Flexibilité et Robustesse en Ordonnancement

GOThA, groupe flexibilité

Cet article est le fruit de deux années de réflexions et de partage d'expériences au sein du groupe de travail *flexibilité*, qui s'est régulièrement réuni durant les journées du Groupe de recherche sur l'Ordonnancement Théorique et Appliqué (GOThA). Les membres du groupe partagent une préoccupation commune, comment réagir aux aléas lors de la résolution d'un problème d'ordonnancement ; ils viennent cependant d'horizons applicatifs très différents. Un des objectifs du groupe est de proposer un cadre de référence qui soit acceptable par les différents intervenants concernés : problématique, définitions, mise en œuvre de la flexibilité.

1 Introduction

Souvent les données initiales nécessaires à la construction d'un ordonnancement sont incertaines avant la phase d'exécution (d'une production, d'un projet, d'un programme parallèle). Pourtant une première étude est possible (et souhaitable !) avant cette phase d'exécution. Que faire dans cette région un peu floue entre déterminisme et temps réel ? La solution consiste à introduire de la flexibilité dans le processus même de construction de l'ordonnancement.

Nous nous plaçons ici strictement dans la problématique de l'ordonnancement, même si la notion ou le désir de flexibilité peuvent dépendre d'autres niveaux de décision (par exemple, en ordonnancement de production, de la planification amont). La cause des aléas est peu évoquée ici. Ceci facilite la construction d'un cadre commun pour les différentes applications : pilotage d'atelier, gestion de projet, parallélisation d'applications informatiques.

Dans la suite, on considère un problème d'ordonnancement de tâches soumises à des contraintes sur les durées d'exécution, de communication (ou de transfert), des contraintes de précédence, des contraintes de ressources. La performance d'un ordonnancement peut être évaluée par un critère fixé. Le problème a une version **statique** où tout est supposé fixé, et une version **dynamique** où des aléas de natures diverses viennent perturber le problème statique. Dans ce dernier cas, il est nécessaire de prendre des décisions au moment même de l'exécution des tâches. Informellement, la **flexibilité** est

la liberté dont on dispose durant cette phase d'exécution. Nous utiliserons le terme de **robustesse** pour caractériser la performance d'un algorithme (ou plutôt d'un processus complet de construction d'un ordonnancement) en présence d'aléas.

A noter que les termes prédictif / réactif sont employés couramment en automatique à la place de statique / dynamique, que nous avons préférés par souci de généralité.

2 Flexibilité

Suivant le contexte, il est possible d'introduire la flexibilité de différentes manières. On peut l'exprimer comme l'existence de modifications possibles d'un ordonnancement calculé en statique, et entraînant une perte de performance acceptable. Une seconde manière (très proche) de l'exprimer est de définir pour un ordonnancement statique un voisinage de solutions pouvant être acceptées à l'exécution. Une troisième manière (voir section suivante) consiste à considérer que durant l'étape statique une famille d'ordonnancements est proposée, sans en privilégier forcément un seul.

On distingue plusieurs types (ou degrés) de flexibilité :

1. flexibilité sur les **temps**. Seules les dates effectives de début et de fin des tâches peuvent varier.
2. flexibilité sur les **ordres**. Les ordres relatifs d'exécution (séquences) d'un ensemble de tâches peuvent être modifiés durant l'exécution, ce qui implique la flexibilité sur les temps. Cet ensemble de tâches doit utiliser une même ressource qui oblige à les exécuter séquentiellement. Les ressources utilisées par chaque tâche restent donc inchangées.
3. flexibilité sur les **ressources**. Il est possible de changer l'affectation des ressources aux tâches. La flexibilité sur les ressources nécessite d'accepter la flexibilité sur les temps et les ordres.
4. flexibilité sur les **modes d'exécution** : suivant le contexte à l'exécution, on peut décider, au prix peut-être d'une dégradation de performances, de changer le mode d'exécution d'une tâche (ou plusieurs). En présence d'aléas, on permettra ainsi la préemption, la duplication, le recouvrement des exécutions par les communications, le changement de gamme, voire la délocalisation.

La flexibilité sur les temps est en quelque sorte le degré zéro de la flexibilité car indispensable dès qu'un aléa doit être pris en compte. En particulier les études de sensibilité existantes supposent toutes au moins cette flexibilité sur les temps. Fréquemment, elle est implicite dans la définition d'un ordonnancement.

Pour chaque type de flexibilité, il est utile de définir un outil quantitatif de mesure, un **indicateur de flexibilité**. Cette mesure doit porter sur un ensemble d'ordonnancements déterminés par un algorithme statique (voir section suivante). On pourra mesurer par exemple, outre la cardinalité de l'ensemble, la distance maximale entre deux ordonnancements en termes de différences de dates de fin, nombre d'échanges de tâches, nombre de changements de ressources, etc.

3 Prise en compte des aléas : mise en œuvre

Le processus global d'ordonnancement en présence d'aléas peut se décomposer comme suit :

étape 1 Définition du **problème statique**. Cette définition comprend, en plus des spécifications classiques en ordonnancement déterministe, la spécification des aléas possibles. La notion de **qualité** d'un ordonnancement doit être également précisée à ce niveau (critère de performance).

étape 2 Calcul d'un ensemble de solutions (famille d'ordonnancements réalisables) par un **algorithme statique** (ou algorithme déterministe) α.

étape 3 Lors de l'exécution, calcul d'une solution unique (l'ordonnancement réalisé) issue de cet ensemble par un **algorithme dynamique** δ.

Le processus complet nécessite donc le choix de deux algorithmes statique et dynamique. Fréquemment il n'est que partiellement explicité, l'un des deux algorithmes étant trivial.

Un problème voisin de celui de la mise en œuvre consiste à étudier la difficulté du problème (statique) de calcul d'un nouvel ordonnancement en supposant les aléas connus : est-il plus facile de partir du premier ordonnancement statique que de recalculer à partir du début un nouvel ordonnancement dans les nouvelles conditions ?
Dans l'affirmative, on parle de *post-optimisation* lorsqu'on recherche le nouvel optimum, ou de *réparation* si on cherche simplement une solution réalisable

(voir [CKS+00]). On peut également définir alors la stabilité d'une solution (est-elle plus ou moins loin d'une nouvelle solution optimale ?), voir la section suivante pour une définition plus formelle.

Outre un intérêt théorique (ce type d'études a été effectué pour des problèmes comme le voyageur de commerce ou l'arbre de poids min, mais pas en ordonnancement), cette problématique intervient naturellement quand on cherche à évaluer l'efficacité de l'algorithme dynamique de l'étape 3.

Les premiers résultats [Pic00] montrent que pour plusieurs problèmes NP-difficiles (problèmes à 1, 2 ou m machines avec minimisation du makespan), lorsqu'on introduit une perturbation même minimale (incrémentation ou décrémentation d'une date de disponibilité, d'échéance ou d'une durée d'exécution), le problème de post-optimisation est lui aussi NP-difficile.

4 Métriques pour la robustesse

L'étude de la robustesse du processus global d'ordonnancement est la recherche de garanties de performances sur l'ordonnancement réalisé.

Cette section propose des outils quantitatifs de mesure de la robustesse.

4.1 Notations

Les définitions suivantes sont utilisées. La performance d'un ordonnancement est évaluée selon un critère de performance quelconque Z.

- $\mathcal{P}$ un problème statique, avec la description des aléas potentiels (ce qui peut correspondre à une infinité d'instances du problème déterministe)
- Σ^{α} un ensemble de solutions (ordonnancements) obtenus par α, l'algorithme statique.
- S_{δ}^{α} un ordonnancement réalisé obtenu par (α, δ), l'algorithme statique et l'algorithme dynamique.
- $\mathcal{I}$ une instance réalisée de $\mathcal{P}$ (conditions effectives à l'exécution) appelée aussi scénario
- $z_{\mathcal{I}}(S)$ performance d'un ordonnancement S réalisé sur $\mathcal{I}$
- $z_{\mathcal{I}}^{*}$ performance d'un ordonnancement optimal sur $\mathcal{I}$

4.2 Types de robustesses et de métriques

Dans l'étude de la robustesse d'un processus global, α, δ, $\mathcal{P}$ sont fixés. La mesure de robustesse au pire du processus global est donnée par :

$$R_1 = \max_{\mathcal{I} \in \mathcal{P}} d(z^*_{\mathcal{I}}, z_{\mathcal{I}}(S^\alpha_\delta))$$

où d est une différence, une déviation, ou un rapport, entre deux performances. Un ordonnancement est d'autant plus robuste que sa mesure de robustesse R_1 est petite. La mesure de robustesse en moyenne suppose connue la distribution des différentes instances réalisées de $\mathcal{P}$, et s'exprime alors comme une espérance :

$$\bar{R}_1 = E_{\mathcal{I} \in \mathcal{P}} \left(d(z^*_{\mathcal{I}}, z_{\mathcal{I}}(S^\alpha_\delta)) \right)$$

La robustesse "au pire" est une garantie absolue de performance, alors que la robustesse en moyenne ne garantit pas de performance pour chaque réalisation. La définition ci-dessus est très générale (quels que soient α et δ).

On peut aussi limiter l'étude à la robustesse de l'algorithme statique α. Dans ce cas on recherche une garantie de performance vérifiée pour tout algorithme dynamique (ou pour une famille raisonnable d'algorithmes dynamiques). La mesure de la robustesse au pire devient alors :

$$R_2 = \max_{\mathcal{I} \in \mathcal{P}} \max_{S \in \Sigma^\alpha} d\left(z^*_{\mathcal{I}}, z_{\mathcal{I}}(S)\right)$$

La mesure de robustesse en moyenne $\bar{R}_2$ peut être adaptée de la même manière.

D'autres métriques proposées sont :

- la distance à une performance prévue (négociée) $\tilde{z}(\mathcal{P})$. La mesure de la robustesse s'écrit :

$$R_3 = \max_{\mathcal{I} \in \mathcal{P}} d(\tilde{z}(\mathcal{P}), z_{\mathcal{I}}(S^\alpha_\delta))$$

- la mesure du niveau de service. Il s'agit de maximiser la probabilité que z soit en-dessous d'un certain seuil fixé $T(\mathcal{P})$:

$$R_4 = P(z(S^\alpha_\delta) \leq T(\mathcal{P}))$$

– la mesure de la stabilité d'un ordonnancement statique. On cherche à minimiser l'écart entre un ordonnancement privilégié calculé en statique $\tilde{S} \in \Sigma^\alpha$, et celui réalisé par δ, S_δ^α :

$$R_5 = \max_{\mathcal{I} \in \mathcal{P}} d'(\tilde{S}, S_\delta^\alpha)$$

d' mesure l'écart entre deux ordonnancements (similarité), elle est donc utilisable pour la mesure de flexibilité introduite section 2.
Dans le cas particulier où $d' \in \{0, 1\}$ on retrouve la notion de stabilité d'un ordonnancement issue de l'automatique : un ordonnancement reste stable s'il est inchangé en présence des aléas. Enfin si la distance de similarité calcule un écart entre les performances des deux ordonnancements, on retrouve la notion de stabilité de l'algorithme ou du processus global (voir références [SSW97, SWW98]) :

$$R_6 = \max_{\mathcal{I} \in \mathcal{P}} d(z_\mathcal{I}(\tilde{S}), z_\mathcal{I}(S_\delta^\alpha))$$

Dans ce sens, une étude de stabilité s'inscrit donc dans le cadre plus général de l'analyse de sensibilité, historiquement issue de la programmation linéaire : quelle est la dégradation de performance d'une solution quand les conditions initiales varient ? Les études récentes de sensibilité en ordonnancement sont assez nombreuses (voir par exemple [HP01]). Cela revient dans notre cadre à n'accepter qu'une flexibilité temporelle, l'algorithme dynamique se contentant de décaler les dates d'exécution des tâches.

5 Exemples de problèmes traités par des équipes du groupe

5.1 Gestion des retards dans l'approvisionnement d'une ligne d'assemblage

Dans l'industrie automobile, la plupart des sous-traitants sont installés à proximité des usines d'assemblage. Ils délivrent les pièces prêtes à monter plusieurs fois par jour selon les directives de l'usine d'assemblage (basées sur la *car sequence*). Malheureusement, il arrive que les sous-traitants ne puissent livrer à temps.

On se propose alors de calculer un ordonnancement des pièces à assembler sachant que les dates de début au plus tôt sont sujettes à modification. L'objectif ici est la minimisation du nombre pondéré de tâches en retard.

Une approche par algorithme génétique est proposée. A chaque itération, on évalue le critère pour une série d'instances dont les dates de disponibilité ont été modifiées aléatoirement. Le résultat est un ordonnancement plus robuste qui peut absorber des variations des dates de disponibilité (voir [SÖ1, SS02]).

5.2 Recherche de solutions robustes et flexibles pour la gestion d'un parc de machines polyvalentes

Nous considérons un atelier de photolithographie pour la fabrication de semi-conducteurs. Cet atelier est composé de machines polyvalentes. Ces machines doivent être configurées avant la phase de production, ce qui a un coût que l'on souhaite minimiser. Le second objectif est la minimisation du temps total de production. Pour ce type d'atelier, le volume de production demandé varie beaucoup au cours du temps car les perturbations issues des ateliers en amont sont amplifiées par l'aspect réentrant de cet atelier.

Nous nous intéressons ici à l'élaboration de solutions dites robustes, c'est-à-dire qui garantissent un niveau de performance pour un ensemble de jeux de données possibles. L'exigence sur l'optimalité de la solution est alors relâchée. On a modélisé le problème de répartition des tâches sur les machines et de la configuration de ces machines sous la forme d'un programme linéaire à variables mixtes bi-critères. Des résultats de complexité ont été démontrés, une méthode exacte pour déterminer une configuration robuste est en cours de développement, ainsi que des heuristiques (voir [RJ01a, RJ01b]).

5.3 Gestion des indisponibilités des machines dans des ateliers de production

Les problèmes classiques d'ordonnancement considèrent que les machines sont toujours disponibles. Or, dans l'industrie ceci n'est que très rarement le cas. Les machines peuvent connaître des périodes d'indisponibilité, ces périodes correspondant par exemple à des périodes de pannes, à des périodes bloquées pour pouvoir réaliser des commandes urgentes ou encore à des périodes de maintenance préventive. Plusieurs chercheurs du groupe flexi-

bilité se sont intéressés à la prise en compte de ces périodes d'indisponibilités en ordonnancement dans des ateliers à une machine, m machines ou encore dans des ateliers de type flowshop (voir par exemple [EFP99]), openshop et jobshop. Plusieurs résultats de complexité ont été démontrés, des méthodes de résolutions exactes ou approchées ont été proposées. Mais la plupart de ces travaux (voir les revues [LLP97, SS98]) considèrent que les caractéristiques (durée, date de début) de ces périodes sont parfaitement connues en début d'ordonnancement. Or, par exemple, lorsque ces périodes d'indisponibilité correspondent à des périodes de maintenance préventive, souvent seule une durée approximative peut être donnée. Notre objectif est donc d'étudier la prise en compte d'aléas tels qu'une période d'indisponibilité pouvant être plus longue que prévue et ainsi étudier la robustesse des algorithmes proposés auparavant (voir aussi le travail de [GN99] sur le flowshop).

5.4 Gestion en temps réel d'une machine soumise à des aléas

Nous considérons ici des ateliers composés d'une seule machine. Les critères de performance sont la durée totale d'une production et la somme des retards pondérés suivant l'importance de chaque produit fini.

Les aléas pris en compte sont à la fois les retards de livraison (donc retard de la date de disponibilité), les retards lors de l'exécution d'une tâche, et les pannes de la machine (voir [AP01]).

Au lieu de donner un unique ordonnancement à l'atelier, nous fournissons un ensemble d'ordonnancements qui suivent une structure donnée afin de passer d'un ordonnancement à un autre facilement. La structure choisie est un graphe partiel non cyclique contenant les précédences impératives ainsi que les précédences rajoutées par l'ordonnancement statique. La flexibilité choisie est donc une flexibilité sur les ordres. Elle est mesurée par le nombre d'ordonnancements respectant ce graphe. Les ordonnancements proposés sont semi-actifs au sens où une machine peut rester oisive en attendant une opération prioritaire. Chaque fois qu'une décision doit être prise, suite à l'arrivée en temps réel d'un aléa quelconque, un ensemble d'alternatives compatibles avec les contraintes du problème est donné au décideur. Celui-ci peut choisir l'action qui lui convient selon ses préférences, l'état de l'atelier et éventuellement des contraintes non modélisées (voir [AP02]).

5.5 Gestion de l'incertitude sur les temps de communication entre ordinateurs en réseaux

L'utilisation efficace des ordinateurs parallèles ou des réseaux d'ordinateurs nécessite de découper les applications en modules exécutés sur différents processeurs. Ces modules sont liés par des contraintes de précédence. Mais leur temps d'exécution ainsi que les durées de communication entre ces modules sont très difficiles à évaluer exactement. En particulier les durées de communication dépendent à la fois : des modules qui communiquent, des processeurs qui les exécutent, mais aussi de l'état présent du réseau de communication.

L'objectif consiste à minimiser la durée totale d'une application. Pour des modèles simples (2 machines, nombre illimité de machines,...) nous menons des études de sensibilité d'algorithmes existants optimaux dans le cas statique (voir [GMS02]). Pour des modèles plus complexes, nous proposons de conserver une flexibilité sur les ordres lors de l'exécution (changer l'affectation des tâches au moment de l'exécution est en effet souvent impossible, et en tout cas trop coûteux). Par rapport à une approche purement en ligne, nous ajoutons des contraintes issues de l'étude statique. L'algorithme dynamique choisit donc la tâche à exécuter parmi celles disponibles en respectant ces contraintes additionnelles (voir [MSG99]).

5.6 Recherche et maximisation d'un niveau de service en ordonnancement stochastique

Nous nous intéressons aux problèmes d'ordonnancement dans lesquels les durées opératoires sont connues de manière probabiliste, à travers par exemple une densité de probabilité (continue ou discrète). La plupart des travaux dans ce contexte ont pour objectif de minimiser la valeur moyenne (espérance mathématique) d'un critère classique, le makespan habituellement.

Nous trouvons plus pertinent de considérer un niveau de service, comme en gestion des stocks par exemple. En particulier, nous considérons la probabilité que le makespan soit inférieur ou égal à une valeur fixée (par exemple la durée d'une période pendant laquelle les jobs doivent être terminés). On peut facilement montrer sur un exemple qu'un ordonnancement $O1$ peut dominer un autre ordonnancement $O2$ sur la valeur moyenne du makespan, alors que $O1$ aura un moins bon niveau de service que $O2$.

Le premier problème, déjà complexe, consiste à calculer le niveau de service pour un ordonnancement donné (exemple : quelle est la probabilité que le makespan soit inférieur ou égal à 100 ?). Dans un deuxième temps, l'objectif serait de déterminer l'ordonnancement qui maximise le niveau de service (voir références [Iid00, SG00]).

5.7 Insertion d'une tâche imprévue en gestion de projets sous contraintes de ressources

Le problème de détermination des dates de début des tâches d'un projet en respectant les contraintes de succession entre tâches et en assurant une durée de réalisation minimale du projet devient très difficile lorsque les tâches utilisent des ressources disponibles en quantités limitées (contraintes cumulatives). De nombreuses méthodes exactes et heuristiques ont été développées pour résoudre dans un contexte statique ce problème, connu sous le nom de RCPSP (*Resource-Constrained Project Scheduling Problem*). Nous considérons le problème de la mise en œuvre d'une solution précalculée (à l'aide d'un algorithme statique α quelconque) dans un contexte dynamique caractérisé par la possibilité d'arrivée en cours de réalisation d'une tâche imprévue au départ (voir [ARB99]).

Dans le but de proposer un algorithme dynamique rapide et de ne pas trop perturber la solution précalculée, nous utilisons dans [AMR02] un modèle de flots pour représenter cette solution et nous imposons à l'algorithme dynamique de conserver dans un premier temps le séquencement relatif des tâches déjà présentes. Sous cette dernière contrainte, nous pouvons insérer la tâche en un temps polynomial tout en minimisant la durée totale. Dans un second temps, l'algorithme applique une méthode sérielle classique pour rendre actif l'ordonnancement obtenu dans la première phase.

Une étude expérimentale effectuée sur 600 instances à 120 tâches et 4 ressources a visé à comparer notre algorithme (δ) avec un algorithme de réordonnancement classique (α) de même complexité temporelle et basé sur une règle de priorité. Pour chacune des instances, 27 insertions d'une tâche générée aléatoirement ont été réalisées, la solution initiale étant obtenue par α. La comparaison des performances $z_{\mathcal{I}}(S^{\alpha}_{\alpha})$ et $z_{\mathcal{I}}(S^{\alpha}_{\delta})$ pour chaque insertion montre que pour ces instances l'insertion par δ est généralement plus robuste que le réordonnancement par α.

6 Extensions et Conclusions

Notre effort a porté sur l'élaboration d'un cadre de références à l'intérieur duquel chacun puisse insérer sa propre problématique. Mais aussi large qu'il soit, le cadre proposé exclut, ou s'adapte mal à, certains problèmes intéressants. C'est le cas de l'ordonnancement cyclique, non traité ici, ou des perturbations sur le critère lui-même.

6.1 Perturbations sur le critère de performances

Dans certaines applications, c'est la façon de mesurer la performance d'un ordonnancement qui peut changer à l'exécution. C'est en particulier le cas pour des critères associant aux tâches des pondérations (somme pondérée des dates de fin, des retards, ...). Ceci n'introduit pas nécessairement une flexibilité sur l'ordonnancement, qui peut rester figé. Il est néanmoins possible de mesurer la robustesse d'un ordonnancement de manière similaire, si on introduit des hypothèses sur la perturbation possible de Z : robustesse au pire et en moyenne, seuil de performance et niveau de service, mesure de stabilité ...

On peut cependant concevoir que l'on puisse à l'exécution modifier l'ordonnancement pour prendre en compte ces modifications de Z. On retrouve alors la nécessité de fixer le degré de flexibilité acceptable.

6.2 Approche commune à d'autres secteurs de la R. O.

Bien sûr, l'étude de la flexibilité et de la robustesse n'est pas limitée au seul domaine de l'ordonnancement. Bien des domaines de la Recherche Opérationnelle sont concernés, principalement ceux incluant une dimension temporelle comme la planification ou la logistique (par exemple les problèmes de tournées de véhicules, allocation dynamique de fréquences). Il sera donc intéressant dans l'avenir de confronter les différentes approches.

Références

[AMR02] C. Artigues, P. Michelon, and S. Reusser. Insertion techniques for static and dynamic resource constrained project scheduling. *European Journal of Operational Research*, 2002. to appear.

[AP01] M.A. Aloulou and M.C. Portmann. Incorporating flexibility in job sequencing for the single machine total weighted tardiness problem with release dates. In *10th Annual Industrial Engineering Research Conference*. IIE, May 2001.

[AP02] M.A. Aloulou and M.C. Portmann. A genetic algorithm to achieve scheduling flexibility for a single machine problem. *RAIRO-Operations Research*, 2002. soumis, disponible sur http ://www.loria.fr/∼aloulou/papiers/rairo.ps.gz.

[ARB99] C. Artigues, F. Roubellat, and J.-C. Billaut. Characterization of a set of schedules in a resource constrained multi-project scheduling problem with multiple modes. *International Journal of Industrial Engineering, Applications and Practice*, 6 :112–122, 1999.

[BBMN91] J.C. Bean, J.R. Birge, J. Mittenthal, and C.E. Noon. Match-up scheduling with multiple ressources, release dates and disruptions. *Operations Research*, pages 470–483, 1991.

[BR96] J.C. Billaut and F. Roubellat. A new method for workshop real-time scheduling. *International Journal of Production Research*, 34(6) :1555–1579, june 1996.

[CKS+00] S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rapideau. Using iterative repair to improve the responsiveness of planning and scheduling. In *AIPS, Breckenridge, CO, USA*, 2000.

[DB00] A.J. Davenport and J.C. Beck. A survey of techniques for scheduling with uncertainty. **Une grosse revue de la flexibilité en ordonnancement**, disponible sur http ://www.eil.utoronto.ca/profiles/chris/chris.papers.html, 2000.

[EFP99] M.L. Espinouse, P. Formanowicz, and B. Penz. Minimizing the makespan in the two-machine no-wait flow-shop with limited machine availability. rapport de recherche GILCO-RR-99-3, INPG, 1999.

[GMS02] F. Guinand, A. Moukrim, and E. Sanlaville. Sensitivity analysis of tree scheduling on two machines with communication delays. In *8th International Workshop on Project Management and Scheduling, PMS'2002*, pages 179–182, Valencia, Spain, april 2002. ISBN 84-921190-5-5.

[GN99] B. Guo and Y. Nonaka. Rescheduling and optimization of schedules considering machine failures. *IJPE*, 60-61 :503–513, 1999.

[HP01] N.G. Hall and M.E. Posner. Sensitivity analysis for scheduling problems. **Revue sur la sensibilité en ordonnancement**, disponible sur http ://www-iwse.eng.ohio-state.edu/ISECourses/ise824, scheduling class papers, 2001.

[Iid00] T. Iida. Computing bounds on project duration distributions for stochastic pert networks. *Naval Research Logistics*, 47 :559–580, 2000.

[KDV00] P. Kouvelis, R.L. Daniels, and G. Vairaktarakis. Robust scheduling of a two-machine flow shop with uncertain processing times. *IIE Transactions*, 32 :421–432, 2000.

[KHB02] I. Kacem, S. Hammadi, and P. Borne. Approach by localization and multi-objective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics, PART C*, 32, 2002.

[KY97] P. Kouvelis and G. Yu. *Robust Discrete Optimisation and its Applications*. Kluwer Academic Publisher, 1997. **Livre traitant de la flexibilité en recherche opérationnelle**.

[LLP97] C.Y. Lee, L. Lei, and M. Pinedo. Current trends in deterministic scheduling. *Ann Oper Res*, 70 :1–42, 1997.

[LWS94] V.J. Leon, S.D. Wu, and R.H. Storer. Robustness measures and robust scheduling for job-shops. *IIE Transactions*, 26(5) :32–43, 1994.

[MPR98] N.V.R. Mahadev, A. Pekec, and F.S. Roberts. On the meaningfulness of optimal solutions to scheduling problems : can an optimal solution be nonoptimal ? *Operations Research*, 46(sup 3), 1998.

[MSG99] A. Moukrim, E. Sanlaville, and R. Guinand. Scheduling with communication delays and on-line disturbances. In P. Amestoy, editor, *Euro-Par'99, Toulouse, France*, volume LNCS 1685, pages 350–357, 1999.

[MU99] S.V. Mehta and R. Uzsoy. Predictable scheduling of a single machine subject to breakdowns. *International Journal of Computer Integrated Manufacturing*, 12 :15–38, 1999.

[Pic00] C. Picouleau. Small perturbations on some np-complete scheduling problems. CEDRIC Research report 09, CNAM Paris, 2000.

[RJ01a] A. Rossi and M. Jacomino. Performance guarantee for static and dynamic approaches of scheduling problems in uncertain contexts. In *CIRP, Michigan University*, May 2001.

[RJ01b] A. Rossi and M. Jacomino. Robustness and flexibility as performance guarantee for scheduling problems in uncertain contexts. In *3rd World Manufacturing Congress, Rochester*, September 2001.

[SÖ1] K. Sörensen. Tabu searching for robust solutions. In *4th Metaheuristics International Conference*, pages 707–712, Porto, Portugal, July 16-20 2001.

[SG00] C. W. Schmidt and I. E. Grossmann. The exact overall time distribution of a project with uncertain task durations. *EJOR*, 126 :614–636, 2000.

[SS98] E. Sanlaville and G. Schmidt. Machine scheduling with availability constraints. *Acta Informatica*, 35 :795–811, 1998.

[SS02] M. Sevaux and K. Sörensen. Genetic algorithm for robust schedules. In *8th International Workshop on Project Management and Scheduling, PMS'2002*, pages 330–333, Valencia, Spain, April, 3-5 2002. ISBN 84-921190-5-5.

[SSNB95] J.A. Stankovic, M. Spuri, M. Di Natale, and G.C. Buttazzo. Implications of classical scheduling results for real-time systems. *Computer*, 28(6) :16–25, 1995.

[SSW97] Y.N. Sotskov, N.Y. Sotskova, and F. Werner. Stability of an optimal schedule in a job-shop. *IJMS*, 25(4) :397–414, 1997.

[SWW98] Y.N. Sotskov, A.P.M. Wagelmans, and F. Werner. On the calculation of the stability radius of an optimal or an approximate schedule. *Annals of Operational Research*, 83 :213–225, 1998.

[Vin99] P. Vincke. Robust and neutral methods for aggregating preferences into an outranking relation. *European Journal of Operational Research*, 122 :405–412, 1999.

[WBS99] S.D. Wu, E.S. Byeon, and R.H. Storer. A graph-theoretic decomposition of the job-shop scheduling problem to achieve scheduling robustness. *Operations Research*, 47(1) :113–124, 1999.

Liste des membres du groupe flexibilité du GOThA

Amine MAHJOUB, gilco Grenoble, <mahjoub@gilco.inpg.fr>,
André ROSSI, lag Grenoble, <rossi@lag.ensieg.inpg.fr>,
Antoine JOUGLET, heudiasyc Compiègne, <antoine.jouglet@hds.utc.fr>,
Antony VIGNIER, loria Nancy, <avignier@loria.fr>,
Aziz MOUKRIM, heudiasyc Compiègne, <moukrim@utc.fr>,
Bernard PENZ, gilco Grenoble, <Bernard.Penz@gilco.inpg.fr>,
Carl ESSWEIN, li Tours, <carl.esswein@etu.univ-tours.fr>,
Christian ARTIGUES, lia Avignon,
<christian.artigues@lia.univ-avignon.fr>,
Christophe PICOULEAU, lip6 Paris, <christophe.picouleau@cnam.fr>,
Christophe THOMAS, gilco Grenoble, <Christophe.THOMAS@gilco.inpg.fr>,
Claude YUGMA, gilco Grenoble, <yugma@gilco.inpg.fr>,
Cyril BRIAND, laas Toulouse, <briand@laas.fr>,
Denis TRYSTRAM, id imag Grenoble, <Denis.Trystram@imag.fr>,
Emmanuel NERON, li Tours, <neron@univ-tours.fr>
Emmanuel PODER, limos Clermont, <poder@math.univ-bpclermont.fr>,
Eric SANLAVILLE (coordinateur), limos Clermont,
<Eric.Sanlaville@math.univ-bpclermont.fr>
Francis SOURD, lip6 Paris, <Francis.Sourd@lip6.fr>,
Frédéric GUINAND, lih Le Havre,<guinand@ecrins.univ-lehavre.fr>,
Imed KACEM, lail Lille, <imed.Kacem@ec-lille.fr>,
Jean-Charles BILLAUT, li Tours, <billaut@rabelais.univ-tours.fr>,
Julien BIDOT, ilog Paris, <jbidot@ilog.fr>,
Marc SEVAUX, lamih Valenciennes,
<Marc.Sevaux@univ-valenciennes.fr>,
Marie-Claude PORTMANN, loria Nancy, <portmann@mines.u-nancy.fr>,
Marie-Laure ESPINOUSE, losi Troyes,
<Marie_Laure.Espinouse@univ-troyes.fr>,
Mireille JACOMINO, lag Grenoble, <jacomino@lag.ensieg.inpg.fr>,
Mohamed Ali ALOULOU, loria Nancy, <mohamed-ali.aloulou@loria.fr>,
Pierre LOPEZ, laas Toulouse, <lopez@laas.fr>,
Safia KEDAD-SIDHOUM, lip6 Paris, <Safia.Kedad-Sidhoum@lip6.fr>,
Stéphane DAUZERE-PERES, irccyn Nantes, <dauze@emn.fr>,
Yann LE QUERE, lamih Valenciennes et sncf,
<yann.lequere@univ-valenciennes.fr>