Open Archive TOULOUSE Archive Ouverte (OATAO)
OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible.

This is an author-deposited version published in : http://oatao.univ-toulouse.fr/
Eprints ID : 18091

To link to this article : DOI: 10.3166/ria.30.35-59
URL : http://dx.doi.org/10.3166/ria.30.35-59

To cite this version : Bonnet, Jonathan and Gleizes, Marie-Pierre and Kaddoum, Elsy and Rainjonneau, Serge *ATLAS : Planification multi-satellite dynamique et temps réel.* (2016) Revue d'Intelligence Artificielle, vol. 30 (n°1-2). pp. 35-39. ISSN 0992-499X

Any correspondence concerning this service should be sent to the repository administrator: staff-oatao@listes-diff.inp-toulouse.fr

ATLAS : Planification multi-satellite dynamique et temps réel

Jonathan Bonnet [1,2], **Marie-Pierre Gleizes** [2], **Elsy Kaddoum** [2], **Serge Rainjonneau** [1]

1. *Institut de Recherche Technologique Saint Exupéry, Toulouse, France*
   prenom.nom@irt-saintexupery.com
2. *IRIT, Université Paul Sabatier - Toulouse III, France*
   Prenom.Nom@irit.fr

RÉSUMÉ. *La planification de mission de constellations de satellites est un problème complexe soulevant d'importants défis technologiques pour les systèmes spatiaux de demain. Le grand nombre de demandes clients et leur arrivée dynamique implique une combinatoire et une dynamique très élevées. Les techniques actuelles présentent plusieurs limites, il est notamment impossible d'adapter dynamiquement le plan lors de sa construction, et les satellites sont traités individuellement de façon chronologique, ce qui minimise l'apport de la constellation.*
Dans cet article, nous proposons de résoudre ce problème difficile et dynamique à l'aide de systèmes multi-agents adaptatifs, profitant de leurs mécanismes d'auto-adaptation et d'auto-organisation. Ainsi, les interactions locales permettent d'atteindre dynamiquement une bonne solution. Enfin, une comparaison avec un algorithme glouton chronologique, couramment utilisée dans le domaine spatial, met en évidence les avantages du système présenté.

ABSTRACT. *Mission planning for a constellation of satellites is a complex problem raising significant technological challenges for tomorrow's space systems. The large numbers of customers requests and their dynamic introduction result in a huge combinatorial search space. Today's techniques have several limitations, in particular, it is impossible to dynamically adapt the plan during its construction, and satellites are planned in a chronological way instead of a more collective planning which can provide additional load balancing.*
In this paper, we propose to solve this difficult and dynamic problem using adaptive multi-agent systems, taking advantage from their self-adaptation and self-organization mechanisms. Thus, local interactions allow to dynamically reach a good solution. Finally, a comparison with a chronological greedy algorithm, commonly used in the spatial domain, highlights the advantages of the presented system.

MOTS-CLÉS : *système multi-agent adaptatif; planification; multi-satellite;*

KEYWORDS: *adaptive multi-agent system; planning; multi-satellite;*

1. Introduction

Une constellation de satellites d'observation de la Terre est un ensemble potentiellement hétérogène de satellites. Elle permet de couvrir une large surface terrestre avec une fréquence de revisite élevée de chaque zone, tout en proposant des images de types différents et en garantissant la robustesse du système. Planifier une mission d'observation pour une telle constellation est une tâche difficile. En effet, beaucoup de paramètres et de contraintes souvent contradictoires sont à prendre en compte : le nombre de satellites et leurs caractéristiques, le volume des demandes des clients et les nombreuses contraintes qui y sont associées (le type de prise de vue, la priorité du client, la qualité demandée, la plage temporelle de validité, etc.) ainsi que les contraintes extérieures (la couverture nuageuse dans le cas de satellites optiques, etc.). La durée de mise en place d'un plan doit être raisonnable, notamment dans le cadre de demandes urgentes, typiquement inférieures à cinq minutes dans un contexte opérationnel, ce qui n'est pas toujours le cas actuellement.

Aujourd'hui les satellites sont planifiés par une approche chronologique qui s'appuie sur des « rendez-vous » réguliers entre les satellites et le centre de contrôle sol : le moment où ces deux entités peuvent communiquer et que le plan de mission est chargé à bord. Les demandes des clients qui arrivent dans le système sont stockées et, avant chaque « rendez-vous », elles sont transmises à un algorithme de planification. Toute demande arrivant après le début de ce processus de planification est stocké pour la prochaine période de programmation. En effet, les algorithmes de planification actuellement utilisés, comme l'algorithme glouton chronologique (Lemaître *et al.*, 2002), ne peuvent pas prendre en compte des changements dynamiques dans un temps relativement court. Le domaine de l'observation par satellite est en pleine mutation. En effet, le nombre de demandes clients augmente chaque année, ainsi que la taille des constellations. De par leur approche, les méthodes actuellement utilisées montrent des faiblesses : elle ne peuvent gérer une telle complexité. Enfin, cette évolution apporte de nouveaux utilisateurs qui ont besoin d'un *feedback* plus rapide sur l'état de leur demande. Pour améliorer cette situation, de nouvelles approches sont nécessaires. Nous proposons l'utilisation de méthodes basées l'auto-adaptation et l'auto-organisation. En effet, de telles méthodes apportent plus de flexibilité, de robustesse et d'adaptation temps réelle. Ainsi, les plans de mission peuvent être calculés de manière dynamique, tout en traitant d'importants volumes de données et de donner plus rapide des informations sur l'état de la planification aux utilisateurs.

Les systèmes multi-agents coopératifs (Gleizes, 2012) et leur capacité à prendre en compte un grand nombre d'entités et de contraintes sont une bonne approche. En effet, ils fournissent naturellement l'auto-adaptation et l'auto-organisation nécessaire à ce problème. Dans ce travail nous nous appuierons sur AMAS4Opt *(Adaptive Multi-Agent System For Optimization)* (Kaddoum, 2011), un modèle d'agent générique qui fournit des patrons de conception pour résoudre des problèmes d'optimisation combinatoire à l'aide des systèmes multi-agents coopératifs.

Nous supposons connu l'algorithme qui permet à un satellite de construire son plan de mission à partir d'un ensemble de prises de vues qui lui est affecté. Dans cet article nous nous intéressons au problème de la répartition dynamique des requêtes dans une constellation de satellites. Cette répartition consiste à planifier les demandes des clients sur les différents satellites, tout en respectant les contraintes et en assurant une charge équilibrée pour chaque satellite.

La partie 2 de cet article décrit le problème et présente son contexte. La partie 3 développe le fonctionnement du système multi-agent ATLAS *(Adaptive saTellites pLanning for dynAmic earth obServation)*. Enfin, la partie 4 présente une évaluation du système ATLAS ainsi que sa comparaison à un algorithme Glouton Chronologique.

2. Planification multi-satellite

Cette partie décrit le problème puis présente un résumé des méthodes de résolution actuellement utilisées.

2.1. Description du problème

En nous appuyant sur les travaux de (Bensana, Verfaillie, 1999) et (Bonnet, 2008), nous allons tout d'abord décrire formellement notre problème. Dans la partie 3, nous nous appuierons sur cette description pour *agentifier* le système.

2.1.1. Constellation de satellites

Une constellation est un ensemble potentiellement hétérogène de satellites, $Sat = \{s_1, s_2, ..., s_n\}$, dans lequel chaque satellite s_i a ses propres caractéristiques :

– une trajectoire légèrement elliptique autour de la terre,

– une gestion d'énergie,

– une capacité mémoire,

– une charge utile, ici des instruments d'observation optique ou radar, et leurs attributs propres,

– un système de contrôle orbital et de gestion d'attitude, qui permet au satellite de contrôler son pointage *(vers la zone à observer)*.

2.1.2. Requêtes

Une requête est une commande client. Nous noterons l'ensemble des requêtes à planifier $R = \{r_1, r_2, ..., r_m\}$, avec r_i une requête définie par un ensemble de données :

– son type,

– une date de soumission,

– un intervalle de temps qui correspond à la période durant laquelle la requête est valable *(quelques heures à quelques jours, voire plusieurs semaines)*,

– une zone géographique,
– une priorité donnée par le client,
– w, le taux de couverture nuageuse toléré, $0 <= w <= 1$,
– un ensemble de mailles, $M^i = \{m_1^i, m_2^i, ..., m_p^i\}$, associé à la requête r_i.

La figure 1 représente un accès de visibilité *A* sur la maille *M1*, par un satellite. Ici, l'acquisition sera réalisée durant le créneau *C*.

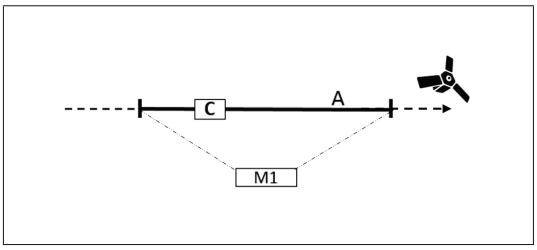


Figure 1. Représentation d'un accès de visibilité et d'un créneau de réalisation

2.1.3. Contraintes dures et souples

Deux catégories de contraintes sont à prendre en compte : les contraintes dures (la plage de validité ou le type d'image souhaité par le client par exemple) et souples (le taux de couverture nuageuse). L'ensemble C_h de contraintes dures doit être satisfait, les contraintes souples, C_s peuvent être relâchées.

2.1.4. But

L'objectif du problème est donc de trouver pour chaque satellite un ensemble de mailles, avec pour chaque maille une date d de couverture. Les mailles d'une même requête ne seront pas obligatoirement attribuées au même satellite. Les satellites devront assurer la couverture tout en :

– satisfaisant l'ensemble des contraintes dures C_h,
– maximisant le nombre de contraintes souples C_s satisfaites,
– maximisant le nombre de mailles à satisfaire.

2.2. État de l'art

De par le grand nombre d'entités et de contraintes, ainsi que sa dynamique, le problème de planification de mission d'observation est un problème d'optimisation complexe. Dans la littérature, il existe un certain nombre de propositions, qui se basent sur des méthodes d'optimisation classique pour résoudre ce problème.

Le premier constat est qu'il est difficile de prendre le problème dans sa totalité. (Lemaître *et al*., 2002) explique ainsi que le problème général est trop complexe pour être mathématiquement modélisé et résolu, et qu'il est nécessaire de le simplifier, en

enlevant certaines contraintes, notamment en raccourcissant l'horizon de planification et en empêchant certaines manœuvres des satellites. Ce problème simplifié est correct, même s'il mène à des solutions sous-optimales. Cependant, il permet une modélisation plus facile. Par exemple, le problème du **sac à dos multidimensionnel** (Vasquez, Hao, 2003) et (Grasset-Bourdel *et al*., 2011) ou au problème du **voyageur de commerce** (Mancel, 2004) sont de bonnes modélisations, et permettent d'appliquer des méthodes d'optimisation classique. Le sac à dos représentant la capacité mémoire des satellites, et le chemin à trouver entre les villes modélisant l'enchaînement entre les acquisitions. Nous regroupons ici les approches les plus couramment utilisées dans le domaine, et les trois critères présentés dans la section 1 seront utilisés pour les comparer.

– la *scalabilité* : capacité à traiter des problèmes de grandes tailles,

– le dynamisme : possibilité d'ajouté de nouvelles requêtes,

– la capacité de donner une réponse en temps réel (si l'algorithme est arrêté, donnera-t-il une solution cohérente ?).

Les méthodes exactes garantissent de trouver la solution optimale, si elle existe. Une modélisation simple du problème permet d'utiliser de l'appréhender comme un **Problème de Satisfaction de Contraintes** ((Bensana *et al*., 1996), (Dago, 1997), (Bouveret, Lemaître, 2006)), et ainsi d'utiliser des outils performants tel que *ILOG Solver* pour les traiter. Mais la résolution de tels **CSP** est très couteuse en temps de calcul donc peu adaptée pour ce problème où le temps est limité. La **Programmation Dynamique** est utilisée pour décomposer le problème en sous-problèmes plus simples. La difficulté de tels algorithmes vient de la possibilité de découper récursivement le problème initial. Comme le montre (Grasset-Bourdel, 2011), si on applique l'algorithme linéairement, son déroulement va rapidement utiliser la majorité des ressources mémoires et le rendre inapplicable en cas de requêtes stéréoscopiques (requêtes qui demandent des prises de vue d'un même endroit, mais avec un décalage temporel). (Mansour, Dessouky, 2010) émettent la même critique concernant des algorithmes construits sur une recherche de type **Séparation et Évaluation**. Même si toutes les solutions ne sont pas explorées car l'algorithme se sert des propriétés du problème pour guider la recherche, le déroulement d'une telle méthode est extrêmement coûteux (en espace mémoire et en temps de calcul). Ces méthodes ne sont donc pas adaptées pour notre problème. Le grand nombre de contraintes et de variables font que ces approches sont extrêmement lentes, et que tout changement impose de reconstruire l'arbre de recherche.

Les méthodes approchées peuvent être appliquées pour pallier ces différents inconvénients. En s'appuyant sur une fonction heuristique, elles permettent de trouver une solution d'une bonne qualité en un temps raisonnable.

L'algorithme le plus utilisé, car rapide à l'exécution et facile à mettre en œuvre, est l'algorithme **Glouton** dont il existe diverses déclinaisons fonction de la manière dont sont parcourues les données d'entrée que sont les requêtes d'acquisition (l'algorithme est souvent nommé par son terme anglophone de Greedy Algorithm). Le principe de base de ce type d'algorithme est que lorsqu'un choix est fait, il n'est jamais remis en

cause, ce qui permet un parcours très rapide des données d'entrée. On trouvera parmi les versions les plus citées de cette famille d'algorithme :

le **Glouton Chronologique** (Bianchessi *et al*., 2007) : la descente glouton se fait chronologiquement, c'est-à-dire qu'à l'instant où l'on se situe, on se pose la question de l'acquisition la plus pertinente (la plus prioritaire en général), on la place temporellement en tenant compte des règles d'enchaînement des acquisitions et des diverses autres contraintes prises en compte, puis on passe à la date de fin de la dernière acquisition planifiée et on recommence.

le **Glouton Hiérarchique** (Wang *et al*., 2011) : les requêtes éligibles sont classées sur l'horizon de planification en fonction d'un certain nombre de critères d'évaluation dont la priorité, la qualité image associée au passage, la prévision météo, *etc*.. Ensuite, la liste est dépilée et chaque requête est placée au meilleur instant en tenant compte d'un certain nombre de critères liés à la qualité du produit souhaité.

le **Glouton Stochastique Itéré** (Frank *et al*., 2001) : cette version est plutôt réservée à des calculateurs de faible puissance ou à des temps de réaction très courts. Elle consiste, partant d'un état donné du plan qui est dit « de référence » (qui peut être éventuellement vide), à prendre aléatoirement les requêtes, à essayer de les placer (souvent dans un horizon court, typiquement moins d'un quart d'orbite) et d'évaluer si le plan obtenu est meilleur que le plan que l'on avait précédemment. Si c'est le cas, le nouveau plan devient le plan de référence. Puis, dans tous les cas, on recommence en essayant de placer les requêtes sur la base d'une autre ordre déterminé aléatoirement. On laissera ainsi tourner l'algorithme en fonction du temps dont on dispose.

De plus, cet algorithme sert souvent de base de travail, comme par exemple pour (Lemaître *et al*., 2002). Les méthodes de **Recherches Locales** tel que le **Recuit Simulé** ((Wu *et al*., 2014) ou (Globus *et al*., 2004)) ou les **Recherches Tabou** ((Bianchessi *et al*., 2007) ou (Cordeau, Laporte, 2005)) sont aussi applicables au problème. Ces algorithmes utilisent différentes heuristiques pour chercher dans le voisinage de la solution courante une meilleure solution. Mais il n'existe pas ici d'algorithme standard et, pour produire la meilleure solution possible, beaucoup de paramètres doivent être adaptés. Ainsi l'algorithme devra être retravaillé pour toute modification du problème. Enfin, les **Algorithmes Génétiques** tels que ceux présentés par (Globus *et al*., 2003), (Mansour, Dessouky, 2010) ou encore (Wolfe, Sorensen, 2000) sont aussi étudiés, mais la modélisation est difficile et ces algorithmes sont assez lent (plusieurs heures).

Tous ces algorithmes, même s'ils produisent des résultats plutôt satisfaisants présentent des limites. Tout d'abord, comme nous l'avons évoqué, ils dépendent fortement de l'heuristique qui guide la recherche. De plus, en cas d'ajout de requêtes pendant le déroulement du processus de planification, il faut reprendre le processus depuis la première opportunité de placement de la requête ajoutée. Enfin, ces algorithmes sont essentiellement conçus pour planifier la mission d'un satellite. Ainsi, ils ne sont pas adaptés pour gérer un important volume de requêtes et de contraintes. Le tableau 1 note chaque méthode présentée ici par rapport aux critères de comparaison. Les notes sont comprises entre - - (le pire cas) et + + (le meilleur cas).

Tableau 1. Comparaison des méthodes

Méthodes	Scalabilité	Dynamisme	Réponse temps réel
Résolution CSP	-	-	-
Programmation Dynamique	- -	-	-
Séparation et Évaluation	-	-	-
Algorithme Glouton	- -	-	+
Recherches Locale	-	+	+
Algorithme Génétique	-	-	+

Récemment, de nouvelles approches ont pour objectif d'optimiser la mission en planifiant une partie du plan à bord des satellites. (Bonnet, Tessier, 2009) proposent une approche basée sur les systèmes multi-agents pour la planification à bord des satellites d'une constellation. Dans cette approche, les satellites sont définis comme étant des agents communicants via des liaisons inter-satellites. L'objectif des satellites est de négocier entre eux la répartition des demandes de prises de vues reçues du sol permettant de maximiser le nombre de demandes satisfaites. Par l'utilisation des liaisons inter-satellites, l'applicabilité de cette approche requiert un certain nombre de conditions (taille de la constellation, distance entre satellites, etc.). (Maillard, 2015) construit des plans « flexibles ». Les plans créés ne tiennent pas compte des contraintes énergétique. Le satellite va acquérir les requêtes de haute priorité et va tenter d'acquérir les requêtes de plus faible priorité. Cette approche est proposée car la gestion énergétique est difficile a prévoir au sol, ainsi les plans générés prennent toujours les contraintes au pire cas possible, ce qui engendre des plans sous-optimaux.

2.3. La théorie AMAS et le modèle AMAS4Opt

Dans le travail présenté, nous proposons l'utilisation des systèmes multi-agents adaptatifs pour la planification au sol de la mission d'une constellation de satellites. Dans la théorie **AMAS** (pour *Adaptive Multi-Agent System*) (Gleizes, 2012), les agents sont définis comme des entités autonomes, adaptatives et coopératives qui possèdent leur propre objectif local à atteindre. La coopération locale entre agents permet au système de s'auto-adapter pour pouvoir réaliser la fonction pour laquelle il a été conçu. Cette fonction est nommée fonction *adéquate*. La coopération est définie comme la capacité des agents à travailler ensemble pour réaliser un but commun. Cela implique que toute activité entre agents est complémentaire et qu'il existe des liens de dépendance de solidarité entre les agents. Pour tenir compte de la dynamique de l'environnement, les agents possèdent des mécanismes leur permettant, de manière autonome, de modifier leur organisation.

Les problèmes complexes d'optimisation sous contraintes sont formalisés comme un ensemble d'entités régis par un ensemble de contraintes. Dans les systèmes multi-agents, les entités sont couramment représentées par des agents. En fonction des interactions en temps réel que le système a avec son environnement, l'organisation entre les agents émerge et constitue la solution du problème.

Pour faciliter le développement d'agents coopératifs capables de résoudre des problèmes d'optimisation en se basant sur l'approche par AMAS, le modèle **AMAS4Opt** a été proposé par (Kaddoum, 2011). Ce modèle fournit les patrons de conception de deux rôles génériques d'agents coopératifs : le rôle « contraint » et le rôle « service ».

Le rôle « contraint » concerne les agents qui, a un instant donné, ont un problème et vont requérir de l'aide d'autres agents : ceux sous le rôle « service ». Les agents sous le rôle « contraint » sont considérés comme les initiateurs de la résolution. En effet, ils expriment le problème, et en relaxant leurs contraintes, la solution peut être atteinte. Pour relaxer leurs contraintes, ces agents doivent interagir avec les agents sous le rôle « service » en demandant leur service. Dans un problème d'optimisation, différents agents sous le rôle « contraint » peuvent demander le service du même agent sous le rôle « service ». En tant qu'agent coopératif, ce dernier va aider le plus critique. Cette criticité est donc le moteur de la coopération entre agent. De plus, un agent peut posséder les deux rôles, en fonction du contexte dans le lequel il se trouve. Nous avons utilisé ce modèle pour concevoir les agents et l'architecture générale du système et nous en proposons ici une amélioration.

3. Le système ATLAS

Dans cette partie nous présentons les agents et leur rôle, puis la mesure de la criticité utilisée par les agents ayant le rôle « contraint ». Enfin, nous introduisons la mesure du coût représentant la criticité des agents ayant le rôle « service ».

3.1. Les agents

A l'aide de la description du problème proposée dans la partie 2.1, et en se basant sur AMAS4Opt nous avons identifié neuf entités, parmi lesquelles :

– **trois types d'agents coopératifs** : le type agent requête, le type agent maille et le type agent satellite (leur comportement sera détaillé plus tard dans cet article),

– **trois types d'entités actives** : la couverture nuageuse, l'éphéméride solaire et les stations de télé-déchargement (ces entités n'ont pas de but à satisfaire),

– **trois types d'entités passives** (ressources du système) : la mémoire des satellites, leur batterie et un module de calcul de trajectoire et d'attitude orbitales.

Dans la version actuelle du système ATLAS, la décomposition des agents Requête en agents Maille n'est pas prise en compte, à cause du manque de données réelles. Ainsi les interactions entre ces deux types d'agents ne sont pas implémentées. Néanmoins, ce manque n'entrave pas la résolution du problème. En effet les agents Requêtes n'influencent qu'une partie de la criticité des agents Mailles. Ainsi, ces derniers peuvent quand même effectuer leur comportement et chercher des services vers les agents Satellite et être planifié.

L'utilisation du modèle AMAS4Opt permet de définir le comportement et les interactions des agents : les agents satellites prennent le rôle « service », et les agents requêtes et mailles le rôle « contraint ». Nous présentons ci-dessous les agents coopératifs Satellite et Maille, en indiquant leur objectif local, les entités du système avec lesquelles ils seront en interaction au cours de leur processus de résolution et les agents avec lesquels ils seront amenés à négocier (échanges d'information, demandes de services, etc.), dans le but d'atteindre leur objectif. Le tableau 2.

Tableau 2. Les agents coopératifs

Agent	Satellite	Maille
Rôle	« Service »	« Contraint »
Objectif	acquérir des prises de vues	être planifiée
Interactions	module de gestion trajectoire couverture nuageuse batterie mémoire éphémérides	éphémérides
Négociations	agents Maille	agents Satellite

Tableau 3. Les agents coopératifs et leurs différents messages

Agent	Message	Signification
Satellite	estimationSlot(Cost c)	Estimation du coût de la planification
	confirmSlot(Boolean b)	Confirmation ou non de la planification
Maille	askForASlot(Access a)	Demande de planification
	askConfirmation(Access a)	Demande de confirmation pour l'accès
	informRequest()	Transmet des informations à sa requête

3.2. Comportement des agents

Le système ATLAS assure la planification de la constellation grâce à la coopération de ses agents. Cette coopération est assurée via l'échange de messages entre les agents et est guidée par deux indicateurs : la criticité et le coût. C'est cette coopération qui permet de produire une planification respectant les critères suivant : un maximum de requêtes planifiées et un équilibre au sein de la constellation (les satellites doivent avoir tous la même charge de travail). Le fonctionnement du système ATLAS est basé sur la répétition par les agents du cycle « Perception - Décision - Action ». La phase de décision est l'étape centrale. En fonction de ses perceptions et de son état, l'agent choisit l'action à réaliser.

Le diagramme de séquence présenté dans la figure 2 décrit un exemple d'interactions entre un agent Maille et deux agents Satellite. Dans cet exemple, l'agent Maille peut être planifié par les deux agents Satellite. Les différents messages émis par les agents sont présentés dans la table 3. Le diagramme peut être décrit comme ceci :

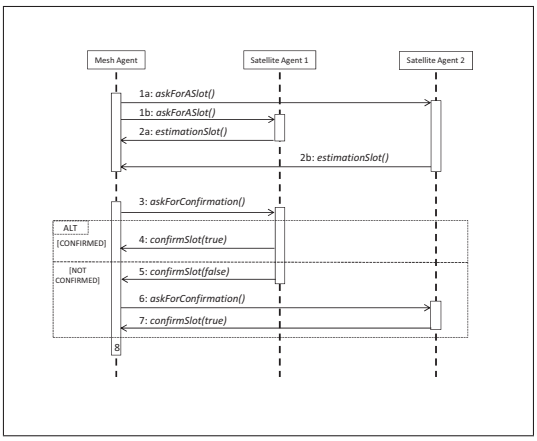


Figure 2. Diagramme de séquence

– Premièrement, l'agent Maille envoie un message *askForASlot()* aux deux agents Satellite (actions 1a et 1b).

– a la réception de ce message, les agent Satellite estime la difficulté de planifier l'agent Maille et répondent via un message *estimationSlot()* (actions 2a et 2b).

– Une fois que tous les agents Satellite ont répondu, l'agent Maille choisi **coopérativement** le coût le plus faible et émet un massage de confirmation : *askForConfirmation()* (action 3). Les coûts non sélectionnés sont gardés en mémoire. Dans cet exemple, l'agent Satellite 1 est sélectionné.

– L'agent Satellite peut confirmer ou non (*confirmSlot()*). Si oui (action 4), l'agent Maille est planifié.

– En l'absence de confirmation (action 5), l'agent Maille sélection un autre agent Satellite *askForConfirmation()* (action 6).

– Une confirmation est émise (action 7).

– Finalement, l'agent Maille est planifié (point 8).

Comme nous l'avons précédemment mentionnée, une importante contrainte est que les requêtes clients n'arrivent pas toutes en même temps. Au contraire des approches actuelles, ATLAS est capable de gérer cette dynamique. En effet, les agents, de par leur comportement coopératif, sont capables de planifier dynamiquement des requêtes ajoutées dans le système. Les agents Satellite vont ainsi privilégier les agents Mailles les plus « critiques ». Ainsi, si un agent Maille de faible priorité était planifié par un agent Satellite et qu'un nouvel agent très urgent arrive, l'agent Satellite va déplanifier le moins critique, qui va recommencer son cycle, et chercher une nouvelle place. Enfin, les interactions entre agents étant locales, le système ATLAS permet de fournir à tout moment une solution valide. En effet, le plan n'est pas construit chronologiquement et toutes modification (ajout de requêtes par exemple) va entrainer des

perturbations locales qui ne remettent pas en question l'ensemble du plan. Le comportement coopératif des agents peut être décrit via ces algorithmes 1 et 2.

Algorithme 1 : Algorithme de l'agent Maille

```
1  while state is not planned do
2      for all my accesses a do
3          askForASlot(a);
4      end
5      while potentiel satellite exist do
6          cooperatively select a satellite;
7          askConfirmation(a);
8          wait answer;
9          if success then
10             state = planned;
11         end
12     end
13 end
14 if message cancel then
15     state = not planned;
16     try to be planned;
17 end
```

Algorithme 2 : Algorithme de l'agent Satellite

```
1  for all messages received do
2      if message is askForASlot then
3          estimate cost;
4          answer;
5      end
6      if message is askConfirmation then
7          if no conflict then
8              plan the mesh;
9              inform success;
10         else
11             choose more critical mesh;
12             cancel the other;
13         end
14     end
15 end
```

3.3. La criticité et le coût

3.3.1. La criticité

Dans ATLAS, le degré de non satisfaction des agents mailles est représenté par la criticité. (Bouziat *et al*., 2014) définissent la criticité comme « la distance qui sépare l'état courant [de l'agent], de l'état dans lequel son objectif local est atteint ».

Les règles de comportement local des agents satellites sont donc écrites pour favoriser les agents mailles les plus *critiques*. Cette criticité, qui indique donc le degré de non satisfaction de l'agent maille, est représentée par un ensemble de valeurs ordonnées. Dans la version actuelle du système, cette mesure comprend deux critères. La criticité d'un agent Maille peut être décrit par un tuple $Cm = \langle P, Ra \rangle$.

Le premier critère concerne l'urgence de la planification de la maille : P. Celle-ci est définie par le client au moment de sa commande. En cas d'égalité de ce critère, les agents satellites utilisent le nombre d'accès de visibilité restants : Ra (deuxième critère). Ainsi, un agent maille qui n'est visible qu'une seule fois par les satellites sera prioritaire par rapport aux autres agents mailles visibles plusieurs fois. En effet, si l'agent satellite privilégie un agent maille ayant plusieurs accès de visibilité sur un agent maille en ayant un seul, ce dernier ne pourra pas être planifié.

L'évolution de la criticité d'un agent maille augmente si le nombre de ses créneaux de visibilité diminuent. Finalement, elle est nulle lorsque l'acquisition de l'agent maille est planifiée.

3.3.2. Le coût

La criticité, telle qu'elle est définie dans le modèle AMAS4Opt, ne permet pas une coopération *totale* entre les agents : la coopération est seulement assurée par les agents ayant le rôle « contraint ». En effet, elle ne permet qu'aux agents ayant le rôle « service » de savoir quels agents ayant le rôle « contraint » sont prioritaires. Ainsi, un agent ayant le rôle « contraint » ne peut pas privilégier un agent ayant le rôle « service » sur un autre. Nous avons donc ajouté la notion de coût pour résoudre ce problème, et rendre les agents ayant le rôle « contraint » plus coopératifs.

Dans ATLAS, le coût est un indicateur sur la difficulté pour l'agent satellite à prendre en compte et planifier un agent maille. Il est calculé et retourné à l'agent maille ayant émis un message de demande de couverture. L'agent maille va donc recevoir un coût pour chaque demande émise aux agents satellites. Pour favoriser la coopération, un agent maille privilégie l'agent satellite lui répondant avec le coût le plus faible. Le coût est représenté par le triplet $C = \langle Cm, Ml, D \rangle$. Voici des éléments qui favorisent un coût élevé :

– la nécessité d'adapter le plan, en déplaçant des éléments planifiés. Le coût comporte dans ce cas là la criticité de l'agent Maille à annuler Cm,

– une charge mémoire importante (beaucoup de mailles sont déjà planifiées par le satellite) Ml,

– un grand nombre de messages de demande de planification reçus D.

4. Résultats et discussions

Dans cette partie, nous présentons les résultats obtenus par le système multi-agent ATLAS pour planifier une constellation de satellites. Pour le tester nous avons développé un générateur de *scenarii*. Ces différents *scenarii* permettent de présenter l'intérêt du système ATLAS et de le comparer avec l'algorithme Glouton Chronologique couramment utilisé dans le domaine spatial.

4.1. Le générateur de solutions

Lors de l'établissement d'un plan de mission, les demandes des clients sont prétraitées pour conduire à une liste de mailles correspondantes à acquérir. Cette phase de prétraitement ne concerne pas ATLAS. Chacune de ces mailles terrestres est visible par un ou plusieurs satellites de la constellation. Chaque maille peut donc être acquise par différents satellites à différents instants, ce qui augmente le nombre de solutions possibles.

Nous avons donc développé et utilisé un générateur pour construire des plans de mission *complets*. Dans un tel plan tous les créneaux sont occupés par des acquisitions. Ainsi, nous sommes sûrs qu'il existe une solution optimale (correspondante au plan complet). Pour éviter de fournir une liste dans laquelle chaque requête n'a qu'un seul accès de visibilité, le générateur ajoute à chaque requête un nombre aléatoire d'accès de visibilité vers d'autres satellites et/ou à d'autres créneaux temporel. Ce nombre est pondéré par un facteur d'accessibilité sur lequel l'utilisateur peut influer, et ainsi produire différents *scenarii*. Ainsi, une liste de requêtes ayant toutes N accès de visibilité. La solution optimale et donc plus complexe à obtenir parmi la multitude de solutions possibles. La figure 3 montre le fonctionnement du générateur.

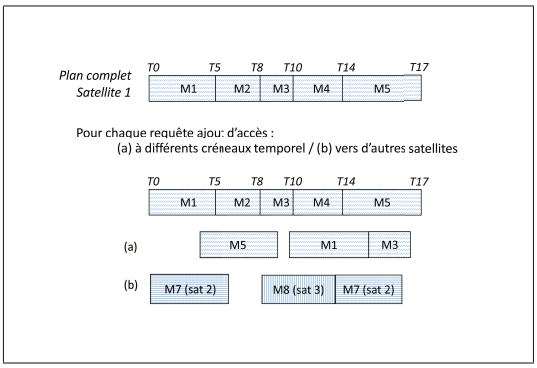


Figure 3. Évolution du nombre de messages par cycle

Ce générateur permet de générer un nombre conséquent de *scenarii* représentatifs de la combinatoire du problème réel. Cette génération a été validée par des experts du

domaine spatial. Nous pouvons ainsi tester le système ATLAS, prouver sa validité, sa robustesse et son passage à l'échelle.

Dans les différentes expérimentions présenté, nous avons utilisé neuf *scenarii* présentés dans le tableau 4. Dans chaque scénario un satellite supplémentaire est rajouté. De plus, le degré d'accessibilité des requêtes par les satellites et aussi progressivement augmenté. Ainsi, plus la constellation est grande, plus les requêtes sont visibles. Cela permet de complexifier la résolution : en augmentant le nombre d'accès possibles, le nombre de solutions possibles augmente aussi. Nous définissons un indicateur sur la solution : le pourcentage de requêtes planifiées. La solution proposée par le générateur est optimale sur ce critère, car toutes les requêtes sont planifiées : le taux de planification est de 100%. Ainsi il est facile de comparer facilement les deux systèmes, un taux élevé nous indiquant donc que la solution est de bonne qualité.

Tableau 4. Description des scenarii

S	Nombre de Satellites	Nombre de Mailles	Nombre d' Accès	Facteur d'accessibilité (%)
1	2	173	481	50
2	3	263	773	55
3	4	341	1041	60
4	5	424	1545	65
5	6	514	2157	70
6	7	580	2661	75
7	8	677	3748	80
8	9	777	5128	90
9	10	861	5677	100

4.2. Expérimentations sur ATLAS

Nous allons tout d'abord nous intéresser à l'apport de l'auto-organisation au sein du système ATLAS. Pour cela nous allons comparer deux versions d'ATLAS. Dans la première version, nommée ATLAS 1, tous les comportements d'auto-organisation ne sont pas implémentés : un agent Satellite ne peut déplanifier un agent Maille. Au contraire, dans la seconde version (ATLAS 2), les agents Satellite ont un meilleur comportement coopératif : ces derniers peuvent annuler un agent Maille pour laisser la place à un autre agent de plus forte criticité. Deux critères sont considérés pour comparer les systèmes : le pourcentage de chaque catégorie d'agents Maille planifiés et l'évolution de la criticité globale du système durant la résolution. Enfin, nous analyseront l'évolution du nombre de messages échangés ainsi que le nombre de cycles décision en fonction de la taille des problèmes à résoudre.

Tableau 5. Dynamic consideration with ATLAS 1 and 2

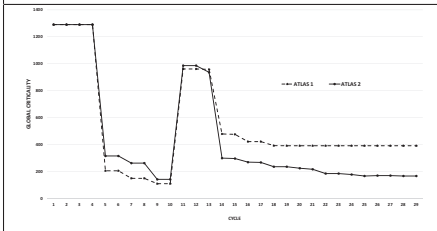
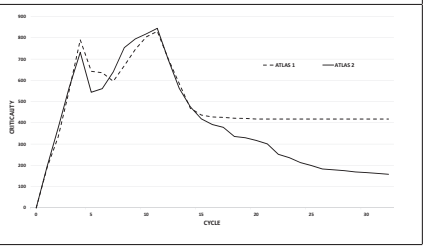
	ATLAS 1	ATLAS 2
Stability	17	24
% Routine	68	91
% Normal	92	95
% Urgent	89	95
% Total	87	94

4.3. Apport d'un comportement dynamique

Dans cette première comparaison, ATLAS 1 et 2 sont testé sur un scénario comportant 5 satellites et 400 mailles. Toutefois, afin de souligner l'apport des mécanismes d'auto-adaptation, l'arrivée de nouvelles mailles dans les deux systèmes est contrôlée. En effet, toutes les mailles de priorité « normale » et « routine » sont disponibles au début de l'exécution, mais les plus prioritaires, les « urgentes » sont insérés au cycle numéro 10. Le tableau 5 présente les résultats. ATLAS 2 planifie plus de mailles « urgentes » (95%) que ATLAS 1 (seulement 89%) et le pourcentage final de mailles planifiées est aussi plus important : 94% contre 87%. La stabilité (*i.e.* le nombre de cycles avant que l'Observateur exposer la solution) est atteint au cycle de 17 dans le cas d'ATLAS 1 contre 24 pour ATLAS 2. Cela est expliqué par le fait que lorsque les mailles « urgentes » arrivent, la plupart des créneaux des plans sont occupés dans les deux systèmes, mais les agents Satellite d'ATLAS 2, grâce à leurs mécanismes d'auto-adaptation, peuvent réorganiser leur plan pour planifier les demandes « urgentes », au contraire d'ATLAS 1 qui ne peut libérer de l'espace. Cette expérience montre l'importance et la contribution des mécanismes d'auto-adaptation : plus de demandes des clients peuvent être planifiées, et les demandes « urgentes » sont dynamiquement prises en compte et planifiées. En outre, le nombre de cycles nécessaires pour atteindre la stabilité est faible : ATLAS 2 s'adapte rapidement aux perturbations.

4.3.1. Evolution de la criticité globale

Tableau 6. Evolution de la criticité globale

Insertion contrôlée	Insertion aléatoire
	

La première figure du tableau 6 montre l'évolution de la criticité globale pour l'exécution des deux systèmes dans le cas où l'arrivée de mailles « urgentes » est

contrôlée. Cette criticité globale n'est qu'un indicateur : elle n'est pas utilisée durant la résolution. Elle est calculée en additionnant les priorités client des agents Maille non-plannifiés à chaque cycle d'exécution: $Gc_i = \sum_{j \in M} P_j$, où i est le cycle, M l'ensemble des agents Maille non planifiées et P_j est la priorité du client de l'agent de Maille j. L'augmentation au cycle 11 est causée par l'insertion des demandes « urgentes ». ATLAS 1 se stabilise au cycle 17 tandis que la criticité d'ATLAS 2 continue de diminuer grâce à ses mécanismes d'auto-adaptation : plus d'agents Maille sont planifiés.

Finalement, La seconde figure du tableau 6 compare l'évolution de la criticité globale dans le cas où l'arrivée de l'ensemble des mailles n'est pas contrôlée. Cette situation est plus réaliste : dans les systèmes opérationnels les demandes clients peuvent arriver à tout moment. Dans un premier temps, nous voyons que la criticité des deux systèmes augmente. Cette rapide augmentation est expliquée par le fait que les mailles arrivent juste dans le système. A partir du cycle 4, les premiers agents Maille sont planifiés : la criticité diminue durant deux cycles. Entre les cycles 7 et 15, l'évolution repart à la hausse en raison des demandes qui continuent d'arriver. Dans le cas d'ATLAS 2, la criticité est plus élevé parce que le système déplanifie des agents Maille et donc libère de l'espace pour des agents Mailles plus critiques. Enfin, à partir du cycle 15, la criticité globale n'évolue plus pour ATLAS 1 : le système ne peut plus rien planifier, alors qu'elle continue à décroître pour ATLAS 2. Dans cette expérience aussi ATLAS 2 est ses comportement d'auto-adaptation est plus efficace. Grâce à leurs mécanismes d'auto-adaptation et d'auto-organisation, les agents Satellite adaptent leur plan et ainsi planifient un maximum d'agent Mailles permettant ainsi de faire diminuer la criticité globale du système.

4.3.2. Communications et cycles de résolution

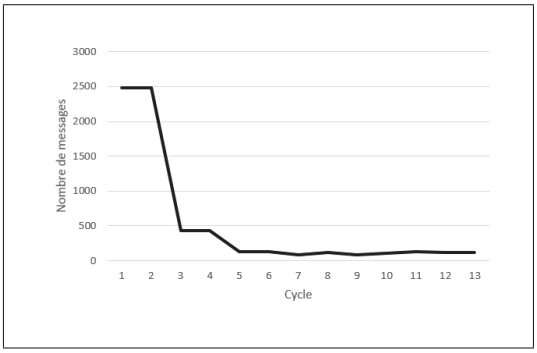


Figure 4. Évolution du nombre de messages par cycle

La figure 4 permet de voir l'évolution du nombre de messages au cours des cycles de décision, et ce pour la résolution de S1. On note deux fortes décroissances entre les cycles 2 - 3 et 4 - 5. Les agents mailles initialisent le système, puis, ayant tous émis des requêtes vers les agents satellites au cycle 1, ce sont ces derniers qui traitent

les demandes lors du cycle 2, les agents mailles étant en attente de réponses. Cinq fois moins de messages sont envoyés au cycle 3, en effet, à ce stade ce sont les agents mailles qui traitent les réponses à leurs requêtes et demandent confirmation au satellite qu'elles choisissent de favoriser. On note le même phénomène, un palier puis une forte diminution aux cycles 3, 4 et 5, mais avec moins de messages car davantage d'agents mailles sont déjà dans un état « planifiés », ils ne communiquent donc plus. En convergeant rapidement vers une solution, les agents n'encombrent pas le système avec un trop-plein de communication.

La table 7 permet de visualiser le nombre de messages émis par les agents mailles et le nombre de cycles de résolution nécessaires pour des constellations de plus en plus grandes. Un satellite est ajouté à chaque nouvelle constellation. La colonne S de la table 7 donne le nombre de satellites pour chaque constellation. Les différents *scenarii* produits par le générateur durent 100 unités de temps, ce qui correspond à une moyenne de 86 requêtes pour chaque satellite, la croissance du nombre de requêtes est donc linéaire. Pour simplifier les expérimentations, nous avons décidé que chaque requête correspond à une maille. La proportion d'accès de visibilité pour chaque maille est donnée par un facteur en entrée du générateur, que nous avons augmenté en fonction du nombre de satellites, afin d'accroître la complexité du problème : rajouter des accès de visibilité revient à accroître le nombre de solutions possibles. L'augmentation de la complexité, et donc du nombre de messages émis, n'empêche pas le système de toujours converger vers une solution en un faible nombre de cycles et dans un temps de calcul restreint.

Tableau 7. Nombre de messages échangés et nombre de cycles de résolutions

S	Mailles	Accès	Messages	Cycles
2	175	546	780	7
3	259	792	1 125	11
4	331	1 177	1 644	13
5	417	1 518	2 126	13
6	518	2 146	2 921	15
7	605	2 478	3 346	17
8	694	3 470	4 526	17
9	767	4 286	5 540	17
10	860	5 738	7 384	20

4.4. Comparaison avec un algorithme Glouton Chronologique : ChronoG

4.4.1. ChronoG

Pour analyser la qualité des solutions d'ATLAS, nous le comparons à la solution standard qui est actuellement utilisée par les références du spatial : l'algorithme Glouton Chronologique, nommé ici **ChronoG**. ChronoG traite la constellation un satellite à la fois. Nous supposerons aussi qu'une maille sera acquise par un seul satellite.

L'algorithme suivant est donc répété pour chaque satellite : pour chaque pas de temps t de la durée de planification parcourue chronologiquement, ChronoG regarde si une maille est planifiée ou si l'espace est libre.

– Si une maille est planifiée, ChronoG passe à $t+1$, sinon, ChronoG vérifie si une maille peut être placée à cet instant en déterminant si la maille en question possède un accès de visibilité qui englobe la date courante t.

– si plusieurs mailles sont possibles, ChronoG utilise une heuristique, détaillée ci-dessous, pour sélectionner une maille.

– en cas d'égalité, ChronoG choisira la maille ayant la plus petite différence entre la fin de son accès de visibilité et t.

Finalement, ChronoG planifie la maille choisie en réservant la durée nécessaire à son acquisition.

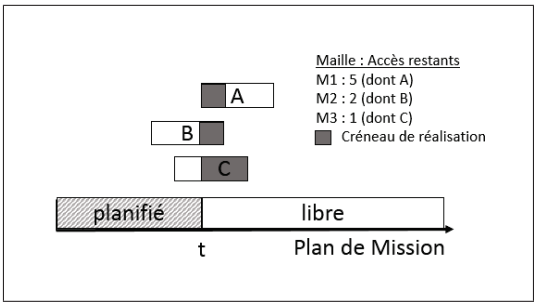


Figure 5. Heuristique de choix

La figure 5 permet de comprendre l'heuristique utilisée par ChronoG : les accès A, B et C (appartenant respectivement aux mailles M1, M2 et M3) sont possibles à t. C sera sélectionné car il appartient à la maille la plus critique : M3 n'a plus qu'un seul accès de disponible.

Comme mentionné dans la sous-section 2.2, il existe plusieurs types d'algorithme Glouton utilisés opérationnellement pour la planification de missions d'observation satellitaire en fonction de la manière dont sont parcourues les données d'entrée (les requêtes). Nous reprendrons ici les trois exemples décrits en sous-section 2.2: le Glouton Chronologique (celui décrit ci-dessus), le Glouton que nous avons appelé « Hiérarchique » et enfin le Glouton Stochastique Itéré.

Le Glouton Chronologique existe sous diverses variantes dont certaines utilisent des heuristiques pour gérer les groupes de conflits que sont les requêtes possédant des accès intersectés temporellement, mais le principe en est néanmoins globalement celui qui a été décrit ci-dessus.

Le Glouton Hiérarchique s'appuie finalement lui aussi sur une sorte d'heuristique constituée de la méthode utilisée pour classer les requêtes dans un ordre absolu. Ainsi, ensuite l'algorithme de planification n'a plus qu'à parcourir la liste dans l'ordre dé-

croissant des notes de hiérarchisation affectées à chaque requête pour chacun de ses accès. La différence majeure avec le Glouton Chronologique repose sur le fait que la version Chronologique n'a pas d'information sur le fait qu'un accès est meilleur qu'un autre pour une requête donnée, il avance de manière assez « myope » en ayant simplement l'information qu'il existe ou non d'autres accès utilisables plus tard pour chacune des requêtes en conflit qu'il est en train de traiter.

Dans la mesure où la note de hiérarchisation du Glouton Hiérarchique peut être utilisée par le Glouton Chronologique de la même manière que la priorité décrite ci-dessus est utilisée, lorsque ce dernier cherche l'acquisition à réaliser à la date courante de traitement, il sera confronté à des choix similaires à ceux auxquels est confronté l'algorithme Glouton Hiérarchique, donc à une combinatoire similaire. Pour être dans des conditions proches entre ces deux types d'algorithme, il faut considérer dans le Glouton Chronologique que les requêtes à prendre en compte à l'instant courant, sont toutes les mailles dont les accès ont des intersections non vides et dont au moins un accès contient la date courante.

Pour ce qui est du Glouton Stochastique Itéré, la combinatoire gérée est par nature moindre que pour les deux autres catégories d'algorithme Glouton du fait de l'horizon considéré qui est très sensiblement plus court. Cependant, ce dernier tente de compenser cette « courte vue » par de multiples tentatives censées améliorer suffisamment la solution obtenue pour ne pas laisser passer de solution acceptable. Comme déjà mentionné, ce désavantage est contrebalancé par le faible temps d'exécution (et donc la faible puissance de calcul) nécessaire à produire des solution pertinentes.

4.4.2. Taux de planification

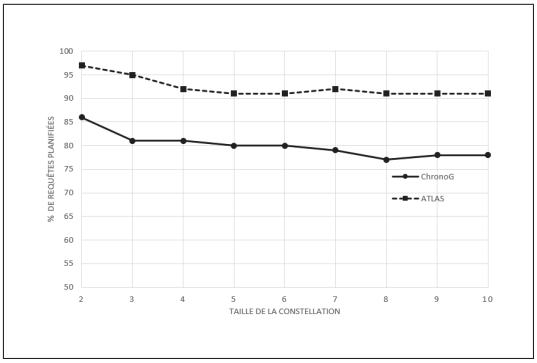


Figure 6. Comparaison des résultats entre ATLAS et ChronoG

La figure 6 montre que les résultats obtenus avec le système ATLAS sont meilleurs. Certes, les deux premières exécutions (sur des constellations de deux et trois satellites et une moyenne de deux accès par maille) produisent de bons résultats sur les deux systèmes, mais cela s'explique par le fait que chaque maille possède peu d'accès de visibilité.

Quand la taille de la constellation et le facteur d'accessibilité augmentent, les résultats se stabilisent, ATLAS assurant un taux de planification avoisinant les 90%, alors que ChronoG est en moyenne à 80%. Cette stabilisation est plus visible sur les résultats produits par ATLAS où la convergence se fait dès que la constellation atteint 4 satellites. Enfin, il est important de noter que l'écart type des pourcentages de planification pour ATLAS est de l'ordre de 2%, et ce quelle que soit la taille de la constellation. ATLAS produit donc des solutions homogènes.

Tableau 8. Comparaison taux de mailles planifiées et temps d'exécution

	% planification		Temps d'exécution	
S	ChronoG	ATLAS	ChronoG	ATLAS
2	86 %	97 %	20 ms	50 ms
3	81 %	95 %	31 ms	31 ms
4	81 %	92 %	32 ms	34 ms
5	80 %	91 %	32 ms	30 ms
6	80 %	91 %	47 ms	28 ms
7	79 %	92 %	50 ms	38 ms
8	77 %	91 %	64 ms	29 ms
9	78 %	91 %	50 ms	36 ms
10	78 %	91 %	50 ms	71 ms

La table 8 détaille les statistiques obtenues lors de l'exécution des neuf *scenarii* : taux de planification et temps moyen d'exécution. Ces résultats montrent qu'ATLAS planifie toujours plus de mailles que ChronoG. Concernant les temps moyens d'exécution, ATLAS est aussi meilleur. La variation de ces durées dépend de la complexité des *scenarii*. Un nombre de conflits important entraînent de nombreux appels à l'heuristique ce qui ralenti ChronoG. A l'inverse, les agents du système ATLAS traitent localement ces conflits, ce qui ne ralentit pas la résolution.

4.4.3. Appel à la fonction de guidage

L'opération la plus couteuse dans un système et la fonction qui calcule les mouvements du satellite. Cette fonction est nommée fonction de guidage. En effet, à partir de deux points (de l'orbite du satellite), elle doit calculer toutes les manœuvres à effectuer tout en assurant que le satellite ne soit pas mis en danger. Minimiser le nombre d'appels à cette fonction est donc un critère important. La figure 7 compare le nombre d'appels à cette fonction pour ATLAS et ChronoG, dans la planification des 9 *scenarii*. Ces derniers étant basé sur des données simulées, nous n'avons pas à proprement parler de fonction qui calcule le guidage dans nos systèmes. Cependant, nous savons à quel moment cette fonction serait appelée si les données étaient réelles. Dans le cas de ChronoG, cela correspond aux appels à la fonction heuristique. Il en va de même pour ATLAS, la fonction serait appelée quand les agents Satellite planifient l'agent Maille.

La figure 7 met en avant le fait qu'ATLAS fait beaucoup moins d'appel à la fonction de guidage que ChronoG (dans le pire cas ce dernier fait presque 4 fois d'appels). Cette différence peut facilement s'expliquer. En effet, ChronoG est un algorithme

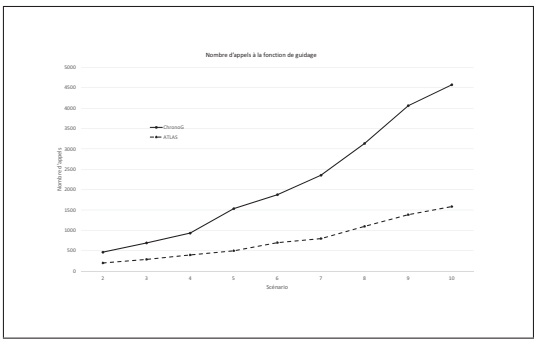


Figure 7. Comparaison du nombre d'appel à la fonction de guidage

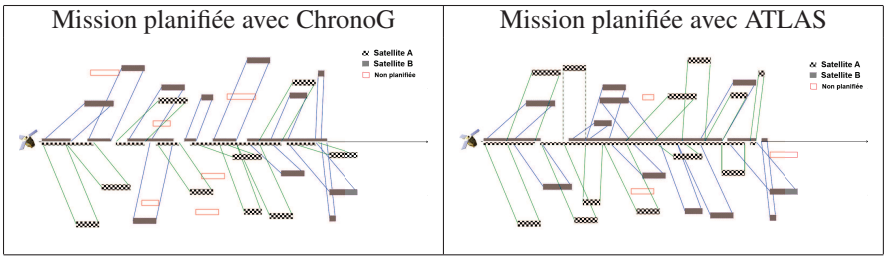
glouton : il va donc toujours essayer de placer des requêtes, sans *a priori*. Un grand nombre d'appels est fait sans placement. Au contraire ATLAS est plus intelligent sur ce point-là. Dans un premier temps les agents Satellite font une estimation : le coût. Et c'est en se basant sur ce coût que les agents Maille vont faire leur choix et donc demander confirmation (demande qui entraînera l'appel à la fonction de guidage). Cette « réflexion » que permet le coût entraîne donc un choix plus « intelligent » des agents. Bien entendue, la difficulté ici est d'avoir le coût le plus juste possible pour ne pas causer des re-planification (qui entraînent des appels à la fonction de guidage), mais aussi de pouvoir estimer le coût le plus rapidement possible, pour ne pas causer un ralentissement du système. Le coût est donc un compromis entre rapidité et justesse.

4.4.4. Équilibre de charge

Pour illustrer cette seconde expérience, nous avons généré un scénario particulier. Dans ce dernier, deux satellites *agiles* identiques (que nous nommerons A et B) se suivent, cette agilité leur permet d'effectuer des prises de vues parallèles à leur déplacement, mais aussi en avant ou en arrière. De plus, ils possèdent les mêmes accès de visibilité. Ainsi, chaque maille peut être acquise par n'importe quel satellite. Le but ici est de montrer la distribution de la charge au sein de la constellation. Les figures présentaient au tableau 9 sont des représentations des tâches que les satellites devront effectuer durant leur passage. L'axe central représente la trace au sol, c'est-à-dire le déplacement vu du sol des deux satellites, avec sur la partie haute (rectangle noir) les tâches du premier satellite et sur la partie basse (damier) celles du second. Les rectangles sont hachurés en fonction du satellite responsable de l'acquisition. Les rectangles vides correspondent quant à eux, aux mailles non affectées.

Il est intéressant de noter que les résultats obtenus sur ce cas d'étude sont proches de ceux que l'on avait pu obtenir via des *scenarii* aléatoires, ATLAS fournissant toujours de meilleurs taux de planification (88% pour ATLAS contre 76% pour ChronoG). Pour ChronoG (figure 1 du tableau 9), le satellite A à seulement neuf tâches de planifiées et B en compte onze. En comparaison, nous voyons sur la seconde figure du tableau 9 que la planification avec ATLAS donne douze prises de vues pour A et onze

Tableau 9. Comparaison de deux missions



pour B. Enfin, nous constatons aussi que la planification via la méthode gloutonne entraîne davantage de périodes d'inactivités. En effet, ChronoG n'étant pas coopératif mais reposant sur une heuristique, les requêtes attribuées bloquent des enchaînements qui auraient permis d'ordonnancer davantage de tâches.

4.5. Discussion

Les systèmes actuels sont basés sur une heuristique globale et chronologique. Avec l'évolution des constellations et l'augmentation du volume de demandes clients, ces approches classiques sont confrontés à de fortes limitations.

Tout d'abord, ajouter ou supprimer une requête lors de la construction du plan est difficile et coûteux. Cela est particulièrement le cas pour l'algorithme glouton. Si une requête impose une modification (en raison de sa priorité par exemple), le plan doit être complètement re-calculé depuis la date de la maille modifiée. C'est pour cette raison que les requêtes qui arrivent au cours du calcul du plan de mission sont traités plusieurs heures plus tard. Au contraire, ATLAS est un système ouvert et dynamique. Ajouter ou supprimer une requête lors de l'exécution est possible. Ces modifications sont des perturbations : les agents vont s'auto-adapter et modifier leur organisation pour converger vers une nouvelle solution locale.

Une autre limitation est l'approche *réductionniste* des méthodes utilisées pour planifier les constellations. Actuellement, les satellites sont planifiés individuellement, et ne tirent pas profit des avantages que présentent les constellations. Les agents composant ATLAS coopèrent pour planifier efficacement la constellation dans son ensemble. De plus, ces méthodes ne sont pas adaptées pour traiter un très grand nombre de requêtes et de contraintes. Dans le système ATLAS, cette complexité globale est ainsi distribué. Les interactions locales sont très simple et permettent ainsi de traiter au mieux ce problème complexe.

Enfin, la distributivité des interactions permettent de proposer des solutions temps réel. Si on arrête le système, une solution existe. Les approches gloutonnes construisant le plan de façon itérative (chronologique ou par priorité), tant que l'ensemble dès requêtes n'a pas été traité, le plan reste valide mais très incomplet.

5. Conclusion et perspectives

Dans cet article, nous avons présenté une approche basée sur les systèmes multi-agents adaptatifs pour planifier efficacement une constellation de satellites d'observation de la Terre : ATLAS. Dans ce système, la coopération des agents est guidée par la criticité des agents mailles, et par le coût indiqué par les agents satellites. Ces deux moteurs de la coopération permettent ainsi d'assurer qu'un nombre de requêtes important soit planifié de façon équilibrée entre les satellites de la constellation. De plus, les requêtes peuvent être prise en compte de façon dynamique durant la planification. Enfin, ATLAS peut être stoppé à tout moment et fournir une solution. Les différentes expérimentations ont montré qu'ATLAS fournissait des résultats de meilleure qualité que l'algorithme Glouton Chronologique couramment utilisé (ChronoG). De plus, ATLAS fournit des solutions ayant un meilleur équilibrage de la charge de la constellation. Ces premiers résultats sont donc très encourageants et prouvent l'intérêt d'une approche par système multi-agent pour planifier efficacement une constellation de satellites.

Nos futurs travaux vont concerner l'ajout d'un nouveau niveau d'adaptation, afin d'atteindre de meilleures solutions. Le comportement manquant des agents Requêtes sera aussi implémenté, et la mesure de criticité des agents Maille tiendra compte de cette influence. ATLAS sera aussi testé sur des cas d'utilisations réels. Ainsi, nous pourrons comparer nos résultats avec des plans de mission produits par le CNES *(Centre National d'Études Spatiales)* ou Airbus D&S-Geo (anciennement Spot Image), deux organismes qui gèrent les constellations des satellites Spot et Pléiades.

Remerciements

Les auteurs souhaitent remercier l'IRT Saint Exupéry pour le financement de cette recherche.

Bibliographie

Bensana E., Verfaillie G. (1999). Earth Observation Satellite Management. In *Constraints*, vol. 299, p. 293–299.

Bensana E., Verfaillie G., Agnese J., Bataille N., Blumstein D. (1996). Exact &inexact methods for daily management of earth observation satellite. In *Space mission operations and ground data systems-spaceops' 96*, vol. 394, p. 507.

Bianchessi N., Cordeau J. F., Desrosiers J., Laporte G., Raymond V. (2007, mars). A heuristic for multi-satellite, multi-orbit and multi-user management of Earth observation satellites. *European Journal of Operational Research*, vol. 177, n° 2, p. 750–762.

Bonnet G. (2008). *Coopération au sein d'une constellation de satellites.* Thèse de Doctorat, Université de Toulouse.

Bonnet G., Tessier C. (2009). Évaluation d'un système multirobot cas d'une constellation de satellites. *Revue d'Intelligence Artificielle*, vol. 23, p. 565–593.

Bouveret S., Lemaître M. (2006). Un algorithme de programmation par contraintes pour la recherche d'allocations leximin-optimales. In *Deuxièmes journées francophones de programmation par contraintes (jfpc06)*.

Bouziat T., Combettes S., Camps V., Glize P. (2014). La criticité comme moteur de la coopération dans les systèmes multi-agents adaptatifs (short paper). In *Journées Francophones sur les Systèmes Multi-Agents, 2014*, p. 149–158. Citeseer.

Cordeau J.-F., Laporte G. (2005). Maximizing the value of an earth observation satellite orbit. *Journal of the Operational Research Society*, vol. 56, n° 8, p. 962–968.

Dago P. (1997). *Extension d'algorithes dans le cadre des problèmes de satisfaction de contraintes valués*. Thèse de Doctorat.

Frank J., Ari J., Morris R., Smith D. E., Field M. (2001). Planning and Scheduling for Fleets of Earth Observing Satellites.

Gleizes M.-P. (2012). Self-adaptive Complex Systems (regular paper). In *European Workshop on Multi-Agent Systems, Maastricht, The Netherlands*, vol. 7541, p. 114–128.

Globus A., Crawford J., Lohn J., Pryor A. (2003). Scheduling earth observing satellites with evolutionary algorithms. In *Conference on space mission challenges for information technology*.

Globus A., Crawford J., Lohn J., Pryor A. (2004). A comparison of techniques for scheduling earth observing satellites. In *Aaai*, p. 836–843.

Grasset-Bourdel R. (2011). *Planification dynamique et réactive pour des satellites agiles d'observation de la Terre*. Thèse de Doctorat, Onera.

Grasset-Bourdel R., Flipo A., Verfaillie G. (2011). Planning and replanning for a constellation of agile Earth observation satellites.

Kaddoum E. (2011). *Optimization under Constraints of Distributed Complex Problems using Cooperative Self-Organization*. Thèse de Doctorat, Université de Toulouse.

Lemaître M., Verfaillie G., Jouhaud F., Lachiver J. M., Bataille N. (2002). Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology*, vol. 6, n° 5, p. 367–381.

Maillard A. (2015). *Flexible scheduling for agile earth-observing satellites*. Thèse de Doctorat, ISAE Supaero.

Mancel C. (2004). *Modélisation et résolution de problèmes d'optimisation combinatoire issus d'application spatiales*. Thèse de Doctorat, Université de Toulouse.

Mansour M. A., Dessouky M. M. (2010). A genetic algorithm approach for solving the daily photograph selection problem of the spot5 satellite. *Computers & Industrial Engineering*, vol. 58, n° 3, p. 509–520.

Vasquez M., Hao J.-K. (2003). Upper bounds for the spot 5 daily photograph scheduling problem. *Journal of Combinatorial Optimization*, vol. 7, n° 1, p. 87–103.

Wang P., Reinelt G., Gao P., Tan Y. (2011). A model, a heuristic and a decision support system to solve the scheduling problem of an earth observing satellite constellation. *Computers & Industrial Engineering*, vol. 61, n° 2, p. 322–335.

Wolfe W., Sorensen S. (2000). Three Scheduling Algorithms. *Inform Journal on Management Science*, vol. 46.

Wu G., Wang H., Li H., Pedrycz W., Qiu D., Ma M. *et al*. (2014, janvier). An adaptive Simulated Annealing-based satellite observation scheduling method combined with a dynamic task clustering strategy. *Computing Research Repository*, vol. abs/1401.6098, p. 23.