Algorithmes d'ordonnancements bicritères en-lignes (Résumé)

Fabien Baille, Evripidis Bampis, Christian Laforest, Nicolas Thibault

LaMI, CNRS / Université d'Evry, Tour Evry 2, 523 place des terrasses, 91000 EVRY France

Nous proposons dans cet article des algorithmes d'ordonnancement permettant à un opérateur d'un lien haut-débit de gérer des demandes de réservations provenant de ses clients. Nos propositions ont les avantages suivants : (1) elles permettent de gérer les demandes *à la volée*, dès qu'elles arrivent à l'opérateur ; (2) nous prouvons qu'elles permettent d'approcher à de faibles facteurs multiplicatifs près, le profit maximum atteignable et, simultanément, le nombre maximum de clients pouvant être servis ; (3) ces facteurs sont ajustables et permettent à l'opérateur de donner plus d'importance relative à l'un ou l'autre des critères.

Keywords: Ordonnancement, algorithme en-ligne, algorithme bicritère

1 Introduction

Dans cet article nous étudions la situation suivante. Un lien réseau, composé de sous-canaux identiques, est géré par un *opérateur*. On peut considérer par exemple un lien transatlantique haut débit (fibre optique ou satellite). Un *client* peut demander à utiliser un sous-canal entre deux dates données. Il fait donc une demande de réservation exclusive de ressources (sous-canal) pendant une période choisie. Il peut par exemple demander un canal entre 13h00 et 15h00. Une demande acceptée peut se traduire par la mise à disposition exclusive de ce client du sous-canal numéro 12 entre 13h00 et 15h00.

Chaque demande, si elle est satisfaite, engendre un *profit* pour l'opérateur. Ce dernier doit bien sûr maximiser son profit (bénéfice à court terme). Cependant, il doit aussi veiller à accepter le plus de demandes possibles pour maximiser la satisfaction globale des clients (notion de bien être social, éventuellement imposé par une autorité de régulation). Étant donné que les ressources sont limitées, toutes les demandes ne peuvent (en général) pas être satisfaites. L'opérateur doit alors tenter de maximiser ces *deux* critères (profit et nombre de demandes satisfaites) *simultanément*.

Les (nombreux) travaux de la littérature traitant ce type de problèmes ont en général les caractéristiques suivantes :

- les auteurs considèrent que l'opérateur connaît la totalité des demandes des clients *à l'avance*. Cela peut s'appliquer par exemple à des demandes de réservations de connexions d'un jour sur l'autre : les clients font des réservations la veille pour le lendemain. Cependant, avec de telles méthodes (de type *hors-ligne*), l'opérateur ne peut pas traiter les données à la volée, dès qu'elles arrivent, pour satisfaire (ou pas) immédiatement la demande du client.
- Les auteurs traitent *séparément* le problème de maximisation du profit et celui de maximisation du nombre de clients servis. On notera que les problèmes algorithmiques sous-jacents étant souvent NP-complets, des algorithmes d'approximation sont souvent proposés. Cependant, une solution viable maximisant (ou approchant) les deux critères à la fois n'est en général pas proposée.

Dans cet article nous levons ces deux contraintes majeures, en proposant un algorithme *en-ligne* (traitant les demandes à la volée) et bicritère (i.e. *garantissant* que les deux critères sont approchés simultanément).

Dans ce qui suit, nous décrivons notre modélisation de la situation initiale grâce à des ordonnancements. Nous décrivons dans la section 2 notre algorithme en-ligne bicritère et nous donnons ses garanties. Nous

explorons ensuite dans la section 3 des généralisations de la méthode de base qui permettent, entre autres, de prendre en charge des demandes de réservation dites *dégradables*.

Notons que la place limitée nous interdit de donner tous les détails et toutes les preuves de nos résultats dans cet article.

Un intervalle. Une machine. Nous représentons la situation décrite en préambule par le modèle d'ordonnancement suivant. Notre système (i.e. le lien réseau) est composé de *k machines identiques* (i.e. les k sous-canaux). Un *intervalle* (i.e. une demande d'un client) σ_i est la donnée des paramètres suivants.

Une *date d'arrivée* $r_i \geq 0$ avant laquelle σ_i ne peut pas être ordonnancé (date à laquelle la demande du client arrive et à partir de laquelle la réservation doit débuter). Une *date de fin* d_i à laquelle σ_i doit se terminer. Un *poids* $w_i = d_i - r_i$, qui est ici égal à la durée d'exécution de l'intervalle.

Modèle d'exécution des intervalles sur les machines. Un intervalle σ_i est dit *ordonnancé* lorsqu'il est exécuté sans interruption, entre les dates r_i et d_i (i.e. pendant une durée d'exactement $p_i = d_i - r_i$).

La traduction en termes de réseaux est la suivante. Si la demande σ_i est acceptée par l'opérateur, un sous-canal est alors exclusivement dédié à (ou loué par) ce client à partir de la date r_i jusqu'à la date d_i. Cette opération génère alors un profit de (ou fonction de) $w_i = d_i - r_i$ pour l'opérateur (même si cela coûte en pratique plus cher au client). Un tel poids, proportionnel à la durée de l'utilisation de la ressource, correspond donc au profit (ou à une partie du profit) de l'opérateur si la demande est satisfaite.

Notons que la préemption n'est pas autorisée dans la mesure où les clients demandent l'utilisation exclusive d'un sous-canal. Notons aussi que l'opérateur peut interrompre un intervalle avant que celui-ci termine son exécution. Dans ce cas, l'intervalle interrompu ne rapporte rien pour les deux critères considérés et il est définitivement perdu.

Ordonnancement. Poids et taille d'un ordonnancement. Un *ordonnancement* O d'un ensemble d'intervalles σ est un sous-ensemble d'intervalles de σ exécutés de telle façon que chaque machine n'exécute qu'au plus un intervalle à chaque instant et que chaque intervalle de O est exécuté sur une seule machine. Par abus de notations, nous confondrons un ordonnancement O avec l'ensemble des intervalles qu'il ordonnance. Nous noterons $W(O) = \sum_{\sigma_i \in O} w_i$ le *poids* d'un ordonnancement O et $N(O) = |\{\sigma_i \in O\}|$ sa *taille*.

Ordonnancements optimaux. Soit σ un ensemble d'intervalles. Nous noterons $W^*(\sigma) = \max\{W(O) : O \text{ ordonnancement de } \sigma\}$ le *poids optimal* de σ et $N^*(\sigma) = \max\{N(O) : O \text{ ordonnancement de } \sigma\}$ la *taille optimale* de σ.

Les algorithmes en-lignes. Dans cet article nous supposons que les intervalles arrivent au fur et à mesure du temps et que l'opérateur *ne connaît pas* à l'avance les intervalles à traiter. Nous supposerons qu'il reçoit ces intervalles un par un, *dans l'ordre des dates de début* ($r_1 \leq r_2 \leq \ldots \leq r_i \leq \ldots$), et qu'une fois *révélé*, un intervalle σ_j doit être traité par l'algorithme de l'opérateur. Pour cela, il peut soit *rejeter* σ_j soit *accepter* σ_j et l'ordonnancer sur une des k machines.

Pour préciser les choses, nous noterons O_{j-1} l'ordonnancement courant avant la révélation de σ_j. Dans ce contexte, tout algorithme d'ordonnancement en-ligne supprime/*interrompt* dans un premier temps un certain ensemble S (éventuellement vide) d'intervalles déjà ordonnancés dans O_{j-1} dans sa *phase de suppression*. Il exécute ensuite sa *phase d'ordonnancement* qui est réduite à deux possibilités :

- Il ordonnance σ_j sur une machine libre. Dans ce cas, le nouvel ordonnancement O_j est $O_j = (O_{j-1} - S) \cup \{\sigma_j\}$.
- Il *rejette* σ_j. Dans ce cas, le nouvel ordonnancement O_j est $O_j = O_{j-1} - S$.

Il est important de noter ici que si un intervalle σ_l a été accepté à un moment donné et qu'il est interrompu ensuite (avant sa date de terminaison d_l) alors aucun bénéfice n'est tiré de lui (ni pour la taille ni pour le poids) dans les futurs ordonnancements (un intervalle rejeté ou interrompu est définitivement perdu).

La mesure de qualité d'un algorithme en-ligne A d'ordonnancement pour un critère C (pour nous $C = W$ ou $C = N$) est connu sous le nom de *rapport de compétitivité* (voir [BEY98, FW98]). A est ρ-compétitif si

pour toute suite d'intervalles révélée de manière en-ligne, $\sigma_1,\ldots,\sigma_i$

$$\rho C(A(\sigma_1,\ldots,\sigma_i) \geq C^*(\{\sigma_1,\ldots,\sigma_i\})$$

(avec $A(\sigma_1,\ldots,\sigma_i)$ l'ordonnancement construit par A lorsque σ_i a été traité).

On trouvera dans [FN95] un algorithme en-ligne, nommé *GOL*, qui résout le problème de la taille de manière optimale, c'est-à-dire avec un rapport de compétitivité de 1. A partir des travaux de [BNCK$^+$99] on peut tirer un algorithme en-ligne, nommé *LR*, nécessitant au moins $k \geq 2$ machines pour son exécution et qui a un rapport de compétitivité de $\frac{2}{1-\frac{2}{k}}$ pour le problème du poids.

2 Notre algorithme bicritère en-ligne

Un algorithme est dit bicritère en-ligne s'il est simultanément α_N-compétitif pour la taille et α_W-compétitif pour le poids. Nous avons déjà proposé dans [BBL03, BBL04b] un algorithme (d'approximation) bicritère pour ce type de mesure. Il nécessite cependant la connaissance préalable totale des intervalles à ordonnancer. Le problème traité ici est donc de *nature* très différente puisque les traitements doivent se faire au "fil de l'eau", dès qu'un intervalle est révélé, sans connaître les intervalles qui seront proposés dans l'avenir. De plus, un intervalle rejeté ou interrompu est définitivement perdu.

Cette section a pour objectif de donner les grandes lignes de notre méthode. L'idée générale est la suivante : on a un système à k machines dites "réelles", sur lesquelles les ordonnancements successifs doivent être produits. Nous allons contrôler la taille de nos ordonnancements successifs en "simulant" GOL et leurs poids en "simulant" LR. Ces deux simulations vont se faire sur des machines que nous qualifions de "virtuelles" car les ordonnancements produits sur celles-ci ne vont servir qu'à prendre des décisions pour les ordonnancements sur les machines réelles. Un paramètre r ($1 \leq r \leq k-3$) choisi au début par l'opérateur va permettre d'attribuer r machines "virtuelles" à GOL et $k-r$ à LR.

Chaque nouvel intervalle révélé σ_i est soumis à GOL^r (i.e. GOL appliqué sur r machines) et à LR^{k-r} (i.e. LR appliqué sur $k-r$ machines), qui traitent chacun et indépendamment σ_i. Comme il a été dit plus haut, GOL^r (resp. LR^{k-r}) fait une phase de suppression qui va éliminer un ensemble S_{GOL} (resp. S_{LR}) (potentiellement vide) des intervalles déjà ordonnancés sur les r (resp. $k-r$) machines virtuelles, avant l'arrivée de σ_i. Ensuite, GOL^r (resp. LR^{k-r}) décide d'ordonnancer ou pas σ_i sur une de ses k machines virtuelles libres.

En fonction des décisions de GOL^r et LR^{k-r}, notre algorithme va faire les opérations suivantes :

- Il va supprimer de ses k machines réelles les intervalles qui n'apparaissent plus ni dans l'ordonnancement courant de GOL^r ni dans celui de LR^{k-r}. NB : si un intervalle a été supprimé par GOL^r par exemple mais qu'il est encore présent dans l'ordonnancement courant de LR^{k-r}, il n'est pas supprimé de l'ordonnancement des machines réelles.
- Si au moins un des deux algorithmes GOL^r ou LR^{k-r} ordonnance σ_i alors notre algorithme fait de même.

On peut montrer qu'en faisant ainsi, à chaque étape, si σ_i est ordonnancé par GOL^r ou LR^{k-r} alors il existe une machine réelle libre pour l'ordonnancer aussi. On montre de même qu'à chaque étape, l'ensemble des intervalles ordonnancés sur les machines réelles est égal à l'union des intervalles ordonnancés par GOL^r et LR^{k-r}.

Grâce à cette propriété, le comportement et les garanties offertes par les deux algorithmes sont conservés. Seul le fait d'utiliser r (resp. $k-r$) machines pour GOL^r (resp. LR^{k-r}) au lieu de k machines "dégrade" son rapport de compétitivité. Tout cela peut être contrôlé analytiquement et conduit au résultat suivant.

Théorème 1 *Notre algorithme appliqué avec le paramètre r est $\frac{k}{r}$ compétitif pour la taille et $\frac{2}{1-\frac{2}{k-r}} \cdot \frac{k}{k-r}$ compétitif pour le poids (pour $k \geq 4$ et $1 \leq r \leq k-3$).*

Remarquons que si on prend par exemple $r = \frac{k-2}{3}$ (si $k-2$ est divisible par 3) on obtient un algorithme $\frac{3}{1-\frac{2}{k}}$-compétitif pour la taille et $\frac{3}{1-\frac{2}{k}}$-compétitif pour le poids. Les rapports sont donc "petits" et ajustables via le paramètre r.

3 Généralisations

La méthode décrite dans la section 2 peut en fait être étendue. L'objet de cette section est de donner les principaux résultats de ces extensions. Celles ci sont utiles pour prendre en compte d'autres fonctions de profit (section 3.1) et d'autres modes de réservation (section 3.2).

3.1 Prise en compte de poids quelconques

Notre façon de "mixer" deux algorithmes monocritères est en fait générale et peut très bien s'appliquer aussi à d'autres fonctions de poids que celles utilisées jusqu'à présent (la maximisation de la taille peut être vue comme une maximisation de poids unitaires). En modifiant la description de l'algorithme on peut arriver, avec les mêmes techniques de preuves, à montrer le résultat plus général suivant.

Théorème 2 *Soit A (resp. B) un algorithme en-ligne d'ordonnancement ayant un rapport de compétitivité ρ_A (resp. ρ_B) pour la maximisation du poids des ordonnancements dans lesquels chaque intervalle σ_i a un poids W_i^A (resp. W_i^B). Notre algorithme utilisant A sur r machines et B sur $k-r$ machines induit un rapport de compétitivité de $\rho_A \frac{k}{r}$ pour la maximisation des poids W^A et, simultanément, un rapport de $\rho_B \frac{k}{k-r}$ pour la maximisation des poids W^B.*

3.2 Le cas des intervalles dégradables

Dans le modèle vu au début de l'article, un intervalle est soit totalement ordonnancé (et dans ce cas on en tire un bénéfice) soit il est rejeté ou interrompu avant sa date de fin et dans ce cas aucun bénéfice n'en est tiré. Cette section est consacrée à un modèle intermédiaire dans lequel chaque demande σ_i a une date de début r_i, une date de fin d_i, un poids $w_i = d_i - r_i$ mais aussi une *date de fin minimale* $r_i < q_i \leq d_i$. Un tel intervalle sera considéré comme ordonnancé s'il est exécuté sur une machine entre les dates r_i et t_i avec $q_i \leq t_i \leq d_i$; ces intervalles sont dits *dégradables*. Dans ce cas, l'intervalle rapporte un profit égal à son temps d'exécution, c'est-à-dire $t_i - r_i$ et ajoute une unité à la taille de l'ordonnancement courant. Sinon, il ne rapporte rien, ni dans l'unité de poids ni dans celle de taille. Ce modèle dégradable a été traité dans [BBL04a] dans le cas hors-ligne et en considérant les deux objectifs séparément.

L'algorithme décrit dans la section 2 peut dans ce cas être modifié et adapté pour prendre en compte ces intervalles *dégradables*. Le résultat est alors le suivant (les rapports de compétitivité sont calculés par rapport aux ordonnancements optimaux dégradables).

Théorème 3 *Notre algorithme en-ligne traitant les intervalles dégradables a un rapport de compétitivité de $\frac{k}{r}$ pour la taille et de $4\frac{k}{k-r-2}$ pour le poids (pour $k \geq 4$ et $1 \leq r \leq k-3$).*

Références

[BBL03] F. Baille, E. Bampis, and C. Laforest. Rapports d'approximation effectifs d'ordonnancements bicritères d'intervalles sur des machines identiques. In *ALGOTEL*, pages 147–154, 2003.

[BBL04a] F. Baille, E. Bampis, and C. Laforest. Maximization of the size and the weight of schedules of degradable intervals. In K.-Y. Chwa and I. Munro, editors, *proceedings of COCOON'04*, LNCS No. 3106, pages 219–228. Springer-Verlag, 2004.

[BBL04b] F. Baille, E. Bampis, and C. Laforest. A note on bicriteria schedules with optimal approximation ratios. *Parallel Processing Letters*, 14:315–323, 2004.

[BEY98] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University press, 1998.

[BNCK+99] Amotz Bar-Noy, Ran Canetti, Shay Kutten, Yishay Mansour, and Baruch Schieber. Bandwidth allocation with preemption. *SIAM J. Comput.*, 28(5):1806–1828, 1999.

[FN95] U. Faigle and M. Nawijn. Note on scheduling intervals on-line. *Discrete Applied Mathematics*, 58:13–17, 1995.

[FW98] A. Fiat and G. J. Woeginger. *Online algorithms: The state of the art*. LNCS no. 1442, Springer, 1998.