+x\beta(\beta-1)(z+x)^{\beta-2}\\
&= (z+x)^{\beta-2}(\beta(z+x)+x\beta(\beta-1))\\
&= (z+x)^{\beta-2}(\beta z+x\beta^2),
\end{aligned}$$

which is positive as required. This concludes the proof. □

Lemma 5.4. *For $t\geq 0$ let the function q be defined as*

$$q(t) := \frac{f(t+p_j)-f(t)}{f(t+p_i)-f(t)} = \frac{(t+p_j)^\beta-t^\beta}{(t+p_i)^\beta-t^\beta}.$$

For $p_i>p_j$, if $\beta>1$ then q is increasing and if $0<\beta<1$ then q is decreasing.

Proof. We show only the convex case, the concave case is analogous. First, we compute the first derivative of q

$$q'(t) = \beta \frac{((t+p_i)^\beta - t^\beta)((t+p_j)^{\beta-1} - t^{\beta-1}) - ((t+p_j)^\beta - t^\beta)((t+p_i)^{\beta-1} - t^{\beta-1})}{((t+p_i)^\beta - t^\beta)^2}$$

We now show that q' is strictly increasing. Since the denominator of this fraction is positive, we can focus on the numerator. Up to factor β this is equivalent to:

$$\begin{aligned}&(t+p_i)^\beta(t+p_j)^{\beta-1} - (t+p_i)^\beta t^{\beta-1} - t^\beta(t+p_j)^{\beta-1} + t^\beta t^{\beta-1}\\ &-(t+p_j)^\beta(t+p_i)^{\beta-1} + (t+p_j)^\beta t^{\beta-1} + t^\beta(t+p_i)^{\beta-1} - t^\beta t^{\beta-1}.\end{aligned} \tag{5.3}$$

We use the transformation

$$\begin{aligned}&(t+p_i)^\beta(t+p_j)^{\beta-1} - (t+p_j)^\beta(t+p_i)^{\beta-1}\\ =&(t+p_i)(t+p_i)^{\beta-1}(t+p_j)^{\beta-1} - (t+p_j)(t+p_j)^{\beta-1}(t+p_i)^{\beta-1}\\ =&(t+p_i)^{\beta-1}(t+p_j)^{\beta-1}(p_i-p_j)\end{aligned}$$

to transform (5.3) into

$$(t+p_i)^{\beta-1}(t+p_j)^{\beta-1}(p_i-p_j) - (t+p_i)^{\beta-1}t^{\beta-1}p_i + (t+p_j)^{\beta-1}t^{\beta-1}p_j$$

which is positive if and only if

$$(p_j/p_i)(1/(t+p_i))^{\beta-1} + (1-p_j/p_i)(1/t)^{\beta-1} > (1/(t+p_j))^{\beta-1}.$$

Using function $h : x \mapsto (1/x)^{\beta-1}$ we reformulate this inequality as

$$(p_j/p_i)h(t+p_i) + (1-p_j/p_i)h(t) > h(t+p_j).$$

Note that $h(x)$ is a strictly convex function for positive x and $\beta > 1$, which implies

$$\lambda h(x_1) + (1-\lambda)h(x_2) > h(\lambda x_1 + (1-\lambda)x_2)$$

for any $0 < \lambda < 1$ and $x_1, x_2 > 0$.

We choose $\lambda = p_j/p_i$, $x_1 = t + p_i, x_2 = t$ and obtain

$$\begin{aligned}\lambda x_1 + (1-\lambda)x_2 &= (p_j/p_i)(t+p_i) + (1-p_j/p_i)(t)\\ &= tp_j/p_i + p_j + t - tp_j/p_i\\ &= t + p_j.\end{aligned}$$

In summary we obtain the required inequality

$$\begin{aligned}(p_j/p_i)h(t+p_i) + (1-p_j/p_i)h(t) >& h((p_j/p_i)(t+p_i) + (1-p_j/p_i)t)\\ =& h(t+p_j).\end{aligned}$$

This concludes the proof. □

5.5 Main Results

In [53] it has been shown that if job j has both longer processing time than i *and* smaller weight than i, then $i \prec_g j$, whereas in [30, 45] some conditions on jobs i, j imply the global order property were given for the quadratic penalty function. We enforce these statements using properties of $\phi_{ij}(t)$.

Theorem 5.1 (Rule 1)**.** *Let f be an arbitrary strictly increasing penalty function. Fix two jobs i, j, an interval $[a, b]$ and suppose $i \prec_{\ell[a,b]} j$. If $p_i \leq p_j$ then $i \prec_{g[a,b]} j$.*

Proof. Suppose that $p_i \leq p_j$. Let I be an instance containing jobs i, j and S a schedule on I of the form

$$S = AjBiD,$$

for some job sequences A, B, D. Let a be the total processing time of A and b the total processing time of AB. Then we have $a = C_j - p_j \leq C_i - p_i - p_j = b$ where C_i, C_j are the respective completion times in S. We show that exchanging the jobs i and j decreases the cost of the schedule. In particular we show the following inequality, where we dropped the suffix D of the schedules, since those jobs cancel in the difference anyway,

$$\max_{t\in[a,b]} \phi_{ij}(t) \left[F(AijB) - F(AiBj)\right] > F(AjiB) - F(AjBi). \tag{5.4}$$

This inequality would conclude the proof, for the following reason. First we claim

$$\max_{t\in[a,b]} \phi_{ij}(t) \leq 1,$$

which holds by definition of f in case $p_i \leq p_j$. This implies the stronger inequality

$$F(AijB) - F(AiBj) > F(AjiB) - F(AjBi),$$

or equivalently

$$F(AjBi) - F(AiBj) > F(AjiB) - F(AijB). \tag{5.5}$$

Since $i \prec_{\ell(a)} j$ implies that the right hand side is non-negative, this would conclude the proof.

In order to show inequality (5.4), for every job $k \in B$ we denote by t_k the completion time of k in the schedule $ABij$. In (5.4) we distinguish the contributions of jobs i, j and all jobs $k \in B$. In particular for every job k in B we have

$$\max_{t\in[a,b]} \phi_{ij}(t) w_k \left[f(t_k + p_i + p_j) - f(t_k + p_i)\right] \geq w_k \left[f(t_k + p_i + p_j) - f(t_k + p_j)\right] \tag{5.6}$$

since

$$\phi_{ij}(t_k) w_k \left[f(t_k + p_i + p_j) - f(t_k + p_i)\right] = w_k \left[f(t_k + p_i + p_j) - f(t_k + p_j)\right].$$

We denote by p_B the total processing time over all jobs in B. Then since $f(C_i - p_B) < f(C_i)$ and since $\max_{t\in[a,b]} \phi_{ij}(t) < w_i/w_j$ we have

$$\max_{t\in[a,b]} \phi_{ij}(t) w_j \left[f(C_i - p_B) - f(C_i)\right] > \frac{w_i}{w_j} \cdot w_j \left[f(C_i - p_B) - f(C_i)\right]. \tag{5.7}$$

Adding (5.6) and (5.7) establishes the required inequality (5.4). □

The following statement permits us to enumerate some conditions on job pairs and their Smith-ratios, which imply either a local or a global order property.

Theorem 5.2 (Rule 2)**.** *If f is an increasing strictly convex function, $w_i/p_i > w_j/p_j$ and $w_i > w_j$, then $i \prec_\ell j$. On the other hand if f is an increasing strictly concave function, $w_i/p_i > w_j/p_j$ and $w_i < w_j$, then $i \prec_g j$.*

Proof. First, we consider the case of a convex function f.

For $0 < \lambda < 1$, we have

$$(1-\lambda)f(x_1) + \lambda f(x_2) \geq f((1-\lambda)x_1 + \lambda x_2).$$

We choose $\lambda = w_j/w_i, x_1 = t + p_i + p_j, x_2 = t + p_j$ and have:

$$\begin{aligned}(1 - w_j/w_i)f(t + p_i + p_j) + w_j/w_i f(t + p_j) \geq & f(t + p_i + p_j - w_j p_i/w_i) \\ > & f(t + p_i).\end{aligned}$$

The latter inequality holds by case assumption on Smith-ratios of jobs i and j, and the strict monotonicity of f. Thus, we have:

$$(1 - w_j/w_i)f(t + p_i + p_j) + w_j/w_i f(t + p_j) - f(t + p_i) = \Delta_{ij}(t)/w_i > 0,$$

which implies $i \prec_\ell j$.

For the concave case, we have the implication $i \prec_\ell j$ by a similar argument. To prove the implication global, we observe that $w_i < w_j$ follows from the case assumption, and applying Theorem 5.1 permits to conclude $i \prec_g j$. □

Finally, we refine the above statements for the function of the form $f(t) : t \mapsto t^\beta$, with $\beta \in \mathbb{R}^+$.

Theorem 5.3 (Rule 3)**.** *Let i, j be two jobs with $p_j < p_i$. If*

$$\begin{aligned}\beta > 1 \quad \textit{and} \quad & \frac{w_i}{w_j} \geq \left(\frac{p_i}{p_j}\right)^\beta \\ \textit{or } 0 < \beta < 1 \quad \textit{and} \quad & \frac{w_i}{w_j} \geq \left(\frac{p_i}{p_j}\right)^2\end{aligned}$$

then

$$i \prec_g j.$$

Proof. Let i, j be two jobs with the required properties. Let A, B be two arbitrary job sequences. We will show that the schedule $AjBi$ is suboptimal, thus showing that $i \prec_g j$.

First if $F(AjBi) \geq F(ABji)$, then by $i \prec_\ell j$ we have $F(AjBi) \geq F(ABji) > F(ABij)$. So from now on we assume $F(AjBi) < F(ABji)$. Note that we could have assumed $F(AjBi) < F(AjiB)$ as well, but do not have use for it.

To conclude the proof, we will show $F(AjBi) > F(AiBj)$. In particular we will show the inequality

$$F(ABji) - F(ABij) < F(AjBi) - F(AiBj)$$

or equivalently

$$F(ABji) - F(AjBi) < F(ABij) - F(AiBj). \tag{5.8}$$

Let a be the total processing time of A and b the total processing time of AB. For every job $k \in B$ we denote by t_k the completion time of k in the schedule AB.

By hypothesis the left hand side is positive, and by $w_i/w_j > p_i/p_j$ and Lemma 5.3 we can bound it as (denoting $W_B(z) :=$)

$$\begin{aligned}
&F(ABji) - F(AjBi)\\
=&w_j(f(b+p_j) - f(a+p_j)) - \sum_{k\in B} w_k(f(t_k+p_j) - f(t_k))\\
<&w_i(f(b+p_i) - f(a+p_i)) - \frac{w_i}{w_j}\frac{f(b+p_i)-f(a+p_i)}{f(b+p_j)-f(a+p_j)} \sum_{k\in B} w_k(f(t_k+p_j) - f(t_k))\\
<&w_i(f(b+p_i) - f(a+p_i))\\
&- \frac{w_i}{w_j}\frac{f(b+p_i)-f(a+p_i)}{f(b+p_j)-f(a+p_j)} \min_{t\geq 0} \frac{f(t+p_j)-f(t)}{f(t+p_i)-f(t)} \sum_{k\in B} w_k(f(t_k+p_i) - f(t_k)).
\end{aligned}$$

In order to upper bound the later expression by

$$\leq w_i(f(b+p_i) - f(a+p_i)) - \sum_{k\in B} w_k(f(t_k+p_i) - f(t_k)) = F(ABij) - F(AiBj)$$

as required, it suffices to show

$$\frac{w_i}{w_j}\frac{f(b+p_i)-f(a+p_i)}{f(b+p_j)-f(a+p_j)} \min_{t\geq 0} \frac{f(t+p_j)-f(t)}{f(t+p_i)-f(t)} \geq 1.$$

This last step distinguishes two cases.

Case $\beta > 1$ By Lemma 5.4 the last fraction is minimum at $t = 0$, where it has the value $(p_j/p_i)^\beta$. By assumption $w_i/w_j \geq (p_i/p_j)^\beta$, the product of the first and last fraction is at least 1. By convexity of f, the second fraction is lower bounded by 1 as well, and we are done.

Case $0 < \beta < 1$ By Lemma 5.4 the last fraction is minimum at the limit $t \to \infty$, where it has the value p_j/p_i. By assumption $w_i/w_j \geq (p_i/p_j)^2$, we have that

$$\begin{aligned}
&\frac{w_i}{w_j}\frac{f(b+p_i)-f(a+p_i)}{f(b+p_j)-f(a+p_j)} \min_{t\geq 0} \frac{f(t+p_j)-f(t)}{f(t+p_i)-f(t)}\\
\geq\ &\left(\frac{p_i}{p_j}\frac{f(b+p_i)-f(a+p_i)}{f(b+p_j)-f(a+p_j)}\right)\left(\frac{p_i}{p_j}\frac{p_j}{p_i}\right),
\end{aligned}$$

which is at least 1 by Lemma 5.3. This concludes the proof of the theorem.

□

We provide a proof of the special case $\beta = 2$ of our conjecture.

Theorem 5.4. *Consider the penalty quadratic monomial function and fix two arbitrary jobs i, j. If $i \prec_\ell j$ then $i \prec_g j$.*

Proof. The case $p_i \leq p_j$, is covered by Theorem 5.1 for the interval $[0, \infty)$.

For the case $p_i > p_j$, by Lemma 5.1, ϕ_{ij} is strictly increasing and by Lemma 5.2 we have for any $t \geq 0$

$$\phi_{ij}(t) < p_i/p_j.$$

Thus, $\phi_{ij}(t) < w_i/w_j$ for any $t \geq 0$ (or equivalently $i \prec_\ell j$) if only if $p_i/p_j \geq w_i/w_j$. Finally, from the Mondal-Sen-Höhn-Jacobs rule we obtain $i \prec_g j$ (see [30]). □

In summary we obtain the following rules

Corollary 5.1 (our rules)**.** *Let i, j be two jobs, with $p_i > p_j$.*

If $\beta > 1$	
if $(p_i/p_j)^\beta \leq w_i/w_j$	*then* $i \prec_g j$
if $p_i/p_j < w_i/w_j$	*then* $i \prec_\ell j$ *(*)*
if $w_i/w_j < \phi_{ij}(0)$	*then* $j \prec_g i$
else $\exists t^* : w_i/w_j = \phi_{ij}(t^*)$ *and*	$i \prec_{\ell[0,t^*)} j$
	and $j \prec_{g[t^*,\infty)} i$
if $0 < \beta < 1$	
if $(p_i/p_j)^2 \leq w_i/w_j$	*then* $i \prec_g j$
if $\phi_{ij}(0) < w_i/w_j$	*then* $i \prec_\ell j$
if $w_i/w_j < p_i/p_j$	*then* $j \prec_{g[0,t_1]} i$
else $\exists t^* : w_i/w_j = \phi_{ij}(t^*)$ *and*	$i \prec_{g[0,t^*)} j$
	and $j \prec_{\ell[t^*,\infty)} i$,

where in case $\beta = 2$ (*) is enforced by the implication $i \prec_g j$.

Proof. We consider the case $\beta > 1$. The case $\beta < 1$ is symmetric, and the case $\beta = 2$ follows from the previous theorem. The first is Rule 3. For the second condition, ϕ_{ij} is strictly increasing by Lemma 5.1, and has limit p_i/p_j. Therefore $p_i/p_j < w_i/w_j$ implies $i \prec_\ell j$.

Monotonicity of ϕ_{ij} together with $\phi_{ij}(0) > w_i/w_j$ implies $\phi_{ij}(t) > w_i/w_j$ for all $t \geq 0$ as well, that is $j \prec_\ell i$. The implication $j \prec_g i$ follows by Theorem 5.1.

If none of the inequalities holds, by Lemma 5.1 and continuity of ϕ_{ij} there must be a unique time t^* such that $\phi_{ij}(t^*) = w_i/w_j$. In addition we have $i \prec_{\ell[0,t^*)} j$ and $j \prec_{\ell(t^*,\infty)} i$. We apply Theorem 5.1 to conclude $j \prec_{g(t^*,\infty)} i$. □

Figure 5.2 illustrates the contribution of our rules.

5.6 Experimental study

We conclude this paper with an experimental study, evaluating the impact of the proposed rules on the performance of a search procedure. Following the approach described in [30], we consider the Algorithm A* [28]. The search space is the directed acyclic graph consisting of all subsets $S \subseteq \{1, \ldots, n\}$. Note that the potential search space has size 2^n which is already less than the space of the $n!$ different schedules. In this graph for every vertex S there is an arc to $S \backslash \{j\}$ for any $j \in S$. It is labeled with j, and has cost $w_j t^\beta$ for $t = \sum_{i \in S} p_i$. Every

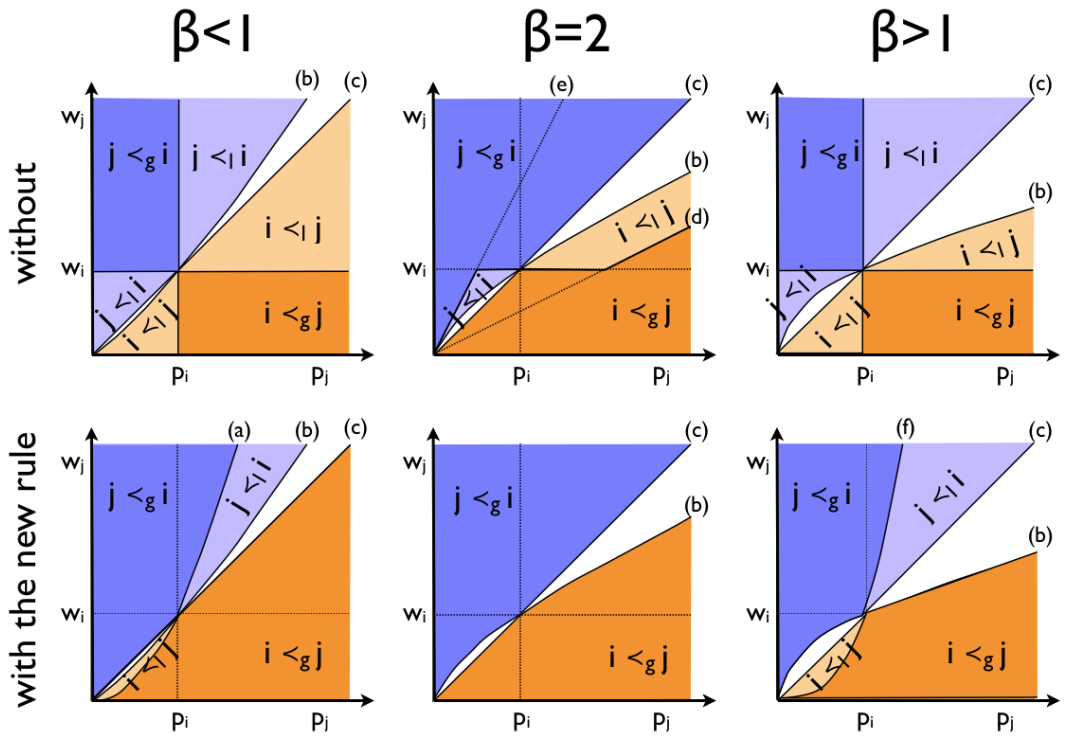


Figure 5.2: Job j compared to a fixed job i. Labels of particular functions: (a) $w_j = w_i(p_j/p_i)^2$, (b) $w_j = w_i((p_i+p_j)^\beta - p_i^\beta)/((p_i+p_j)^\beta - p_j^\beta)$, (c)$w_j = w_i p_j/p_i$, (d) $w_j = w_i p_j/2p_i$, (e) $w_j = 2w_i p_j/p_i$ and (f) $w_j = w_i(p_j/p_i)^\beta$.

directed path from the root $\{1,\ldots,n\}$ to $\{\}$ corresponds to a schedule of an objective value being the total arc cost.

So the goal is to compute the minimum distance between these two vertices. We use the algorithm A* for this purpose, which explores the graph using a priority queue containing arcs pointing to vertices that still need to be visited. An arc (S', S) has a weight corresponding to the distance from the root to S through this arc plus a basic lower bound of the optimum cost of scheduling S, which we choose to be simply $\sum_{i\in S} w_i p_i^\beta$.

Pruning is done when constructing the list of outgoing arcs at some vertex S. Potentially every job $i \in S$ can generate an arc, but ordering constraints might prevent that. Let j be the label of the arc leading to S (assuming S is not the root). Let $t_1 = \sum_{k\in S} p_k$. Now if $j \prec_{\ell(t_1-p_i)} i$, then no arc is generated for job $i \in S$. The same thing happens when there is a job $k \in S$ with $i \prec_{g[0,t_1]} k$. In a search tree such a pruning would cut the whole subtree attached to that arc, but in a directed acyclic graph the improvement is not so dramatic, as the typical indegree of a vertex is linear in n.

5.6.1 Random instances

We adopt the model of random instances described by Höhn and Jacobs. All previous experimental results were made by generating processing times and weights uniformly from some interval, which leads to easy instances, since any job pair i, j satisfies with probability 1/2 global precedence, i.e. $i \prec_g j$ or $j \prec_g i$. As an alternative, Höhn and Jacobs [30] proposed a

random model, where the Smith-ratio of a job is selected according to $2^{N(0,\sigma^2)}$ with N being the normal distribution centered at 0 with variance σ. Therefore for $\beta = 2$ the probability that two jobs satisfy global precedence depends on σ, since the Höhn-Jacobs-1 rule compares the Smith-ratio among the jobs.

We adopted their model for other values of β as follows. When $\beta > 1$, the condition for $i \prec_g j$ of our rules can be approximated, when p_j/p_i tends to infinity, by the relation $w_i/p_i \geq \beta w_j/p_j$. Therefore in order to obtain a similar "hardness" of the random instances for the same parameter σ for different values of $\beta > 1$, we choose the Smith-ratio according to $2^{N(0,\beta^2\sigma^2)}$. This way the ratio between the Smith-ratios of two jobs is a random variable from the distribution $2^{2N(0,\beta^2\sigma^2)}$, and the probability that this value is at least β depends only on σ.

However when β is between 0 and 1, the condition for $i \prec_g j$ of our rule can be approximated when p_j/p_i tends to infinity by the relation $w_i/p_i \geq 2w_j/p_j$, and therefore we choose the Smith-ratio of the jobs according to the β-independent distribution $2^{N(0,4\sigma^2)}$.

The instances of our main test sets are generated as follows. For each choice of $\sigma \in \{0.1, 0.2, \ldots, 1\}$ and $\beta \in \{0.5, 0.8, 1.1, \ldots, 3.2\}$. We generated 25 instances of 20 jobs each. The processing time of every job is uniformly generated in $\{1, 2, \ldots, 100\}$. Then the weight is generated according to the above described distribution. Note that the problem is independent on scaling of processing time or weights, motivating the arbitrary choice of the constant 100.

5.6.2 Hardness of instances

As a measure of the hardness of instances, we consider the portion of job pairs i, j which satisfy global precedence. By this we mean that we have either $i \prec_{g[0,t_1]} j$ or $j \prec_{g[0,t_1]} i$ for t_1 being the total processing time over all jobs excepting jobs i, j. Figure 5.4 shows this measure for various choices of β.

The results depicted in Figure 5.4 confirm the choice of the model of random instances. Indeed the hardness of the instances seems to depend only little on β, except for $\beta = 2$ where particular strong precedence rules have been established. In addition the impact of our new rules is significant, and further experiments show how this improvement influences the number of generated nodes, and therefore the running time. Moreover it is quite visible from the measures that the instances are more difficult to solve when they are generated with a small σ value.

5.6.3 Comparison between forward and backward variant

In this section, we consider two variants of the above mentioned algorithm. In the *forward* approach, a partial schedule describes a prefix of length t of a complete schedule and is extended to its right along an edge of the search tree, and in this variant the basic lower bound is $\sum_{i \in S} w_i(t + p_i)^\beta$. However in the *backward* approach, a partial schedule S describes a suffix of a complete schedule and is extended to its left. Kaindl, Kainz and Radda [38] give experimental evidence that the backward variant generates for some problems less nodes in the search tree, and this fact has also been observed by Höhn and Jacobs [30].

We conducted an experimental study in order to find out which variant is most likely to be more efficient. The results are shown in Figure 5.5. The values are most significative for small σ values, since for large values the instances are easy anyway and the choice of the variant is not very important. The results indicate that without our rules the forward variant

should be used whenever $\beta < 1$ or $\beta = 2$, while with our rules the forward variant should be used when $\beta > 1$.

Later on, when we measured the impact of our rules in the subsequent experiments, we compared the behavior of the algorithm using the most favorable variant dependent on the value of β as described above.

5.6.4 Timeout

During the resolution a timeout was set, aborting executions that needed more than a million nodes. In Figure 5.3 we show the fraction of instances that could be solved within the limited number of nodes. From these experiments we measure the instance sizes that can be efficiently solved, and observe that this limit is of course smaller when σ is small, as the instances become harder. But we also observe that with the usage of our rules much larger instances can be solved.

When β is close to 1, and instances consist of jobs of almost equal Smith-ratio, the different schedules diverge only slightly in cost, and intuitively one has to develop a schedule prefix close to the makespan, in order to find out that it cannot lead to the optimum. However for $\beta = 2$, the Mondal-Sen-Höhn-Jacobs rule make the instances easier to solve than for other values of β, even close to 2. Note that we had to consider different instance sizes, in order to obtain comparable results, as with our rules all 20 job instances could be solved.

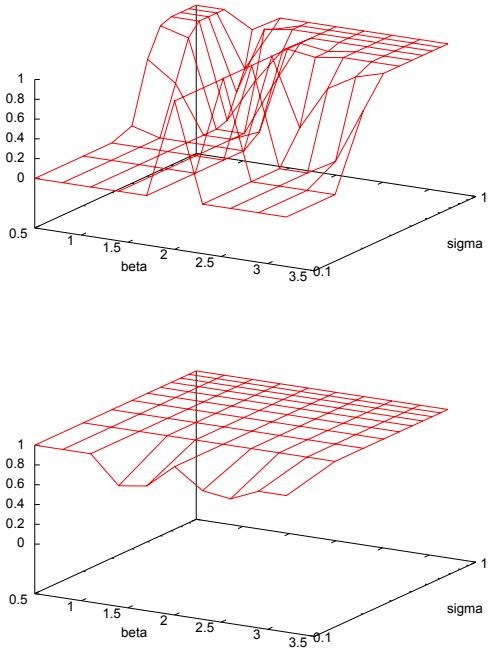


Figure 5.3: Proportion of instances which could be solved within the imposed time limit of a million nodes, with (below) and without (above) the new rules.

5.6.5 Improvement factor

In this section we measure the influence on the number of nodes generated during a resolution when our rules are used. For $\beta = 2$ we compare our performance with the Mondal-Sen-Höhn-Jacobs rule, while for other values of β we compare with the Sen-Dileepan-Ruparel rule. For fairness we excluded instances where the timeout was reached without the use of our rules. Figure 5.6 shows the ratio between the average number of generated nodes when the algorithm is run with our rules, and when it is run without our rules. Clearly this factor is smaller for $\beta = 2$, since the Mondal-Sen-Höhn-Jacobs rules apply here.

We observe that the improvement factor is more important for hard instances, i.e. when σ is small. From the figures it seems that this behavior is not monotone, for $\beta = 1.1$ the factor is less important with $\sigma = 0.1$ than with $\sigma = 0.3$. However this is an artifact of our pessimistic measurements, since we average only over instances which could be solved within the time limit, so in the statistics we filtered out the really hard instances.

5.6.6 Performance measurements for $\beta = 2$

For $\beta = 2$, the authors of [30] provide several test sets to measure the impact of their rules in different variants, see [31]. For completeness we selected two data sets from their collection to compare our rules with theirs.

The first set called `set-n` contains for every number of jobs $n = 1, 2, \ldots, 35$, 10 instances generated with parameter $\sigma = 0.5$. This file permits to measure the impact of our rules as a function on the instance size.

The second test set that we considered is called `set-T` and contains for every parameter $\sigma = 0.100, 0.101, 0.102, \ldots, 1.000$ 3 instances of 25 jobs. Results are depicted in figure 5.7.

For a general analysis, we generated instances described in section 5.6.1, and compared the average number of nodes generated with and without our rules.

5.6.7 Performance depending on input size

In addition we show the performance of the algorithm with our rules, in dependence on the number of jobs. Figure 5.8 shows for different number of jobs the number of generated nodes averaged over 100 instances generated with different σ parameters, exposing an expected running time which strongly depends on the hardness of the instances.

5.7 Special case $\beta = -1$: the airplane refueling problem

5.7.1 Statement of problem

The airplane refueling problem is motivated by George Gamow and Marvin Stern in the aeronautica chapter of the book "Puzzle-Math" [24]. They describe the problem as follows:

"Well, here's another problem which may interest you fellows", said another pilot. "Suppose you have to deliver a bomb in some distant point of the globe, the distance being much greater than the range of the plane you are going to use. Thus, you have to use the technique of refueling in the air. Starting with several identical planes which refuel one another, and gradually drop out of the flight until the single plane carrying the bomb reaches the target,

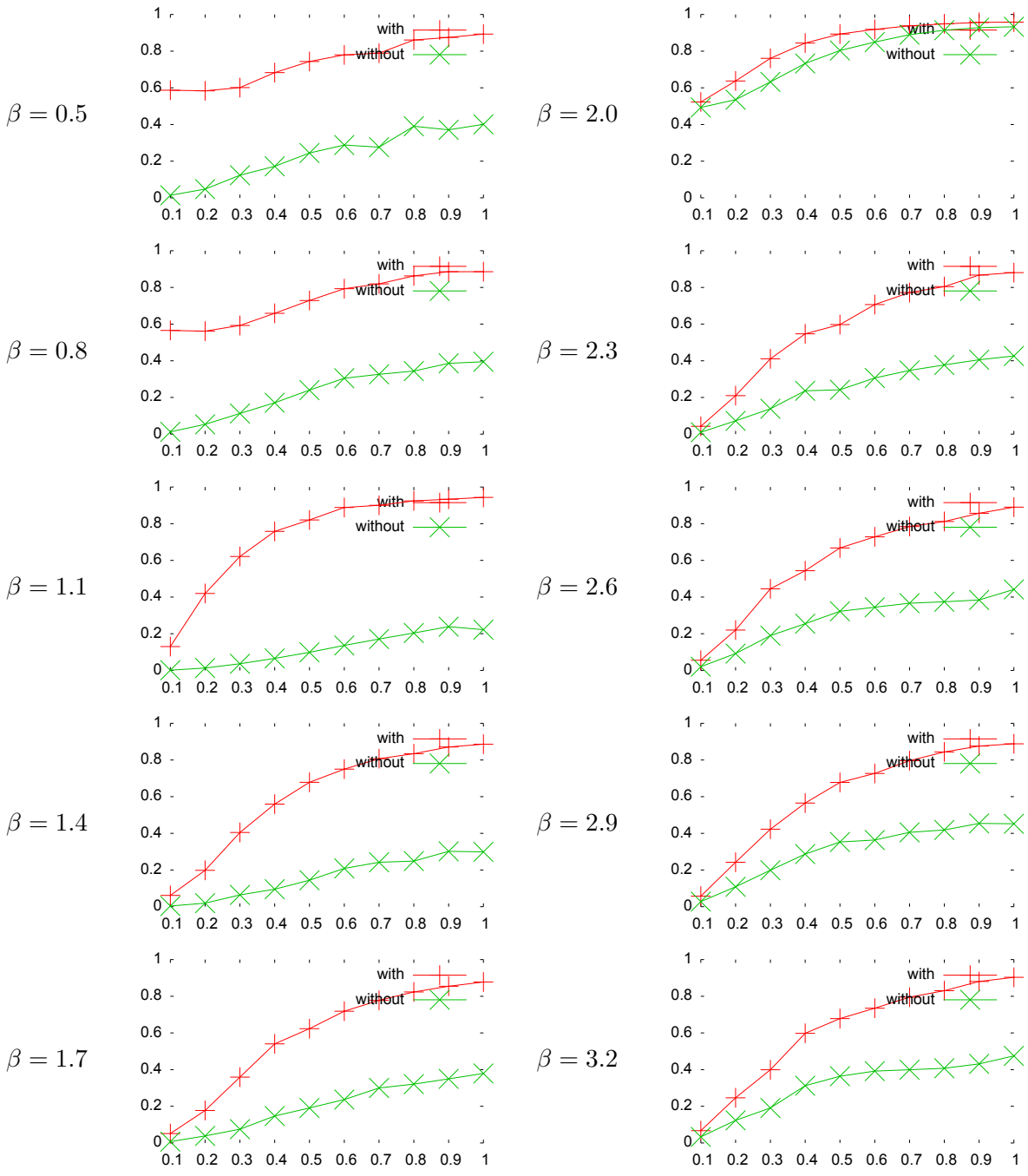


Figure 5.4: Proportion of job pairs that satisfy a global precedence relation as function of the parameter σ used in the random generation of the instances.

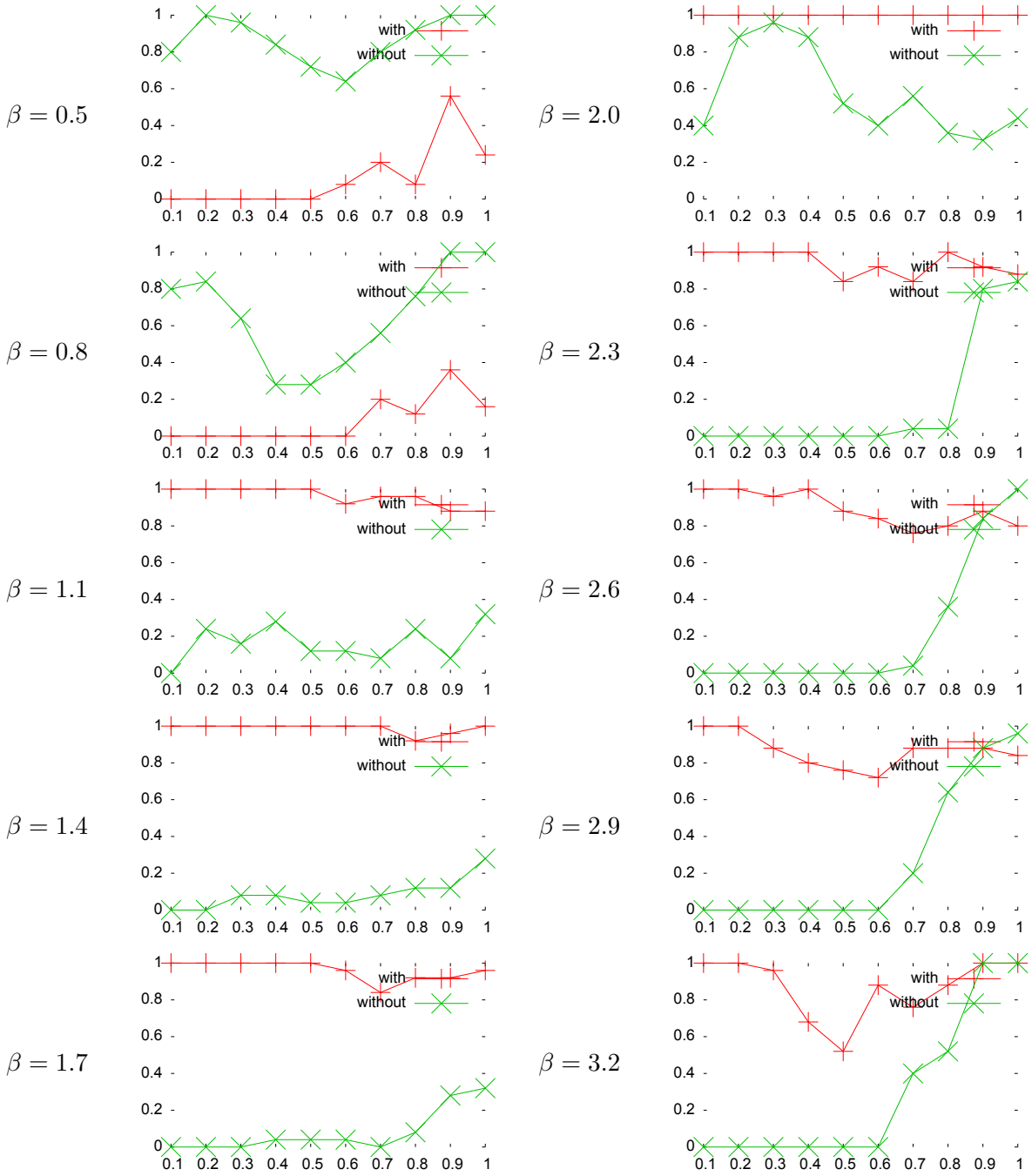


Figure 5.5: Proportion of instances for which the forward variant generated less nodes than the backward variant. The values are plotted as function of σ, both for the resolution with our new rules and without.

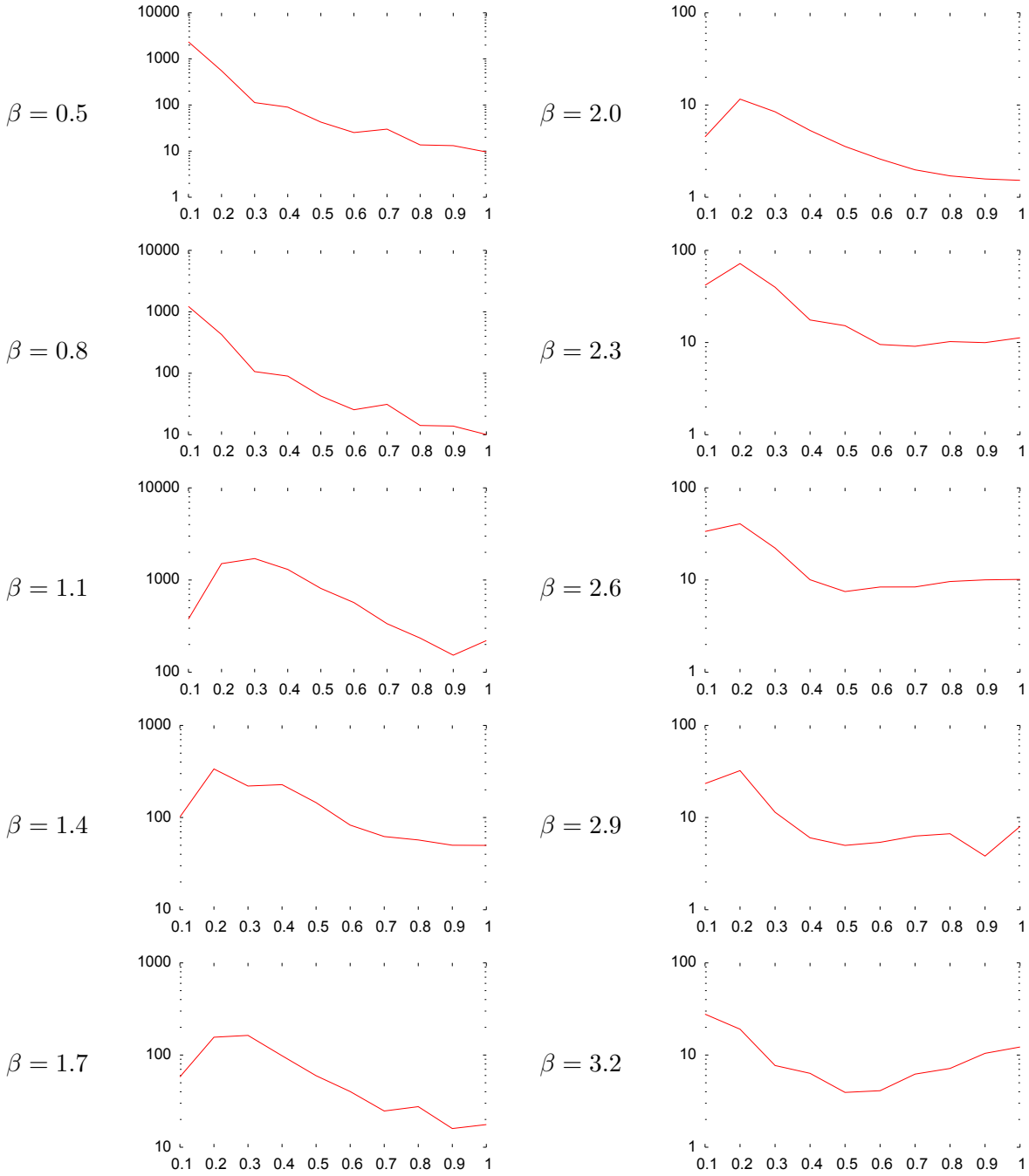


Figure 5.6: Average improvement factor as function of β and σ

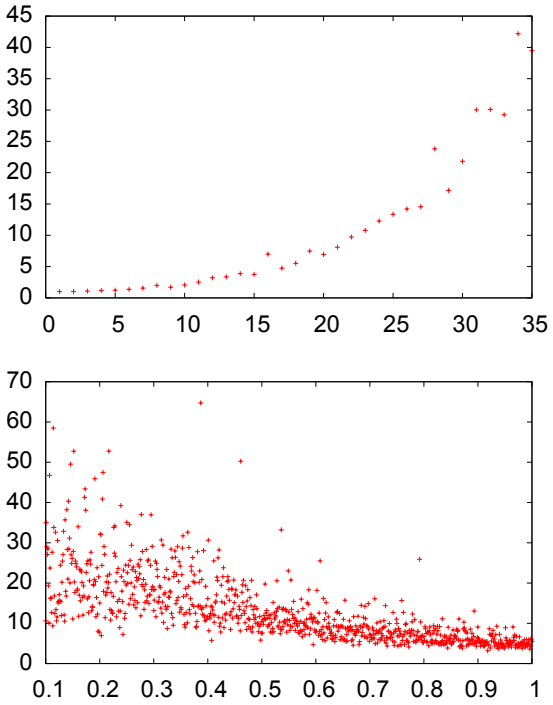


Figure 5.7: Improvement ratio for test sets `set-n` (left) and `set-T` (right)

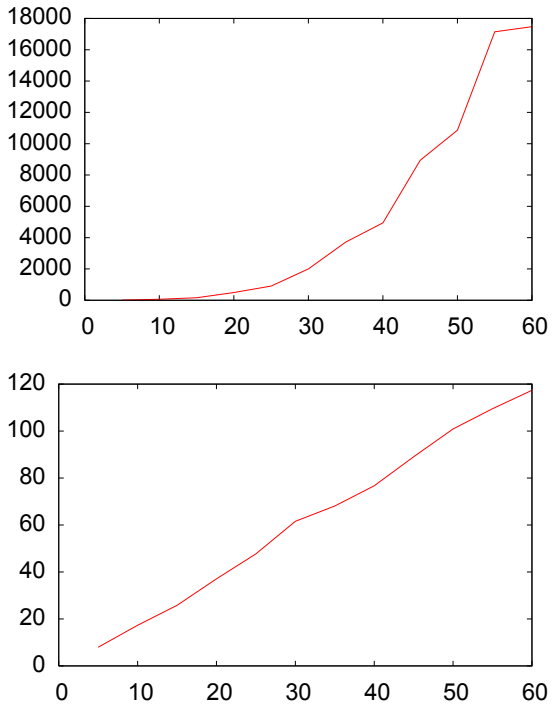


Figure 5.8: Average number of nodes in dependance on the size of the instances, generated with $\sigma = 0.1$ on the left and $\sigma = 0.5$ on the right.

how would you plan the refueling program, and how many planes will you need to carry out the operation? We will assume for simplicity that airplane fuel consumption can be measured in miles per gallon and is independent of load."
"Oh, just tell us," said one of the pilots. "We are all too tired to work out these problems".

On a Dagstuhl scheduling workshop in 2010, Gerhard J. Woeginger presented a generalization of this problem and proposed the following mathematical formulation [62]: We are given n airplanes. Each airplane j has a full reservoir of capacity w_j liters and consumption rate of p_j liters per kilometer. At time zero, the airplanes flight at the same speed in the same straight direction, and can freely redistribute the fuel between their reservoirs. As soon as one airplane runs out of fuel, it is dropping out of the flight, and the goal is to reach a maximal distance with the last plane in the air.

A solution is completely described by a *drop out ordering* σ as follows. Initially all planes leave the origin with total fuel $\sum w_j$ and with total consumption rate $\sum p_j$. The first time airplane $\sigma(1)$ can drop out is the time t such that $w_{\sigma(1)} = t/\sum p_j$, since by the limited capacity of the other planes it is impossible to empty his reservoir earlier. Also it does not make sense to drop out $\sigma(1)$ later, because it will consume fuel which will be missed later on. Therefore the objective value of the drop out order σ is

$$\sum_{j=1}^{n} \left(w_{\sigma(j)} / \sum_{k=j}^{n} p_{\sigma(k)} \right).$$

Phrased as a scheduling setting, the problem is equivalent to finding a permutation π (the reverse of σ), which maximizes

$$\sum_{j=1}^{n} \left(w_{\pi(j)} / \sum_{k=1}^{j} p_{\pi(k)} \right) = \sum_{j=1}^{n} w_j / C_j, \tag{5.9}$$

where C_j is the completion time of job j, p_j its processing time and w_j its weight.

The problem is known to be easy for agreeable instances, i.e. whenever $w_i > w_j$ implies $p_i \leq p_j$, while the computational complexity for general instances is open. Recently, Wiebke Höhn on a Dagstuhl scheduling workshop in 2013 [29], used this problem to motivate the study of a larger class of scheduling problems with the goal of minimizing $\sum w_j f(C_j)$ for some given concave and increasing monotone penalty function. For example the function $f(t) = \frac{-1}{t}$ models precisely the jeep refueling problem.

For most problems of this form, the complexity status is open (see [59] or Section 4.3). In the previous section we stated the conjecture that local precedence implies global precedence on a single machine problem minimizing $\sum w_j f(C_j)$ for penalty functions of the form $f(t) = t^\beta$ for any constant $\beta > 0$, formally that $i \prec_\ell j$ implies $i \prec_g j$. In this paper we prove an analogue of this conjecture for $\beta = -1$, that is when the objective is to *maximize* $\sum w_j / C_j$.

5.7.2 Preliminaries

We introduce three functions, $H(S)$ which is the profit of schedule S as defined in (5.9), $h(t) = 1/t$ and the following function, parameterized by jobs i, j and depending on $t \in \mathbb{R}^+$,

$$\begin{aligned}\phi_{ij}(t) = \frac{h(t+p_j) - h(t+p_i+p_j)}{h(t+p_i) - h(t+p_i+p_j)} &= \frac{\frac{1}{t+p_j} - \frac{1}{t+p_i+p_j}}{\frac{1}{t+p_i} - \frac{1}{t+p_i+p_j}} \\ &= \frac{p_i}{p_j} \cdot \frac{(t+p_i)(t+p_i+p_j)}{(t+p_i)(t+p_i+p_j)} \\ &= \frac{p_i}{p_j} \cdot \frac{t+p_i}{t+p_j}.\end{aligned}$$

Note that $\phi_{ij}(t)$ is well defined since the durations p_i, p_j are non-zero. By definition we have the following equivalence.

$$i \prec_{\ell(t)} j \;\Leftrightarrow\; \phi_{ij}(t) < \frac{w_i}{w_j}.$$

We start by analyzing properties of $\phi_{ij}(t)$.

Lemma 5.5. *For any jobs i, j, we have*

$$\lim_{t \mapsto \infty} \phi_{ij}(t) = p_i/p_j.$$

Proof. Indeed

$$\lim_{t \mapsto \infty} \phi_{ij}(t) = \lim_{t \mapsto \infty} \frac{p_i}{p_j} \frac{t+p_i}{t+p_j} = \frac{p_i}{p_j} \lim_{t \mapsto \infty} \frac{1+p_i/t}{1+p_j/t} = \frac{p_i}{p_j}.$$

□

Lemma 5.6. *If $p_i \neq p_j$ then $\phi_{ij}(t)$ is strictly monotone and bounded, in particular:*

- *If $p_i > p_j$, then $\phi_{ij}(t)$ is strictly decreasing, convex and $\phi_{ij}(t) \in \left(\frac{p_i}{p_j}, \left(\frac{p_i}{p_j}\right)^2\right]$.*
- *If $p_i < p_j$, then $\phi_{ij}(t)$ is strictly increasing, concave and $\phi_{ij}(t) \in \left[\left(\frac{p_i}{p_j}\right)^2, \frac{p_i}{p_j}\right)$.*

Proof. To prove the monotonicity of $\phi_{ij}(t)$, we use its first derivative

$$\phi'_{ij}(t) = \frac{p_i}{p_j} \frac{p_j - p_i}{(t+p_j)^2},$$

and second derivative

$$\phi''_{ij}(t) = -2\frac{p_i}{p_j} \frac{p_j - p_i}{(t+p_j)^3}.$$

Clearly, the function $\phi_{ij}(t)$ is strictly decreasing and convex when $p_i > p_j$ and strictly increasing and concave when $p_j > p_i$.

To bound the function $\phi_{ij}(t)$, we analyze its codomain. Since $\phi_{ij}(t)$ is strictly monotone, it is suffices to evaluate its extreme points. At $t = 0$, we have

$$\phi_{ij}(0) = \left(\frac{p_i}{p_j}\right)^2,$$

whereas, the limit of $\phi_{ij}(t)$ when t tends to infinity is $\frac{p_i}{p_j}$ by Lemma 5.5. □

Lemma 5.7. *Fix two jobs i and j. If there exist $t^* \in \mathbb{R}^+$ such that*

$$\frac{w_i}{w_j} = \phi_{ij}(t^*),$$

then t^* is unique and equal to

$$\frac{w_j p_i^2 - w_i p_j^2}{w_i p_j - w_j p_i}$$

Proof. We assume that there exist $t^* \in \mathbb{R}^+$ such that

$$\frac{w_i}{w_j} = \phi_{ij}(t^*).$$

The strict monotonicity of $\phi_{ij}(t)$ by Lemma 5.6 implies uniqueness of t^*. We have

$$\frac{w_i}{w_j} = \phi_{ij}(t^*) = \frac{p_i}{p_j} \cdot \frac{t^* + p_i}{t^* + p_j},$$

and arranging the above expression, we obtain

$$t^* = \frac{w_j p_i^2 - w_i p_j^2}{w_i p_j - w_j p_i}$$

□

See Figure 5.9 for an illustration of the claimed properties.

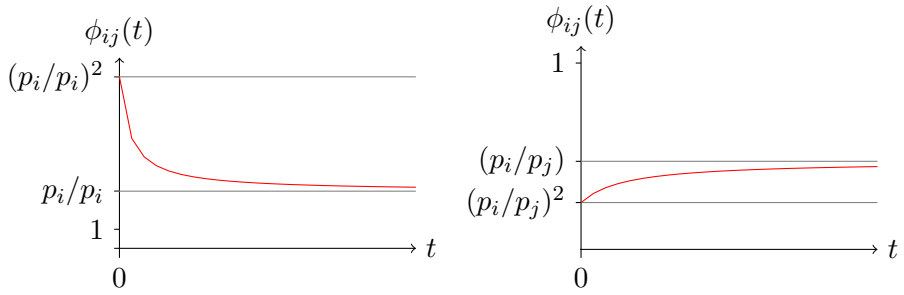


Figure 5.9: Function $\phi_{ij}(t)$ in case $p_i > p_j$ (left) and case $p_i < p_j$ (right).

5.7.3 Main Result

The main result of our paper is that local precedence implies global precedence for this problem, which is stated formally as follows.

Theorem 5.5. *For all jobs i, j and time points a, b the property $i \prec_{\ell[a,b]} j$ implies $i \prec_{g[a,b]} j$.*

Proof. We consider time points a, b and an arbitrary instance containing jobs i, j. Assume that there is an optimal schedule $AjBiD$ for some job sequences A, B, D. Let a' be the total length of A and b' be the total length of AB. For the proof we will show that if $[a', b'] \subseteq [a, b]$ and $i \prec_{\ell[a,b]} j$, then the schedule $AiBjD$ is optimal as well.

Since $i \prec_{\ell[a,b]} j$ implies $i \prec_{\ell[a',b']} j$ for the purpose of simplifying notation, without loss of generality we can assume $a' = a$ and $b' = b$.

We start the proof with a technical claim, comparing the effect of exchanging B and i with the effect of exchanging B and j.

Claim 5.1.

$$\max_{t\in[a,b]} \phi_{ij}(t)\,[H(AiBjD) - H(AijBD)] > H(AjBiD) - H(AjiBD). \tag{5.10}$$

Proof. This follows directly from Theorem (5.1). □

The remaining of the proof distinguishes two cases.

Case $p_i \leq p_j$ In this case we only need that h is strictly decreasing. The optimality assumption means that the right hand side of (5.10) is non-negative and therefore the left hand side is positive. Fix an arbitrary $t \in [a, b]$. Note that $\phi_{ij}(t) \leq 1$ if $p_i \leq p_j$ by definition of ϕ_{ij}. Also we have $\phi_{ij}(t) < w_i/w_j$ by case assumption $i \prec_{\ell[a,b]} j$.

This implies by (5.4)

$$H(AiBjD) - H(AijBD) > H(AjBiD) - H(AjiBD),$$

or equivalently

$$H(AiBjD) - H(AjBiD) > H(AijBD) - H(AjiBD). \tag{5.11}$$

Local precedence of i and j implies that the right hand side is positive, and therefore $H(AiBjD) > H(AjBiD)$ contradicting optimality of $AjBiD$.

Case $p_i > p_j$ Again we will show $H(AiBjD) > H(AjBiD)$ contradicting optimality. For this purpose we decompose the difference as follows.

$$\begin{aligned} H(AjBiD) - H(AiBjD) = \;& H(AjBiD) - H(ABjiD) \\ & + H(ABjiD) - H(ABijD) \\ & + H(ABijD) - H(AiBjD). \end{aligned}$$

Local precedence implies $H(ABjiD) < H(ABijD)$, hence to conclude the proof it suffices to show

$$H(AjBiD) - H(ABjiD) < H(AiBjD) - H(ABijD).$$

For every job $k \in B$ we denote by t_k the completion time of k in the schedule $ABijD$.

We show the following inequalities with explanations below.

$$
\begin{aligned}
0 <& H(AjBiD) - H(ABjiD) \\
=& \frac{w_j}{a+p_j} + \sum_{k\in B} \frac{w_k}{t_k+p_j} - \frac{w_j}{b+p_j} - \sum_{k\in B} \frac{w_k}{t_k} \\
=& \frac{w_j(b-a)}{(a+p_j)(b+p_j)} - \sum_{k\in B} \frac{w_k p_j}{(t_k+p_j)t_k} \\
<& \frac{w_i}{w_j}\left(\frac{p_i}{p_j}\right)^2 \phi_{ji}(a)\phi_{ji}(b)\left(\frac{w_j(b-a)}{(a+p_j)(b+p_j)} - \sum_{k\in B} \frac{w_k p_j}{(t_k+p_j)t_k}\right) \\
=& \frac{w_i(b-a)}{(a+p_i)(b+p_i)} - \frac{w_i}{w_j}\left(\frac{p_i}{p_j}\right)^2 \phi_{ji}(a)\phi_{ji}(b)\left(\sum_{k\in B} \frac{w_k p_j}{(t_k+p_j)t_k}\right) \\
=& \frac{w_i(b-a)}{(a+p_i)(b+p_i)} - \frac{w_i}{w_j}\left(\frac{p_i}{p_j}\right)^2 \phi_{ji}(a)\phi_{ji}(b)\left(\sum_{k\in B} \frac{w_k p_i}{(t_k+p_i)t_k}\phi_{ij}(t_k)\left(\frac{p_j}{p_i}\right)^2\right) \\
=& \frac{w_i(b-a)}{(a+p_i)(b+p_i)} - \frac{w_i}{w_j}\phi_{ji}(a)\phi_{ji}(b)\left(\sum_{k\in B} \frac{w_k p_i}{(t_k+p_i)t_k}\phi_{ij}(t_k)\right) \\
<& \frac{w_i(b-a)}{(a+p_i)(b+p_i)} - \frac{w_i}{w_j}\phi_{ji}(a)\phi_{ji}(b)\min_{t\in[a,b]}\phi_{ij}(t)\left(\sum_{k\in B} \frac{w_k p_i}{(t_k+p_i)t_k}\right) \\
=& \frac{w_i(b-a)}{(a+p_i)(b+p_i)} - \frac{w_i}{w_j}\phi_{ji}(a)\phi_{ji}(b)\phi_{ij}(b)\left(\sum_{k\in B} \frac{w_k p_i}{(t_k+p_i)t_k}\right) \\
=& \frac{w_i(b-a)}{(a+p_i)(b+p_i)} - \frac{w_i}{w_j}\phi_{ji}(a)\left(\sum_{k\in B} \frac{w_k p_i}{(t_k+p_i)t_k}\right) \\
<& \frac{w_i(b-a)}{(a+p_i)(b+p_i)} - \sum_{k\in B} \frac{w_k p_i}{(t_k+p_i)t_k} \\
=& \frac{w_i}{a+p_i} - \frac{w_i}{b+p_i} + \sum_{k\in B} \frac{w_k}{t_k+p_i} - \sum_{k\in B} \frac{w_k}{t_k} \\
=& \frac{w_i}{a+p_i} + \sum_{k\in B} \frac{w_k}{t_k+p_i} - \sum_{k\in B} \frac{w_k}{t_k} - \frac{w_i}{b+p_i} \\
=& H(AiBjD) - H(ABijD)
\end{aligned}
$$

The first inequality holds by case assumption, the second inequality by Lemma 5.6 which implies

$$\left(\frac{p_i}{p_j}\right)^2 \geq \max_{t\in[a,b]} \phi_{ij}(t)$$

and by case assumption,

$$\frac{w_i}{w_j} > \max_{t\in[a,b]} \phi_{ij}(t).$$

The third inequality follows from strict monotonicity of $\phi_{ij}(t)$ implied by Lemma 5.6 when $p_i > p_j$, therefore

$$\min_{t\in[a,b]} \phi_{ij}(t) = \phi_{ij}(b) = \frac{1}{\phi_{ji}(b)}.$$

Finally the later inequality is implied by the locality assumption.

$$\frac{w_i}{w_j} > \phi_{ij}(a),$$

which implies

$$\frac{w_i}{w_j}\phi_{ji}(a) > 1,$$

concluding the proof.

□

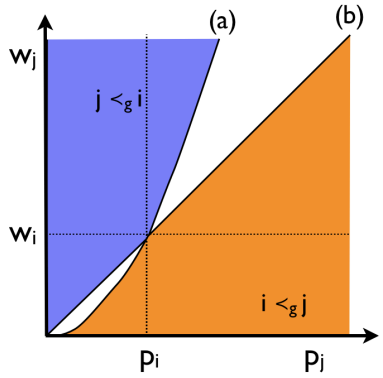


Figure 5.10: Job j compared to a fixed job i. Labels of particular functions: (a) $w_j = w_i(p_i/p_j)^2$, (b) $w_j = p_j w_i/p_i$

In summary we obtain the following precedence rules, see Figure 5.10.

Corollary 5.2. *For any two jobs i, j with $w_i > w_j$:*

$$\begin{array}{ll} \textit{If} & w_i/w_j \geq \left(\frac{p_i}{p_j}\right)^2 \quad \textit{then } i \prec_g j \\ \textit{If} & p_i/p_j \geq w_i/w_j \quad \textit{then } j \prec_g i \\ \textit{else } \exists t^* = \frac{w_j p_i^2 - w_i p_j^2}{w_i p_j - w_j p_i} \geq 0 & \textit{with } i \prec_{g[0,t^*)} j \textit{ and } j \prec_{g[t^*,\infty)} i. \end{array}$$

5.7.4 Experimental study

Again, in the absence of a polynomial time algorithm we solve the problem using the algorithm A*. We implemented the algorithm described above to measure the impact of the prunning rules, and tested it against randomly generated instances. The processing time was generated uniformly between 1 and 100. If we would have generated weights also uniformly from some interval, then for any job pair, with probability at least 1/2 we would have either $i \prec_g j$ or $j \prec_g i$ which would make the instances too easy to solve. In this case, we have that the condition for $i \prec_g j$ of our rule can be approximated when p_j/p_i tends to infinity by the relation $w_i/p_i \geq 2w_j/p_j$, and therefore we choose the Smith-ratio of the jobs according to distribution $2^{N(0,\sigma^2)}$, where $N(0,\sigma^2)$ is the normal distribution of mean 0 and standard

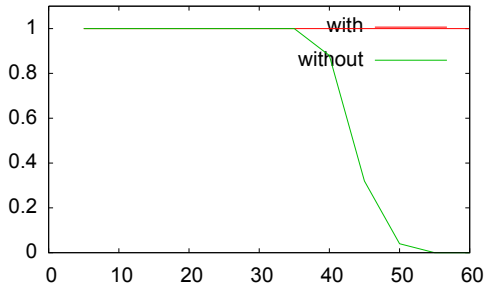


Figure 5.11: Fraction of instances solved within the timeout limit of a million nodes as function of the number of jobs per instance, when the algorithm is run with or without the rules of this paper. Even for smaller σ values we were not able to generate instances on which the algorithm fails by timeout when used with our rules.

deviation σ. This parameter permitted to tune the difficulty of the instances, as for small σ, jobs tend to have similar Smith-ratios.

We generated two data sets. The first data set contains 25 instances of 20 jobs each per σ value, which we draw from $\{0.1, 0.2, \ldots, 1\}$ while the second data set contains 25 instances for each number of jobs from $\{5, 10, \ldots, 60\}$ with a fixed value $\sigma = 0.5$.

In the above described graph modelization vertices correspond to prefixes of schedules. An alternative approach would have been to construct schedules from the end, where vertices would correspond to suffixes of schedules. We implemented this approach as well, and run the experiments on the first data set. For all instances, this alternative approach generated strictly more nodes in the search tree. We believe that the reason is that the early jobs of a schedule contribute with a higher value to the objective than the later jobs, and therefore it is more important to determine the prefix of an optimal schedule than the suffix.

In order to run the experiments in a reasonable time, we stopped the algorithm after a timeout of a million generated nodes. Every instances was solved twice, with and without the precedence rules provided by this paper. Without our rules, we only considered $i \prec_g j$ whenever $p_i < p_j$ and $w_i \geq w_j$. Our experiments on the second data set showed, that with our rules, instances of up to 60 jobs can be solved within the selected timeout, while without our rules roughly half of the instances with 40 jobs could not be solved, see Figure 5.11.

Furthermore we measured the behavior of our algorithm in terms of number of nodes on both data sets. Results are depicted in Figure 5.12. It is quite visible from the plots that the difficulty of the problem increases when σ decreases, and that the algorithm shows exponential running time, which is not a surprise.

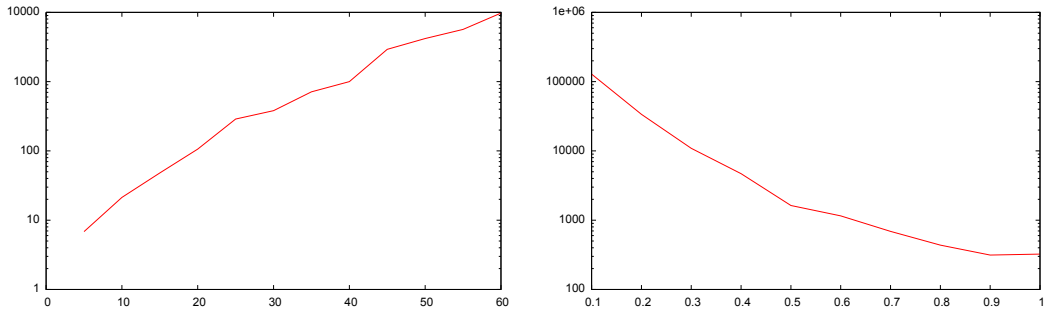


Figure 5.12: Average number of nodes generated for solving random instances, as a function of the number of jobs with fixed $\sigma = 0.5$ (left) and as a function on σ with number of jobs fixed to 20 (right).

Chapitre 6
Perspectives (français)

Cette thèse s'inscrit dans le contexte des systèmes informatiques dotés de microprocesseurs de dernière génération (e.g. Intel SpeedStep, AMD PowerNow ! ou IBM EnergyScale) qui varient dynamiquement la fréquence du processeur et influencent ainsi à la fois la qualité de service que la consommation d'énergie. Nous avons abordé justement le problème d'ordonnancement sur un processeur capable de varier la vitesse dans le but de minimiser la somme de l'énergie consommée plus le temps d'attente pondéré des tâches, quantité que nous définissons comme le *le coût social* à optimiser.

Nous avons abordé ce problème sous deux formes. D'une part sous forme d'un jeu stratégique. Les joueurs veulent une chose, que leur tâche termine le plus tôt possible. Par contre la machine veut un ordonnancement qui consomme le moins d'énergie possible, et ces deux objectifs sont en opposition. Ici la contribution principale consiste à définir un mécanisme de facturation qui force les joueurs à adopter un comportement aux bonnes propriétés pour la société.

Notre deuxième contribution à ce problème est une réduction à un problème d'ordonnancement avec une fonction objective concave, pour lequel nous avons apporté des propriétés de dominance, utile dans une résolution exacte.

Notre travail est loin de clore le sujet, et tout au contraire représente un point de départ et une motivation pour de nouvelles questions de recherche, telles que :

- Quelle est la complexité du problème $1||\sum_j w_j sign(\beta) C_j^\beta$? Est-ce que les résultats structurels forts pour le cas $\beta = -1$ permettraient un algorithme polynomial ?
- Est-ce que l'ordre de précédence local entre deux tâches i, j implique l'ordre de précédence global ?
- Quand est-ce que l'équilibre de Nash pur est unique ?
- Comment améliorer le prix d'anarchie du jeu de pénalité ? Est-ce qu'une enchère sur les rangs dans l'ordonnancement pourrait avoir de bonnes propriétés ?
- Plus d'impact encore sur la société aurait l'étude des problèmes de minimisation d'énergie dans le modèle d'ordonnancement avec hibernation, dans les grilles de calcul, ou une politique d'incitation pour les fabricants d'appareils électroniques et électroménagers.
- Une caractéristique importante du modèle étudié est que le coût énergétique est une fonction polynomiale dans la vitesse de travail. Il existe par contre de nombreuses ressources dont le coût n'est pas linéaire dans leur utilisation, comme la fabrication d'électricité ou de produits par exemple. Aussi l'efficacité d'un travailleur ou d'une machine peut changer au cours du temps, par effet d'usure ou d'apprentissage au contraire.

Ainsi nous espérons que les techniques développées dans cette thèse, permettront d'aborder de nouveaux problèmes avec des coûts non linéaires.

Chapter 7
Perspectives (english)

The studied environment in this thesis concerns computing systems with last generation microprocessors (like Intel SpeedStep, AMD PowerNow or IBM EnergyScale) which can change dynamically their clock speed, and hence influence both on the energy consumption and the quality of service. In particular, we addressed the scheduling problem for a single processor with speed scaling ability in order to minimize the consumed energy and the total flow time. We called this amount the *cost social*, and tried optimize it in two settings. On the one hand, we formulated this problem as a strategic game. Every player wants to minimize a weighted sum of the waiting time of his job. But the machine wants a schedule that consumes as little energy as possible and therefore we face opposite goals. Here our main contribution was to define a cost sharing mechanism that incites players to adopt a good behavior for society. In the centralized setting, our second contribution was to reduce the optimization problem to a scheduling problem with a concave objective function, for which we have generated dominance properties, which turn out to be quite useful in an exact resolution method. Our work is far from closing the topic, but instead is a starting point and motivation for new research questions, such as:

- What is the complexity of the problem $1||\sum_j w_j sign(\beta) C_j^{\beta}$? Do our strong structural results for the case β = -1 allow a polynomial algorithm?
- Does the local precedence order between jobs i, j implies the global precedence order?
- When is a pure Nash equilibrium unique?
- How to improve the price of anarchy of the penalty game? Could we improve the game by defining a second price auction on the rank positions in the schedule?
- In order to produce a stronger impact on the society we would like to study the energy minimization in the scheduling model with hibernation in grid computing. Also we want to study mechanism design for ecological incentives for electronics manufacturers and appliances.
- An important feature of the studied model is that the energy cost is a polynomial function in the working speed. This behavior can be observed as well for other resources, where the usage cost is not linear in their utilization. Examples are the electricity production and the manufacturing of goods. Also the efficiency of a worker or a machine can change over time, by a wear and tear effect, or even a learning effect. So we hope that the techniques developed in this thesis will generalize to new problems with nonlinear costs.

Bibliography

[1] S. Albers. Energy-efficient algorithms. *Communications of the ACM*, 53(5):86–96, 2010.

[2] S. Albers and H. Fujiwara. Energy-efficient algorithms for flow time minimization. *ACM Transactions on Algorithms (TALG)*, 3(4):49, 2007.

[3] B. Alidaee. Numerical methods for single machine scheduling with non-linear cost functions to minimize total cost. *Journal of the Operational Research Society*, 44(2):125–132, 1993.

[4] E. Angel, E. Bampis, and F. Kacem. Energy aware scheduling for unrelated parallel machines. In *Proc. of the IEEE International Conference on Green Computing and Communications (GreenCom)*, pages 533–540, 2012.

[5] T. Badics and E. Boros. Minimization of half-products. *Mathematics of Operations Research*, 23(3):649–600, 1998.

[6] U. Bagchi, R. S. Sullivan, and Y. L. Chang. Minimizing mean squared deviation of completion times about a common due date. *Management Science*, 33(7):894–906, 1987.

[7] P. Bagga and K. Karlra. A node elimination procedure for Townsend's algorithm for solving the single machine quadratic penalty function scheduling problem. *Management Science*, 26(6):633–636, 1980.

[8] N. Bansal, H. L. Chan, D. Katz, and K. Pruhs. Improved bounds for speed scaling in devices obeying the cube-root rule. *Theory of Computing*, 8:209–229, 2012.

[9] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *Journal of the ACM (JACM)*, 54(1):3, 2007.

[10] N. Bansal and K. Pruhs. The geometry of scheduling. In *Proc. of the IEEE 51st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 407–414, 2010.

[11] D. M. Brooks, P. Bose, S. E. Schuster, H. Jacobson, P. N. Kudva, A. Buyuktosunoglu, J. Wellman, V. Zyuban, M. Gupta, and P. W. Cook. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *Micro, IEEE*, 20(6):26–44, 2000.

[12] R. A. Carrasco, G. Iyengar, and C. Stein. Energy aware scheduling for weighted completion time and weighted tardiness. Technical Report arXiv:1110.0685, arxiv.org, 2011.

[13] S. H. Chan, T. W. Lam, and L. K. Lee. Non-clairvoyant speed scaling for weighted flow time. In *Proc. of the 18th Annual European Symposium (ESA)*, pages 23–35, 2010.

[14] T. Cheng, C. Ng, J. Yuan, and Z. Liu. Single machine scheduling to minimize total weighted tardiness. *European Journal of Operational Research*, 165(2):423–443, 2005.

[15] T. E. Cheng and Z. Liu. Parallel machine scheduling to minimize the sum of quadratic completion times. *IIE transactions*, 36(1):11–17, 2004.

[16] M. Cheung and D. Shmoys. A primal-dual approximation algorithm for min-sum single-machine scheduling problems. In *Proc. of the 14th International Workshop APPROX and 15th International Workshop RANDOM*, pages 135–146, 2011.

[17] F. Croce, R. Tadei, P. Baracco, and R. Di Tullio. On minimizing the weighted sum of quadratic completion times on a single machine. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 816–820, 1993.

[18] J. Du and J. Leung. Minimizing total tardiness on one machine is NP-hard. *Mathematics of Operations Research*, 15(3):483–495, 1990.

[19] C. Dürr. Scheduling zoo. "http://www-desir.lip6.fr/ durrc/query/", 2013.

[20] C. Dürr and O. C. Vásquez. Order constraints for single machine scheduling with non-linear cost. In *Proc. of the 16th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 98–111, 2014.

[21] D. Easley and J. Kleinberg. *Networks, crowds, and markets*. Cambridge University Press, 2010.

[22] L. Epstein, A. Levin, A. Marchetti-Spaccamela, N. Megow, J. Mestre, M. Skutella, and L. Stougie. Universal sequencing on a single machine. *In Proc. of the 14th International Conference of Integer Programming and Combinatorial Optimization (IPCO)*, pages 230–243, 2010.

[23] E. Erel and J. B. Ghosh. FPTAS for half-products minimization with scheduling applications. *Discrete Applied Mathematics*, 156(15):3046–3056, 2008.

[24] G. Gamow and M. Stern. *Puzzle-math*. Macmillan, 1958.

[25] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[26] R. Graham, E. Lawler, J. Lenstra, and A. Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5(2):287–326, 1979.

[27] N. Hall, W. Kubiak, and S. Sethi. Earliness-tardiness scheduling problems, II: Deviation of completion times about a restrictive common due date. *Operations Research*, 39(5):847–856, 1991.

[28] P. E. Hart, N. J. Nilsson, and B. Raphael. Correction to a formal basis for the heuristic determination of minimum cost paths. *ACM SIGART Bulletin*, 37:28–29, 1972.

[29] W. Höhn. Scheduling (Dagstuhl Seminar 13111). *Dagstuhl Research Online Publication Server*, pages 32–33, 2013.

[30] W. Höhn and T. Jacobs. An experimental and analytical study of order constraints for single machine scheduling with quadratic cost. In *Proc. of the 14th Workshop on Algorithm Engineering and Experiments (ALENEX'12)*, pages 103–117, 2012.

[31] W. Höhn and T. Jacobs. Generalized min sum scheduling instance library. "http://www.coga.tu-berlin.de/v-menue/projekte/complex_scheduling/generalized_min-sum_scheduling_instance_library/", 2012.

[32] W. Höhn and T. Jacobs. On the performance of Smith's rule in single-machine scheduling with nonlinear cost. In *Proc. of the 10th Latin American Theoretical Informatics Symposium (LATIN)*, pages 482–493, 2012.

[33] J. Hoogeveen and S. Van de Velde. Scheduling around a small common due date. *European Journal of Operational Research*, 55(2):237–242, 1991.

[34] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM (JACM)*, 22(4):463–468, 1975.

[35] S. Irani and K. Pruhs. Algorithmic problems in power management. *ACM SIGACT News*, 36(2):63–76, 2005.

[36] B. Jurisch, W. Kubiak, and J. Jozefowska. Algorithms for minclique scheduling problems. *Discrete Applied Mathematics*, 72(1):115–139, 1997.

[37] I. Kacem. Fully polynomial time approximation scheme for the total weighted tardiness minimization with a common due date. *Discrete Applied Mathematics*, 158(9):1035–1040, 2010.

[38] H. Kaindl, G. Kainz, and K. Radda. Asymmetry in search. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 31(5):791–796, 2001.

[39] H. Kellerer and V. A. Strusevich. A fully polynomial approximation scheme for the single machine weighted total tardiness problem with a common due date. *Theoretical Computer Science*, 369(1):230–238, 2006.

[40] H. Kellerer and A. S. Vitaly. Minimizing total weighted earliness-tardiness on a single machine around a small common due date: An FPTAS using quadratic knapsack. *International Journal of Foundations of Computer Science*, 21(3):357–383, 2010.

[41] W. Kubiak. Completion time variance minimization on a single machine is difficult. *Operations Research Letters*, 14(1):49–59, 1993.

[42] E. L. Lawler and J. M. Moore. A functional equation and its application to resource allocation and sequencing problems. *Management Science*, 16(1):77–84, 1969.

[43] M. Li, A. C. Yao, and F. F. Yao. Discrete and continuous min-energy schedules for variable voltage processor. 103(11):3983–3987, 2006.

[44] N. Megow and J. Verschae. Dual techniques for scheduling on a machine with varying speed. In *Proc. of the 40th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 745–756, 2013.

[45] S. Mondal and A. Sen. An improved precedence rule for single machine sequencing problems with quadratic penalty. *European Journal of Operational Research*, 125(2):425–428, 2000.

[46] S. A. Mondal. Minimization of squared deviation of completion times about a common due date. *Computers & Operations Research*, 29(14):2073–2085, 2002.

[47] D. Monderer and L. Shapley. Potential games. *Games and Economic Behavior*, 14:124–143, 1996.

[48] H. Moulin and S. Shenker. Strategyproof sharing of submodular costs: budget balance versus efficiency. *Economic Theory*, 18(3):511–533, 2001.

[49] N. Nisan. *Algorithmic game theory*. Cambridge University Press, 2007.

[50] K. Pruhs, P. Uthaisombut, and G. Woeginger. Getting the best response for your erg. *ACM Transactions on Algorithms (TALG)*, 4(3):38, 2008.

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

[52] A. K. Sen, A. Bagchi, and R. Ramaswamy. Searching graphs with A*: applications to job sequencing. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 26(1):168–173, 1996.

[53] T. Sen, P. Dileepan, and B. Ruparel. Minimizing a generalized quadratic penalty function of job completion times: an improved branch-and-bound approach. *Engineering Costs and Production Economics*, 18(3):197–202, 1990.

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

[55] B. Srirangacharyulu and G. Srinivasan. An exact algorithm to minimize mean squared deviation of job completion times about a common due date. *European Journal of Operational Research*, 231(3):547–556, 2013.

[56] W. Szwarc. Decomposition in single-machine scheduling. *Annals of Operations Research*, 83:271–287, 1998.

[57] W. Townsend. The single machine problem with quadratic penalty function of completion times: a branch-and-bound solution. *Management Science*, 24(5):530–534, 1978.

[58] O. C. Vasquez. Energy in computing systems with speed scaling: optimization and mechanisms design. Technical Report arXiv:1212.6375, Arxiv.org, 2012.

[59] O. C. Vásquez. On the complexity of the single machine scheduling problem minimizing total weighted delay penalty. manuscript, 2013.

[60] J. Verschae. Personal communication. August 2013.

[61] X. Weng and J. A. Ventura. Scheduling about a given common due date to minimize mean squared deviation of completion times. *European Journal of Operational Research*, 88(2):328–335, 1996.

[62] G. J. Woeginger. Scheduling (Dagstuhl Seminar 10071). *Dagstuhl Research Online Publication Server*, page 24, 2010.

[63] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *Proc. of the 36th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 374–382, 1995.

[64] J. Yuan. The NP-hardness of the single machine common due date weighted tardiness problem. *System Science and Mathematical Sciences*, 5(4):328–333, 1992.