Gestion du vidage de données satellite avec incertitude sur les volumes

C. Pralet, A. Maillard, G. Verfaillie, Emmanuel Hébrard, Nicolas Jozefowiez, Marie-José Huguet, T. Desmousceaux, P. Blanc-Paques, J. Jaubert

Gestion du vidage de données satellite avec incertitude sur les volumes

Cédric Pralet[1] **Adrien Maillard**[1] **Gérard Verfaillie**[1]
Emmanuel Hébrard[2] **Nicolas Jozefowiez**[2] **Marie-Jo Huguet**[2]
Thierry Desmousceaux[3] **Pierre Blanc-Paques**[3] **Jean Jaubert**[4]

[1] Onera - The French Aerospace Lab, Toulouse
[2] LAAS/CNRS, Toulouse
[3] Airbus Defence and Space, Toulouse
[4] CNES, Toulouse

Prénom.Nom@onera.fr Prénom.Nom@laas.fr
Prénom.Nom@astrium.eads.net Prénom.Nom@cnes.fr

Résumé

Les satellites d'observation de la Terre sont des senseurs qui acquièrent des données, les compressent et les mémorisent à bord, puis les vident vers le sol. Du fait de l'utilisation d'algorithmes de compression de plus en plus sophistiqués, le volume de données résultant d'une acquisition est de moins en moins prédictible. Dans de telles conditions, planifier les activités de vidage du satellite hors-ligne au sol est de plus en plus problématique. Dans ce papier, nous rapportons les résultats d'une étude visant à évaluer le bénéfice d'une planification des activités de vidage à bord, quand le volume de données produit par chaque acquisition est connu.

Le problème de vidage des données à résoudre à bord est un problème d'affectation et d'ordonnancement avec des ressources non partageables, des contraintes de précédence, des durées minimum dépendantes du temps et un critère d'optimisation complexe. La librairie générique InCELL [10] est utilisée pour modéliser contraintes et critère, vérifier les contraintes non temporelles, propager les contraintes temporelles et évaluer le critère. Au dessus de cette librairie, des algorithmes de recherche gloutonne et locale ont été développés pour produire des plans de vidage, compte-tenu du temps et des ressources de calcul limités disponibles à bord.

Abstract

Earth observation satellites are space sensors which acquire data, compress and record it on board, and then download it to the ground. Because of the use of more and more sophisticated compression algorithms, the amount of data resulting from an acquisition is less and less predictable. In such conditions, planning satellite data download activities offline on the ground is more and more problematic. In this paper, we report the results of a work aiming at evaluating the positive impact of planning downloads onboard when the amount of data produced by each acquisition is known.

The data download problem to be solved on board is an assignment and scheduling problem with unsharable resources, precedence constraints, time-dependent minimum durations, and a complex optimization criterion. The generic InCELL library [10]) is used to model constraints and criterion, to check non temporal constraints, to propagate temporal constraints, and to evaluate the criterion. On top of this library, greedy and local search algorithms have been designed to produce download plans with limited time and computing resources available on board.

1 Introduction

Les satellites d'observation de la Terre sont des senseurs qui acquièrent des données, les compressent et les mémorisent à bord, puis les vident vers le sol. Du fait de l'utilisation d'algorithmes de compression de plus en plus sophistiqués, le volume de données qui résulte d'une acquisition et qui doit être mémorisé à bord et ensuite vidé vers le sol est de moins en moins prédictible. Il dépend des données acquises. Par exemple, dans le cas d'instruments optiques, la présence de nuages sur la zone observée permet une forte compression dont le résultat est un faible volume de don-

nées à mémoriser et à vider.

Dans de telles conditions, la façon classique de gérer ces satellites devient de plus en plus problématique. En effet, jusqu'à maintenant, toutes les décisions sont prises hors-ligne au sol et le satellite est un simple exécutant qui ne prend, ni ne modifie aucune décision. Typiquement, chaque jour, un plan d'activité incluant les activités d'acquisition et de vidage avec des dates de début précises est construit au sol pour le jour suivant. Ce plan est téléchargé vers le satellite via une station de contrôle et ensuite exécuté tel quel sans aucune modification. Dans un contexte où le volume de données généré par les acquisitions est incertain, si les volumes maximaux sont considérés au moment de la planification, les plans seront toujours exécutables, mais le système pourra être sous-utilisé. Si des volumes espérés ou n'importe quels volumes inférieurs aux volumes maximaux sont considérés, les plans pourront ne pas être exécutables.

Une option alternative consiste à modifier au moins en partie la façon de gérer ces satellites, en prenant les décisions de vidage le plus tard possible, quand les acquisitions sont réalisées et les volumes générés sont connus. Comme ces satellites ne sont pas de façon permanente accessibles par une station de contrôle et comme les volumes générés sont d'abord connus à bord, ces décisions doivent et peuvent être prises à bord. Par exemple, juste avant une fenêtre de visibilité d'une station de réception pendant laquelle des vidages sont possibles, le satellite peut construire un plan de vidage qui prenne en compte les volumes générés par toutes les acquisitions déjà réalisées.

Dans cette étude, nous considérons le contexte des futurs satellites Post-Pléiades : la génération des satellites d'observation optique de la Terre qui suivra celle des satellites Pléiades actuellement opérationnels. Dans un tel contexte, de nouvelles contraintes de vidage doivent être prises en compte telles que la mémorisation des données sur différentes tranches mémoire, l'utilisation de plusieurs canaux de vidage ou l'emploi de clés de cryptage. De plus, les critères de vidage à optimiser sont le nombre et l'importance des acquisitions vidées, le délai entre acquisitions et vidages et le partage équitable de la ressource satellite entre utilisateurs.

Le premier objectif de l'étude était de développer au dessus d'une bibliothèque générique d'optimisation sous contraintes des algorithmes efficaces capables de prendre des décisions de vidage de façon autonome à bord. Le second objectif était d'évaluer l'impact opérationnel de l'utilisation de tels algorithmes par rapport à la façon classique de gérer ces satellites entièrement depuis le sol.

Le papier est organisé de la façon suivante. Dans la section suivante, l'architecture mixte de décision bord-sol que nous supposons est décrite. Puis, le problème de planification des vidages de données est informellement décrit et un modèle à base de contraintes de ce problème est proposé et analysé. L'approche algorithmique retenue, reprenant les idées de recherche locale à base de contraintes (*Constraint-based Local Search*, CLS [4]) et utilisant la bibliothèque générique InCELL (*Invariant-based Constraint EvaLuation Library* [10]), est ensuite décrite. Après une présentation des divers scénarios utilisés pour l'évaluation, les résultats expérimentaux permettent de comparer l'efficacité des différentes variantes algorithmiques et d'évaluer l'impact positif de la prise de décision à bord.

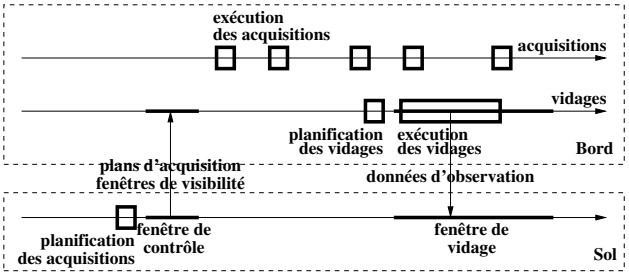

FIGURE 1 – Architecture de prise de décision.

2 Architecture de prise de décision

Nous considérons le contexte des futurs satellites Post-Pléiades avec les hypothèses suivantes sur le système physique :

- l'instrument d'observation est fixe sur le satellite, mais l'antenne de vidage des données est mobile avec une vitesse et un angle de débattement maximaux ;
- pour observer une zone sol, l'instrument d'observation et donc tout le satellite doit être pointé vers elle ; pour vider des données vers une station sol, l'antenne de vidage doit être pointée vers elle ;
- grâce à des actionneurs gyroscopiques, le satellite est agile et capable de se mouvoir rapidement en attitude selon les trois axes autour de son centre de gravité tout en se déplaçant sur son orbite ;
- acquisitions et vidages peuvent être réalisés en parallèle.

Dans ce contexte, la figure 1 montre l'architecture de prise de décision que nous supposons.

Planification des acquisitions Les plans d'acquisition sont construits comme d'habitude hors-ligne au sol. Ceci est justifié par le fait que leur construction est coûteuse en temps de calcul et que les requêtes d'acquisition sont d'abord connues au sol. Ces plans ne considèrent ni les limitations en termes de mémoire et de vidage, ni les activités de vidage. D'un plan d'acquisition, il est possible de déduire, pour chaque station de réception st, les fenêtres sur lesquelles vider des données vers st est effectivement possible (pointer l'antenne mobile vers st est possible, en prenant en compte la position et l'orientation du satellite, la

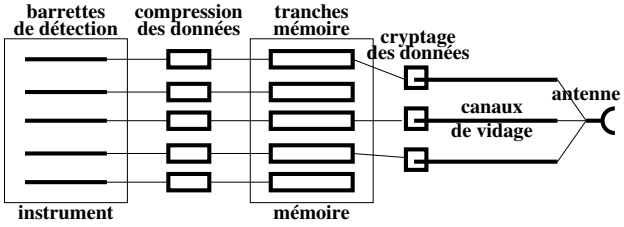

FIGURE 2 – Comment les données sont produites, mémorisées et vidées.

position de la station et l'angle maximum de débattement de l'antenne). Le plan d'acquisition et les fenêtres de visibilité résultantes sont alors téléchargés vers le satellite via n'importe quelle station de contrôle.

Exécution des acquisitions Les plans d'acquisition sont exécutés à bord sans aucune modification, sauf quand une acquisition a conduirait à un dépassement mémoire. Dans ce cas, a est réalisée si et seulement si il est possible de libérer suffisamment de mémoire en effaçant des acquisitions de plus faible priorité.

Planification des vidages Les plans de vidage sont construits à bord avant une fenêtre de visibilité ou un groupe de fenêtres chevauchantes. Ceci est justifié par le fait que les volumes générés par les acquisitions sont d'abord connus à bord. Ces plans prennent en compte les volumes exacts pour les acquisitions déjà réalisées et les volumes maximaux pour les acquisitions qui se terminent durant la(les) fenêtre(s) de visibilité concernées.

Exécution des vidages Les plans de vidage sont ensuite exécutés de façon flexible, en prenant en compte le fait que les volumes générés par les acquisitions qui se terminent durant la(les) fenêtre(s) de visibilité peuvent être plus faibles que les volumes maximaux pris en compte par la planification. Cela peut permettre de commencer des vidages plus tôt que ce qui avait été planifié.

3 Problème de planification des vidages

La figure 2 montre comment les données sont produites, mémorisées et vidées.

Production et mémorisation des données Pour obtenir une observation multi-fréquence, chaque acquisition active un sous-ensemble de barrettes de détection. Elle produit un ensemble de fichiers de données, chacun étant mémorisé sur une tranche mémoire avec une taille qui dépend du taux de compression effectif.

Vidage des données Le vidage peut utiliser plusieurs canaux d'émission concurrents. Chaque fichier doit être vidé sans interruption sur un canal. Le débit de vidage est une fonction constante par morceaux de la distance satellite-station. Du fait du mouvement du satellite sur son orbite et de la Terre sur elle-même, la distance satellite-station évolue. En conséquence la durée de vidage d'un fichier dépend de l'instant auquel il débute. Les fichiers issus d'une acquisition peuvent être vidés dans n'importe quel ordre sur n'importe quels canaux, mais doivent être tous vidés dans la même fenêtre de visibilité. Canaux et tranches mémoire sont des ressources non partageables. Cela implique que, si deux fichiers sont mémorisés sur la même tranche ou utilisent le même canal, leurs vidages ne peuvent pas se chevaucher.

Cryptage des données Les données sont cryptées avant vidage. Une clé est associée à chaque utilisateur. Physiquement, un composant de cryptage est associé à chaque canal, ce qui permet à des données associées à des utilisateurs différents d'être vidées en parallèle. Une table de changement de clé, qui contient tous les changements à effectuer et leurs instants précis, est associée à chaque canal, ce qui permet à des données associées à des utilisateurs différents d'être vidées en séquence sur chaque canal. Le nombre de changements qu'il est possible de mémoriser dans la table est limité. Réinitialiser cette table prend un certain temps et interrompt les vidages sur tous les canaux.

Fenêtres de vidage Vider des données est possible uniquement à l'intérieur des fenêtres de visibilité. De plus, en fonction de l'utilisateur, seules certaines stations et donc certaines fenêtres sont autorisées. Nous supposons qu'une fenêtre de visibilité peut inclure plusieurs fenêtres de vidage, chacune avec sa table de changement de clé associée. En conséquence, entre deux fenêtres de vidage successives, un temps minimum est nécessaire pour réinitialiser la table. De plus, si les deux fenêtres successives sont associées à deux stations différentes, passer de l'une à l'autre prend un certain temps requis pour pointer l'antenne vers la nouvelle station. Ce temps de transition dépend de l'instant auquel la transition est enclenchée du fait du mouvement du satellite sur son orbite et sur lui-même et du mouvement de la Terre sur elle-même.

Dates de vidage au plus tôt et au plus tard Le vidage d'une acquisition doit débuter après l'acquisition elle-même et se terminer avant une date limite au delà de laquelle les données n'ont plus aucune valeur.

La figure 3 représente un plan de vidage valide où les décisions d'affectation et de vidage ont été prises, mais les dates de début de vidage n'ont pas encore été fixées (plan temporellement flexible). Ce plan implique 4 acquisitions A, B, C, and D, chacune produisant 5 fichiers (A1, A2, A3, A4 et A5 pour l'acquisition A) distribués sur 5 tranches mémoire, 2 fenêtres de visibilité chevauchantes (Vw1 vers la station S1 et Vw2 vers la station S2), une fenêtre de vidage par fenêtre de visibilité et 3 canaux de vidage.

Seules les contraintes temporelles de précédence sont représentées : (1) séquences de vidage sur chaque canal, (2)

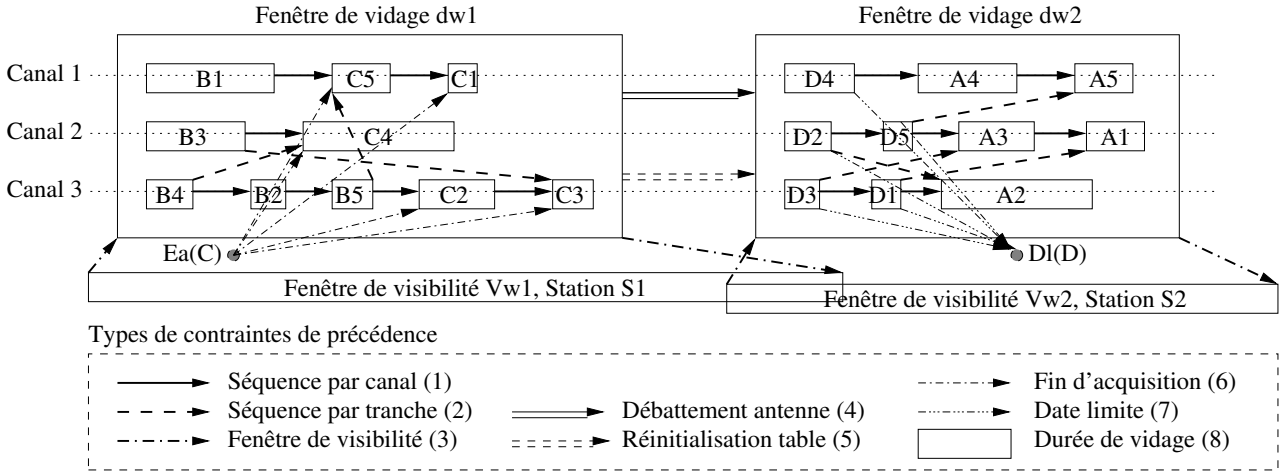

FIGURE 3 – Exemple de plan de vidage avec ses contraintes temporelles de précédence.

séquences de vidage depuis chaque tranche, (3) vidages dans les fenêtres de visibilité, temps de transition entre deux fenêtres de vidage successives dû (4) au mouvement de l'antenne et (5) à la ré-initialisation de la table de changement de clé, (6) acquisition avant vidage (pour l'acquisition C qui se termine à l'instant Ea(C) dans la fenêtre Vw1) et (7) vidage avant la date limite (pour l'acquisition D dont la date limite de vidage Dl(D) se situe dans la fenêtre Ww2). Les durées de vidage (8) sont implicitement représentées par les rectangles associés à chaque fichier.

Les contraintes non temporelles ne sont pas représentées : (9) tous les fichiers issus d'une acquisition vidés dans une même fenêtre de visibilité dont la station associée est autorisée et (10) nombre maximum de changements de clé sur chaque canal pour chaque fenêtre de vidage.

4 Modèle

4.1 Variables de décision

Les variables de décision peuvent être classées en variables non temporelles et variables temporelles. Les variables non temporelles incluent :

- un nombre ndw de fenêtres de vidage et la séquence $dwSeq$ (de longueur ndw) de fenêtres de vidage ;
- pour chaque acquisition a, la fenêtre de vidage dw_a à l'intérieur de laquelle a est vidée et, pour chaque fichier associé f, le canal $ch_{a,f}$ sur lequel f est vidé ;
- pour chaque fenêtre de vidage w, la fenêtre de visibilité associée vw_w, pour chaque canal c, la séquence $cSeq_{w,c}$ de vidages de fichiers sur c et, pour chaque tranche m, la séquence $mSeq_{w,m}$ de vidages de fichiers depuis m.

Les variables temporelles incluent :

- pour chaque acquisition a et chaque fichier associé f, les dates de début et de fin $sdf_{a,f}$ et $edf_{a,f}$ du vidage de f ;
- pour chaque fenêtre de vidage w, ses dates de début et de fin sdw_w et edw_w.

4.2 Contraintes

Sur la base de cette définition des variables de décision, toutes les contraintes de (1) à (10) informellement présentées à la fin de la section 3 peuvent être formulées.

Beaucoup d'entre elles sont des contraintes temporelles simples [2] comme la contrainte (3) qui impose que toute fenêtre de vidage soit incluse dans sa fenêtre de visibilité associée, avec $\mathbf{Sw}_{vw}$ and $\mathbf{Ew}_{vw}$ les dates de début et de fin d'une fenêtre de visibilité vw :

$$\forall w \in [\![1; ndw]\!] : \ (\mathbf{Sw}_{vw_w} \leq sdw_w) \wedge (edw_w \leq \mathbf{Ew}_{vw_w})$$

ou les contraintes (6) et (7) qui expriment que le vidage d'une acquisition doit commencer après l'acquisition elle-même et finir avant la date limite, avec **Na** le nombre d'acquisitions, $\mathbf{Nf}_a$ le nombre de fichiers résultant d'une acquisition a, $\mathbf{Ea}_a$ la date de fin d'une acquisition a et $\mathbf{Dl}_a$ sa date limite de vidage :

$$\forall a \in [\![1; \mathbf{Na}]\!], \forall f \in [\![1; \mathbf{Nf}_a]\!] :$$
$$(\mathbf{Ea}_a \leq sdf_{a,f}) \wedge (edf_{a,f} \leq \mathbf{Dl}_a)$$

Certaines d'entre elles sont des contraintes temporelles simples dépendantes du temps [11] comme la contrainte (8) qui exprime la durée du vidage d'un fichier en fonction de la date de début de vidage, avec $\mathbf{V}_{a,f}$ le volume d'un fichier f issu d'une acquisition a et **DuDl** la durée de vidage en fonction du volume à vider, de la fenêtre de visibilité et de la date de début de vidage dans cette fenêtre :

$$\forall a \in [\![1; \mathbf{Na}]\!], \forall f \in [\![1; \mathbf{Nf}_a]\!] :$$
$$edf_{a,f} - sdf_{a,f} = \mathbf{DuDl}(\mathbf{V}_{a,f}, vw_{dw_a}, sdf_{a,f})$$

Certaines d'entre elles sont des contraintes non temporelles comme la contrainte (8) qui impose qu'une acquisition soit vidée vers une station autorisée, avec $\mathbf{St}_{vw}$ la station associée à une fenêtre de visibilité vw, $\mathbf{Ss}_u$ l'ensemble des stations autorisées par un utilisateur u et $\mathbf{U}_a$ l'utilisateur demandeur d'une acquisition a :

$$\forall a \in [\![1; \mathbf{Na}]\!] : \ \mathbf{St}_{vw_{dw_a}} \in \mathbf{Ss}_{\mathbf{U}_a}$$

4.3 Critère d'optimisation

Dans un but de simplification, le critère d'optimisation considéré dans ce papier est purement utilitariste et ne considère pas l'objectif de partage équitable de l'usage du satellite entre ses différents utilisateurs. Il est défini comme un vecteur d'utilités, avec une utilité par niveau de priorité. Pour chaque niveau de priorité p, l'utilité U_p est simplement définie comme la somme des poids des acquisitions vidées de priorité p, pondérés par leur coefficient de fraîcheur, de façon à favoriser des délais courts entre acquisition et vidage :

$$\forall p \in [\![1; \mathbf{Np}]\!] : \ U_p = \sum_{a \in [\![1; \mathbf{Na}]\!] \mid \mathbf{P}_a = p} \mathbf{W}_a \cdot \mathbf{Fr}_a(eda_a)$$

où **Np** est le nombre de niveaux de priorité, **Na** le nombre d'acquisitions, $\mathbf{P}_a$ et $\mathbf{W}_a$ la priorité et le poids d'une acquisition a, $\mathbf{Fr}_a$ son niveau de fraîcheur (compris entre 0 et 1) en fonction (monotone décroissante) de sa date eda_a de fin de vidage : $eda_a = \max_{f \in [\![1; \mathbf{Nf}_a]\!]} edf_{a,f}$.

Deux plans de vidage sont comparés en comparant leurs deux vecteurs d'utilité de façon lexicographique des priorités 1 à **Np** : n'importe quelle amélioration à un niveau de priorité p est préféré à n'importe quelle amélioration à des niveaux de priorité inférieurs à p.

5 Analyse du problème

Toujours dans un but de simplification, nous supposons dans ce papier qu'au plus une fenêtre de vidage (éventuellement aucune) est associée à une fenêtre de visibilité station et que les fenêtres de vidage résultantes sont ordonnées suivant la date de début de leur fenêtre de visibilité associée. Voir l'exemple de la figure 3 où nous considérons deux fenêtres de vidage $dw1$ et $dw2$ et la séquence $[dw1, dw2]$.

Dans ces conditions, le problème à traiter combine trois sous-problèmes connectés :

1. un problème d'affectation où une fenêtre de vidage (éventuellement aucune) est associée à chaque acquisition : variables dw_a ;
2. un problème d'ordonnancement où un canal (éventuellement aucun) est associé à chaque fichier et où les vidages de fichiers sont ordonnés sur chaque canal et chaque tranche : variables $ch_{a,f}$, $cSeq_{w,c}$ et $mSeq_{w,m}$;
3. un problème temporel où les dates de début et de fin sont fixées : variables $sdf_{a,f}$, $edf_{a,f}$, sdw_w et edw_w.

Problème d'affectation Le problème d'affectation est proche du problème *Multi-knapsack* [6] où les objets sont les acquisitions et les sacs sont les fenêtres, mais les premiers sacs sont préférés et des conflits peuvent exister entre sacs (fenêtres chevauchantes).

Problème d'ordonnancement Dans chaque fenêtre de vidage, le problème d'ordonnancement est proche du problème *Flexible Open-shop Scheduling* [8] avec deux types de ressources non partageables : canaux et tranches. Chaque vidage requiert une ressource de chaque type, mais le choix du canal est libre alors que la tranche est préaffectée. L'ordre des vidages est libre.

Problème temporel Quand les problèmes d'affectation et d'ordonnancement ont été résolus, c'est-à-dire quand toutes les variables non temporelles ont été affectées, le problème temporel résultant a la forme d'un *Simple Temporal Network* (STN [2]). Les seules exceptions sont les contraintes de durée de vidage et de transition entre fenêtres de vidage qui sont dépendantes du temps (durées de vidage et de transition entre fenêtres dépendent des instants auxquels vidages et transitions commencent). Le résultat est un *Time-dependent* STN (TSTN [11]) pour lequel les techniques STN peuvent être étendues et des algorithmes polynomiaux peuvent décider de la cohérence et, en cas de cohérence, calculer les dates au plus tôt et au plus tard de toutes les variables temporelles.

6 Algorithmes de planification

Globalement, les algorithmes que nous avons développés sont des algorithmes gloutons heuristiques non chronologiques qui, de façon séquentielle et sans aucun *backtrack*, choisissent une acquisition et tentent de l'ajouter au plan de vidage. Ce choix algorithmique est justifié par les limitations bord en termes de temps et de ressources de calcul. Il faut cependant souligner que ce choix n'est pas définitif et que d'autres choix sont possibles sur la base du même modèle, tels que des algorithmes de type *squeaky wheel optimization* [5], *tabu search* [3] ou *simulated annealing* [7]. Il faut aussi souligner que ces algorithmes gloutons incluent des mécanismes sophistiqués de choix

d'affectation et d'ordonnancement des vidages, de vérification des contraintes non temporelles et de propagation des contraintes temporelles. Conformément à l'analyse du problème en section 5, ces algorithmes comportent trois parties : affectation des vidages, ordonnancement des vidages et vérification et propagation des contraintes.

6.1 Affectation des vidages

Les algorithmes d'affectation prennent en entrée un plan de vidage courant cohérent et un ensemble d'acquisitions candidates (non encore affectées). Ils fournissent en sortie une acquisition a (sélectionnée parmi les candidates) et une fenêtre de vidage pour a. Nous avons développé deux algorithmes d'affectation : **MaxWeight** et **MinRegret**.

MaxWeight est l'algorithme le plus basique. À chaque étape, il sélectionne une acquisition a de priorité maximum et de poids maximum, avec tirage aléatoire en cas d'égalité, et sélectionne la première fenêtre (au sens chronologique) dans laquelle insérer a est possible.

MinRegret est un peu plus sophistiqué. Il maintient pour chaque acquisition a la première et la seconde fenêtre ($w1_a$ et $w2_a$) dans lesquelles insérer a est possible, ainsi que le regret qui résulterait du fait de ne pas choisir la première ($w1_a$). Si $eda1_a$ et $eda2_a$ sont les dates de fin de vidage de a qui résulterait du choix de $w1_a$ et de $w2_a$, le regret R peut être défini comme suit :

$$\forall a \in [\![1; \mathbf{Na}]\!] : \; R(a) = \mathbf{W}_a \cdot (\mathbf{Fr}_a(eda1_a) - \mathbf{Fr}_a(eda2_a))$$

En s'inspirant des heuristiques classiques utilisées pour traiter les problèmes de sac-à-dos, l'algorithme sélectionne à chaque étape une acquisition a de priorité maximum et de ratio maximum $R(a)/\sum_{f \in [\![1;\mathbf{Nf}_a]\!]} \mathbf{V}_{a,f}$ entre regret et volume et sélectionne la première fenêtre ($w1_a$) pour a.

6.2 Ordonnancement des vidages

Les algorithmes d'ordonnancement prennent en entrée un plan de vidage courant cohérent, une fenêtre de vidage w et une acquisition a à insérer dans w. Ils produisent un nouveau plan de vidage cohérent avec a inséré dans w ou un échec si l'insertion a échoué.

Nous avons développé trois algorithmes d'ordonnancement : **EnQueue**, **IdleFill** et **Scheduler**. Les deux premiers sont des algorithmes gloutons, tandis que le troisième utilise des mécanismes de recherche locale. De plus, les deux premiers produisent des ordonnancements où les vidages des acquisitions sont totalement ordonnés : vider $a1$ avant $a2$ implique que, sur chaque canal et chaque tranche, aucun fichier de $a2$ n'est vidé avant un fichier de $a1$. Au contraire, le troisième algorithme construit des ordonnancements où il est possible d'entrelacer les vidages des acquisitions.

EnQueue est l'algorithme le plus basique. Pour insérer une acquisition a dans une fenêtre w, il l'insère systématiquement à la fin de la séquence courante de vidage.

IdleFill remédie au défaut le plus évident de l'algorithme précédent : comme certaines acquisitions se terminent au cours de fenêtres de visibilité, certains vidages doivent attendre la fin d'acquisition ; ceci peut engendrer des périodes d'inactivité dans l'ordonnancement. Pour éviter ce phénomène, **IdleFill** insère une acquisition a à la première position dans la séquence courante de vidage où au moins un canal est inactif et l'insertion est possible.

Pour les deux algorithmes, suivant une heuristique de type plus contraint en premier, l'insertion de tous les fichiers de a est réalisée de façon séquentielle suivant un ordre de volume décroissant. Toujours, pour limiter les périodes d'inactivité dans l'ordonnancement, pour chaque fichier f, s'il existe un canal c sur lequel le dernier fichier f' vient de la même tranche que f, alors f est placé sur c juste après f' ; sinon f est placé sur un canal c qui permet un vidage au plus tôt.

Scheduler est un algorithme de recherche locale inspiré de mécanismes efficaces utilisés pour traiter des problèmes classiques d'ordonnancement. Il est appelé en cas d'échec des algorithmes gloutons précédents et peut démarrer de l'ordonnancement qu'ils produisent. Dans cet ordonnancement, les contraintes sur les fenêtres de visibilité sont violées et le but est de produire un ordonnancement plus court qui les satisfasse. Pour cela, l'algorithme réalise une séquence de mouvements locaux dont le but est de réduire la longueur du chemin critique (la séquence de vidages dont la longueur induit la longueur de l'ordonnancement ; si rien n'est changé dans cette séquence, la longueur de l'ordonnancement ne sera pas réduite et les contraintes sur les fenêtres de visibilité resteront violées). Ces mouvements locaux sont conçus pour éviter de produire des cycles dans l'ordonnancement : ordre sur les canaux contradictoire avec l'ordre sur les tranches.

6.3 Vérification et propagation des contraintes

La librairie générique InCELL (*Invariant-based Constraint EvaLuation Library* [10]) est utilisée pour modéliser variables, contraintes et critère, pour vérifier les contraintes non temporelles, pour propager les contraintes temporelles et pour évaluer le critère. InCELL tire son inspiration des idées de *Constraint-based Local Search* (CLS [4]).

Avec CLS, comme avec d'autres approches déclaratives pour l'optimisation sous contraintes, l'utilisateur définit un modèle de son problème en termes de variables de décision, de contraintes et de critère à optimiser. Puis, il définit un algorithme de recherche gloutonne ou locale [1] dans l'espace d'affectation des variables. Le modèle n'est pas utilisé pour propager les contraintes comme avec les al-

gorithmes classiques de recherche arborescente, mais pour vérifier les contraintes et évaluer le critère en fonction de l'affectation courante. Parce que le nombre de mouvements locaux réalisés en un temps limité est une des clés du succès de ces algorithmes, des techniques efficaces sont utilisées pour réaliser chaque mouvement local aussi rapidement que possible. Ces techniques utilisent une traduction du modèle (variables, contraintes et critère) en un DAG (*Directed Acyclic Graph*) d'invariants. Chaque invariant a un ensemble d'entrées et une sortie. Il maintient une certaine fonction des entrées vers la sortie, par exemple le fait qu'une variable est égale à la somme d'autres variables : $x = \sum_{i=1}^{N} y_i$ et il la maintient de façon incrémentale : sur cet exemple, si la valeur d'une certaine variable y_k est modifiée, il n'est pas nécessaire de recalculer toute la somme ; il suffit d'ajouter à x la différence entre la nouvelle et l'ancienne valeur de y_k. Globalement, après chaque changement dans l'affectation des variables, le DAG d'invariants est ré-évalué de façon paresseuse et incrémentale : seuls les invariants dont une des entrées est modifiée sont ré-évalués de façon incrémentale.

InCELL est une implémentation des idées de CLS visant plus particulièrement les problèmes d'ordonnancement et donc la gestion du temps et des ressources. Avec InCELL, des invariants à entrées et sorties multiples permettent de représenter des expressions, des contraintes arithmétiques et logiques et des contraintes sur le temps et les ressources. De plus, dans InCELL, les contraintes temporelles simples [2] de la forme $y - x \leq D$, où x et y représentent deux positions temporelles et D une constante, sont gérées de façon spécifique : les variables temporelles ne sont pas affectées comme les variables non temporelles le sont ; les techniques de type STN sont utilisées pour propager les contraintes temporelles, pour vérifier leur cohérence et pour calculer les dates au plus tôt et au plus tard de chaque variable temporelle. Ce traitement spécifique est justifié par l'existence d'algorithmes polynomiaux capables de vérifier la cohérence d'un STN et de calculer les dates au plus tôt et au plus tard de chaque variable du STN. De tels algorithmes n'existent en général pas pour les contraintes non temporelles. En plus, InCELL permet d'exprimer et de traiter de la même façon des contraintes temporelles simples dépendantes du temps [11] de la forme $y - x \leq D(x, y)$, où D n'est plus une constante, mais une fonction de x et y.

Une fois que les contraintes non temporelles ont été vérifiées et les contraintes temporelles propagées par InCELL, le résultat est un plan de vidage qui affecte à chaque variable temporelle sa date au plus tôt, car il est préférable de vider les données au plus tôt. A partir de ce plan de vidage, les tables de changement de clé sont fixées avec des dates de changement précises pour chaque fenêtre de vidage et chaque canal.

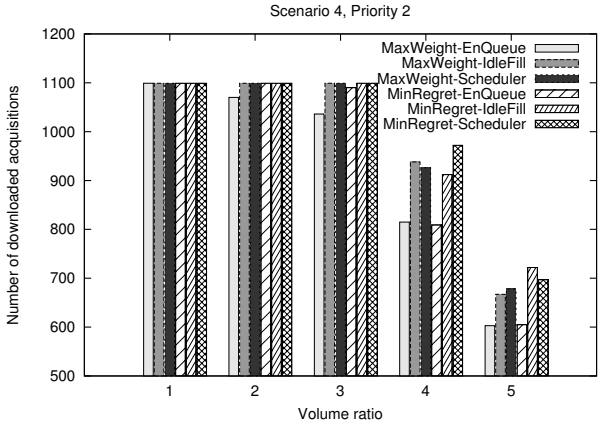

FIGURE 4 – Nombre d'acquisitions vidées de priorité 2 sur le scénario 4. On cherche à maximiser ce nombre.

	Scénario 1		Scénario 4	
	moyen	max	moyen	max
MaxWeight-EnQueue	0.041	0.221	0.048	0.460
MaxWeight-IdleFill	0.035	0.221	0.049	0.686
MaxWeight-Scheduler	0.232	4.149	0.655	21.708
MinRegret-EnQueue	0.161	1.434	0.239	3.531
MinRegret-IdleFill	0.152	1.544	0.215	2.839
MinRegret-Scheduler	0.157	1.521	2.116	78.511

TABLE 1 – Temps de calcul moyen et maximum (en secondes) sur les scénarios 1 et 4 (processeur Intel i5-520, 1.2GHz, 4GBRAM).

7 Algorithmes d'exécution

Dans l'architecture de prise de décision retenue, des plans de vidage sont construits avant chaque fenêtre ou groupe de fenêtres de visibilité en faisant l'hypothèse de volumes maximaux pour toutes les acquisitions se terminant pendant la(les) fenêtre(s) de visibilité. Si les volumes réels sont plus faibles que les volumes maximums, il est possible de démarrer des vidages plus tôt que prévu et donc d'améliorer la qualité du plan sans modifier les décisions d'affectation et d'ordonnancement.

Pour implémenter une exécution flexible et réactive des plans, nous nous inspirons de l'approche *Partial Order Schedule* (POS [9]) et construisons à partir de n'importe quel plan de vidage un graphe de précédence (un DAG) qui représente toutes les précédences qui doivent être satisfaites par l'exécution. Une fois ce graphe construit, il peut être exécuté dans un ordre topologique : chaque nœud est exécuté dès que tous ses prédécesseurs dans le graphe ont été exécutés. On peut facilement montrer que, du fait des volumes maximaux pris en compte par la planification, une telle exécution ne conduira jamais à une violation des

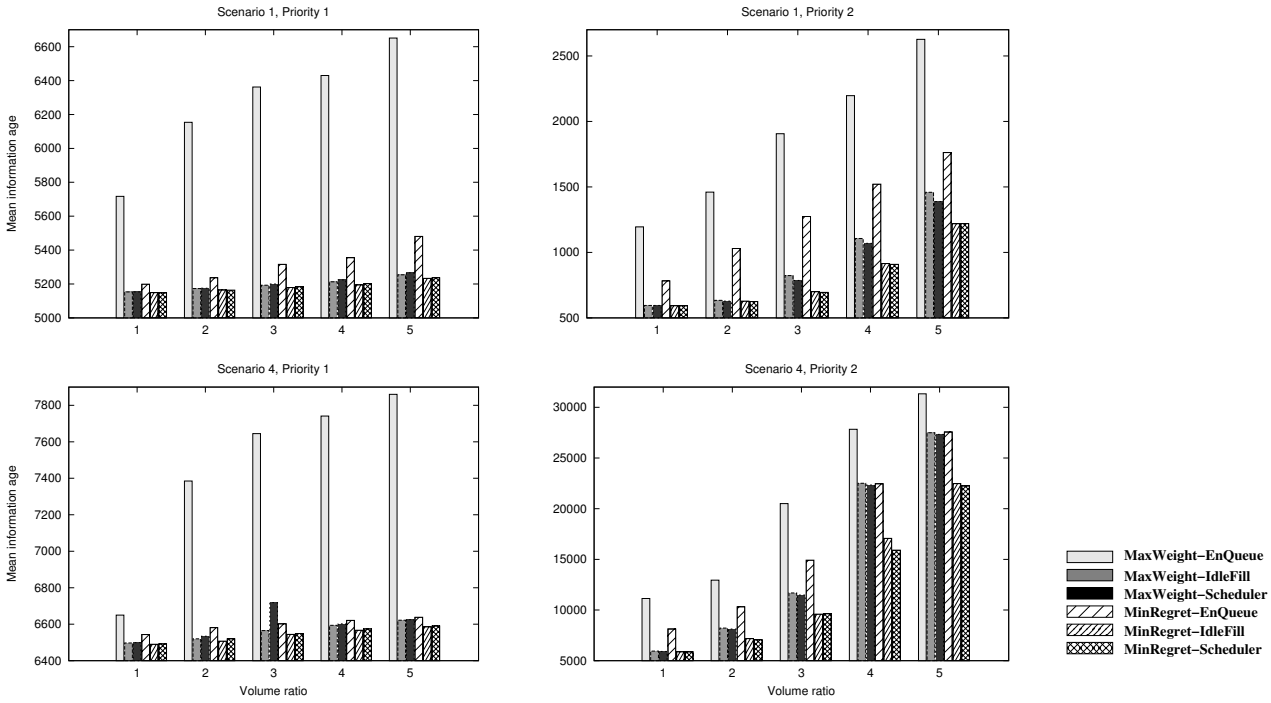

FIGURE 5 – Age moyen de l'information (en secondes) pour les priorités 1 et 2 (à gauche et à droite) sur les scénarios 1 et 4 (en haut et en bas). On cherche à minimiser cet âge.

contraintes sur les fenêtres de visibilité et les dates limites de vidage.

8 Résultats expérimentaux

8.1 Scénarios

L'architecture de prise de décision et les divers algorithmes de planification ont été expérimentés sur différents scénarios réalistes fournis par Airbus Defence and Space. Chaque scénario couvre un jour d'activité du satellite.

Ces scénarios impliquent 5 tranches mémoire, 3 canaux de vidage, 5 utilisateurs, 2 niveaux de priorité, de 3 à 23 stations de réception, de 20 à 115 fenêtres de visibilité associées et 1364 acquisitions à vider. Comme chaque acquisition produit 5 fichiers, le nombre de fichiers à vider est égal à 6820. Si $Vmax$ est le volume maximum d'un fichier (sans compression), son volume réel (après compression) est généré de façon aléatoire entre $Vmax/4$ et $Vmax$. Les scénarios diffèrent suivant le nombre de stations de réception disponibles et la façon dont les acquisitions sont distribuées entre utilisateurs et niveaux de priorité.

Nous présentons les résultats obtenus sur deux scénarios typiques : le scénario 1 qui implique un grand nombre de stations de réception (23) induisant de nombreuses opportunités de vidage et le scénario 4 qui implique un petit nombre de stations de réception (3) induisant peu d'opportunités de vidage. De plus, nous présentons les résultats obtenus sur les deux scénarios avec différentes hypothèses concernant le volume de données généré par une acquisition avant compression : ratio en volume variant de 1 à 5 conduisant à des problèmes de plus en plus sur-contraints. Par exemple, le scénario 1 avec un ratio en volume égal à 1 est largement sous-contraint : chaque acquisition peut être vidée dans la première fenêtre de visibilité suivant l'acquisition. Au contraire, le scénario 4 avec un ratio en volume égal à 5 est fortement sur-contraint : les acquisitions ne peuvent pas toutes être vidées et un grand nombre reste en mémoire à bord à la fin de la journée

8.2 Comparaison ente algorithmes de planification

Nous comparons tout d'abord les six algorithmes de planification résultant de la combinaison des algorithmes d'affectation et d'ordonnancement : **MaxWeight** ou **MinRegret** pour l'affectation et **EnQueue**, **IdleFill** ou **Scheduler** pour l'ordonnancement.

En ce qui concerne le nombre d'acquisitions vidées, toutes les acquisitions sont vidées dans le scénario 1 et toutes les acquisitions de priorité 1 le sont dans le scénario 4, mais pas toutes les acquisitions de priorité 2 avec des ratios en volume élevés. La figure 4 montre le nombre d'acquisitions de priorité 2 vidées dans le scénario 4 en fonction

Scénario 1 : Nombre d'acquisitions vidées						
ratio en volume		1	2	3	4	5
prio. 1	sol	269	269	269	269	269
	bord	269	269	269	269	269
prio. 2	sol	1087	1087	1087	988	729
	bord	1087	1087	1087	1087	1087

Scénario 1 : Age moyen de l'information						
ratio en volume		1	2	3	4	5
prio. 1	sol	5167	5200	5232	5268	5419
	bord	5149	5165	5179	5195	5233
prio. 2	sol	610	692	993	1505	2403
	bord	593	628	700	915	1219

Scénario 4 : Nombre d'acquisitions vidées						
ratio en volume		1	2	3	4	5
prio. 1	sol	244	244	244	244	244
	bord	244	244	244	244	244
prio. 2	sol	1099	1099	595	297	246
	bord	1099	1099	1099	912	722

Scénario 4 : Age moyen de l'information						
ratio en volume		1	2	3	4	5
prio. 1	sol	6639	6725	6758	6914	7125
	bord	6490	6507	6544	6567	6586
prio. 2	sol	6700	12898	13539	13771	11784
	bord	5875	7180	9591	17073	22474

TABLE 2 – Comparaison entre planification sol et bord : nombre d'acquisitions vidées (à gauche) et âge moyen de l'information (en secondes ; à droite) pour les priorités 1 et 2 sur les scénarios 1 et 4 (en haut et en bas).

du ratio en volume (de 1 à 5).

La figure 5 montre l'âge moyen de l'information en secondes (distance moyenne entre acquisition et vidage sur toutes les acquisitions vidées) pour les priorités 1 et 2 (à gauche et à droite) dans les scénarios 1 et 4 (en haut et en bas) en fonction du ratio en volume (de 1 à 5). Le fait que l'âge de l'information pour les acquisitions de priorité 1 soit plus grand que pour celles de priorité 2 est dû à un plus petit nombre de stations de réception autorisées pour les acquisitions de priorité 1.

Ces résultats en termes de nombre d'acquisitions vidées et d'âge moyen de l'information montrent que, pour l'affectation, **MinRegret** est meilleur que **MaxWeight** et que, pour l'ordonnancement, **IdleFill** est clairement meilleur que **EnQueue**, mais que **Scheduler** est seulement légèrement meilleur que **IdleFill**. Finalement, en termes de qualité des plans, **MinRegret-IdleFill** et **MinRegret-Scheduler** semblent être les meilleurs algorithmes.

Le tableau 1 montre le temps de calcul moyen et maximum en secondes (sur tous les appels à la planification, avant chaque fenêtre ou groupe de fenêtres de visibilité) dans les scénarios 1 et 4, uniquement pour un ratio en volume de 1 : ces temps de calcul ne changent pas de façon significative pour des ratios supérieurs (de 2 à 5).

Ces résultats en termes de temps de calcul montrent que **MaxWeight-EnQueue** and **MaxWeight-IdleFill** sont les plus efficaces. Remplacer **MaxWeight** par **MinRegret** multiplie le temps de calcul environ par 5. Remplacer **EnQueue** ou **IdleFill** par **Scheduler** peut le multiplier par 30.

En conséquence, le choix du meilleur algorithme dépend des ressources de calcul disponibles à bord. Si elles sont suffisantes, **MinRegret-IdleFill** semble être le meilleur candidat. Sinon, il peut être remplacé par **MaxWeight-IdleFill**.

8.3 Comparaison entre planification au sol ou à bord

Nous avons ensuite comparé ce qui peut être obtenu en planifiant les vidages au sol comme cela est actuellement fait (planification sur un horizon d'un jour, faisant l'hypothèse de volumes maximaux, fixant des dates précises de vidage et ne permettant aucune flexibilité à bord) et ce qui peut être obtenu en planifiant les vidages à bord (planification des vidages avant chaque fenêtre ou groupe de fenêtres de visibilité, suivie d'une exécution flexible). Dans les deux cas, l'algorithme de planification utilisé est **MinRegret-IdleFill**.

Le tableau 2 montre le nombre d'acquisitions vidées (à gauche) et l'âge moyen de l'information (à droite) pour les priorités 1 et 2 dans les scénarios 1 et 4 (en haut et en bas) en fonction du ratio en volume (de 1 à 5).

Ces résultats montrent l'impact positif de la planification à bord en termes de nombre d'acquisitions vidées qui peut être multiplié par 3 pour les acquisitions de priorité 2 à partir de ratios en volume de 3 ou 4, mais aussi en termes d'âge moyen de l'information qui peut être divisé par 2 pour les acquisitions de priorité 2. Dans certains cas, nous pouvons cependant observer que la planification à bord augmente l'âge moyen de l'information. Ceci est dû à un plus grand nombre d'acquisitions vidées, mais vidées assez tard.

En plus, ces résultats montrent le bon comportement des algorithmes de planification bord qui privilégient les acquisitions de priorité 1 : quels que soient le scénario et le ratio en volume, les acquisitions de priorité 1 sont toutes vidées et, quand le ratio en volume augmente (problème de plus en plus sur-contraint), l'âge moyen de l'information croit moins vite pour les acquisitions de priorité 1 que pour celles de priorité 2.

9 Conclusion

Dans ce papier, nous avons montré comment l'incertitude sur le volume de données généré par les acquisitions peut être gérée en utilisant des algorithmes bord simples et efficaces de planification et d'exécution des vidages qui combinent une recherche gloutonne ou locale, un modèle à base de contraintes et des appels à une librairie générique d'évaluation et de propagation des contraintes.

Nous avons aussi montré les avantages opérationnels qui peuvent être tirés de mécanismes de décision autonome à bord concernant les vidages : augmentation du nombre d'acquisitions vidées, diminution de l'âge de l'information sur les données vidées, utilisation possible de moins de stations de réception ou de moins de fenêtres de visibilité et augmentation possible de la taille des données acquises (fauchée plus large et/ou plus forte résolution de l'instrument d'observation).

Références

[1] E. Aarts and J. Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley & Sons, 1997.

[2] R. Dechter, I. Meiri, and J. Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49 :61–95, 1991.

[3] F. Glover and M. Laguna. Tabu Search. In *Modern Heuristic Techniques for Combinatorial Problems*, pages 70–141. Blackwell Scientific Publishing, 1993.

[4] P. Van Hentenryck and L. Michel. *Constraint-based Local Search*. MIT Press, 2005.

[5] D. Joslin and D. Clements. Squeaky Wheel Optimization. *Journal of Artificial Intelligence Research*, 10 :353–373, 1999.

[6] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[7] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598) :671–680, 1983.

[8] M. Pinedo. *Scheduling : Theory, Algorithms, and Systems*. Springer, 2012.

[9] N. Policella, S. Smith, A. Cesta, and A. Oddi. Generating Robust Schedules through Temporal Flexibility. In *Proc. of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, pages 209–218, Whistler, Canada, 2004.

[10] C. Pralet and G. Verfaillie. Dynamic Online Planning and Scheduling using a Static Invariant-based Evaluation Model. In *Proc. of the 23rd International Conference on Automated Planning and Scheduling (ICAPS-13)*, Rome, Italy, 2013.

[11] C. Pralet and G. Verfaillie. Time-dependent Simple Temporal Networks : Properties and Algorithms. *RAIRO Operations Research*, 47(2) :173–198, 2013.