ATLAS : planification multi-satellite dynamique en 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 de Toulouse, Toulouse, France*
   Prenom.Nom@irit.fr

RÉSUMÉ. *La planification de missions de constellations de satellites est un problème complexe soulevant d'importants défis technologiques pour les systèmes de demain. Le grand nombre de demandes clients et leurs arrivées dynamiques implique une combinatoire et une dynamique très élevées. Les techniques actuellement utilisées 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 utilisé 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.*

DOI:10.3166/RIA.30.35-59

1. Introduction

Une constellation de satellites d'observation de la Terre est un ensemble de satellites potentiellement hétérogènes. 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. La planification des missions d'observation pour une telle constellation est un problème complexe. 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*).

Actuellement, la planification d'une constellation de satellites suit un schéma en trois temps :

1. Réception des requêtes des clients et leurs contraintes ainsi que leur découpage en zones géographiques de petites tailles (appelées mailles cf. section 2.1) pouvant être acquises par les satellites.
2. Élaboration du plan de prise de vues que les satellites doivent réaliser afin de satisfaire un maximum de requêtes.
3. Chargement à bord de chaque satellite de la constellation du plan de mission le concernant lorsque ce dernier est en visibilité du sol.

Ces temps de visibilité n'étant pas fréquents et étant de courte durée, il est important que le plan de mission soit élaboré et validé avant que le satellite soit visible par le sol. Les algorithmes de planification actuellement utilisés montrent des limites quant à la gestion des changements dynamiques (prise en compte de nouvelles requêtes urgentes ou de changements climatiques) en temps réel. Ainsi, l'élaboration du plan de mission est retardé au maximum afin d'avoir les informations les plus à jour et de minimiser le nombre de requêtes clients qui seront stockées pour la prochaine période de planification.

Dans un contexte où le domaine de l'observation est en pleine expansion ; augmentation de la taille des constellations (par exemple le projet *Skybox* de Google avec 24 satellites), accroissement des requêtes des clients demandeurs d'un *feedback* plus rapide quant à la gestion de leur demande, *etc.* Il est indispensable de rendre cette planification plus réactive et dynamique. Pour cela, nous proposons l'utilisation de méthodes basées sur l'auto-adapation et l'auto-organisation, et notamment les systèmes multi-agents coopératifs (Gleizes, 2012). Ces méthodes apportent plus de flexibilité, de robustesse et d'adaptivité temps réel. Ainsi, les plans de mission peuvent être calculés de manière dynamique, tout en traitant d'importants volumes de données. Le système de planification est ainsi capable de fournir plus rapidement des informations sur l'état de la planification aux utilisateurs.

Dans cet article, nous nous intéressons au problème de la répartition dynamique des acquisitions dans une constellation de satellites. Cette répartition consiste à planifier les requêtes clients découpées en mailles (cf. section 2.1) sur les différents satellites, tout en respectant les contraintes et en assurant une charge équilibrée pour chaque satellite. Nous supposons que l'algorithme qui permet à un satellite de construire son plan de mission à partir d'un ensemble de mailles lui étant affectées est donné et donc nous ne le décrirons pas.

La suite de l'article est organisée comme suit : la section 2 décrit le problème et présente son contexte. La section 3 développe le fonctionnement du système multi-agent développé : ATLAS (*Adaptive saTellites pLanning for dynAmic earth obServation*). Enfin, la section 4 présente l'évaluation du système ATLAS ainsi que sa comparaison à un algorithme glouton chronologique qui est proche des techniques utilisées actuellement en contexte opérationnel.

2. Planification multi-satellite

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

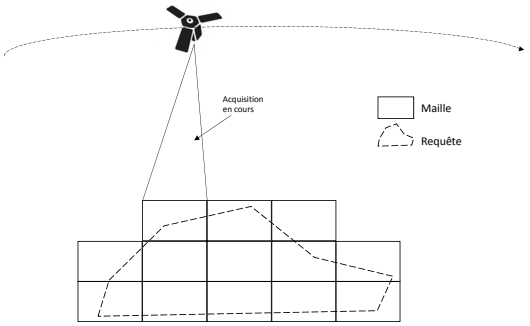


Figure 1. Un satellite en train d'acquérir une maille

2.1. Description du problème

En nous appuyant sur les travaux de (Bensana, Verfaillie, 1999) et (Bonnet, 2008), nous allons tout d'abord décrire notre problème. Dans la section 3, nous utiliserons 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. Requête

Une requête r_i est une commande client définie par un ensemble de données (nous noterons l'ensemble des requêtes à planifier R) :

– 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. Une requête pouvant représenter une grande zone (un pays, un continent, *etc.*), un satellite ne peut pas l'acquérir en une seule prise. Chaque requête est donc découpée (en amont) en entités élémentaires que le satellite peut acquérir en une prise de vue : les mailles. La requête r_i est acquise si et seulement si toutes les mailles de M^i sont acquises. La figure 1 montre une requête découpée en plusieurs mailles, et un satellite en train d'acquérir l'une d'elle.

2.1.3. Contraintes dures et souples

Deux catégories de contraintes sont à prendre en compte : les contraintes dures, comme la plage de validité, les possibilités d'enchaînement des manœuvres ou le type d'image souhaitée par le client ; et les contraintes souples (comme le taux de couverture nuageuse qui impacte la qualité de l'acquisition mais qui peut être dégradé). L'ensemble C_h de contraintes dures doit être satisfait, les contraintes souples, C_s peuvent être relâchées.

2.1.4. But du problème de planification

L'objectif du problème est donc de trouver pour chaque satellite un ensemble de mailles à acquérir, et d'attribuer une date de début pour chaque acquisition. 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 et de requêtes à satisfaire.

2.2. Méthodes de planification de missions

De par sa dynamique, du grand nombre d'entités et de contraintes qui sont en interactions, la planification de missions d'observation est un problème d'optimisation complexe possédant une forte combinatoire. (Lemaître *et al*., 2002) le catégorisent comme **NP-Difficile**. Dans la littérature, il existe un certain nombre de propositions, qui se basent sur des méthodes d'optimisation classique, pour tenter de le résoudre. Nous regroupons ici les approches les plus couramment utilisées dans le domaine, et les trois critères suivants sont utilisés pour les comparer :

1. le passage à l'échelle : capacité à traiter des problèmes de grandes tailles,
2. la dynamique : possibilité d'ajouter de nouvelles requêtes,
3. la capacité de donner une réponse en temps réel ou au cours de l'exécution.

Tel que décrit à la section 2.1, le problème peut être formalisé comme un problème de satisfaction de contraintes (**CSP**). Néanmoins et étant donné sa complexité, il est difficile de le considérer dans sa totalité et de le modéliser correctement de manière mathématique. Il est donc nécessaire de le simplifier (Lemaître *et al*., 2002). Cette simplification peut être réalisée en relaxant certaines contraintes, en raccourcissant l'horizon de planification ou encore en limitant les manœuvres des satellites. Ce problème simplifié est correct, même s'il mène à des solutions sous-optimales.

Appréhender comme un problème de satisfaction de contraintes, il est donc possible de le résoudre via des méthodes de résolution exactes. Dans (Bensana *et al*., 1996), (Dago, 1997) et (Bouveret, Lemaître, 2006) des outils performants tel qu'**ILOG Solver** ont été utilisés. (Grasset-Bourdel *et al*., 2011) ont étudié l'utilisation de la **Programmation Dynamique** pour décomposer le problème en sous-problèmes plus simples. Les problèmes liés à ces algorithmes viennent de la difficulté à découper récursivement le problème initial. Ainsi, si l'algorithme est appliqué linéairement, son déroulement utilise rapidement la majorité des ressources mémoires du planificateur et le rendre inapplicable en cas de requêtes demandant la gestion d'un profil mémoire telle que la stéréoscopie (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).

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. Ces méthodes ne sont donc pas adaptées pour notre problème.

Les méthodes approchées peuvent être appliquées pour pallier ces différents inconvénients. En s'appuyant sur une fonction heuristique, elles permettent de trouver une solution de 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**. Diverses implémentations existent, en fonction de la manière dont sont parcourues les requêtes à planifier. 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 requêtes. Les versions les plus utilisées de cette famille d'algorithme sont :

– Le **glouton chronologique** (Bianchessi *et al*., 2007) : la planification se fait chronologiquement : à l'instant du plan où l'algorithme se situe, un choix est fait sur l'acquisition la plus pertinente (généralement la plus prioritaire). Ainsi cette acquisition est placée temporellement. L'algorithme planifie par la suite une nouvelle requête à la suite du plan courant.

– Le **glouton hiérarchique** (Wang *et al*., 2011) : les requêtes éligibles sont classées sur la période temporelle à planifier en fonction d'un certain nombre de critères d'évaluation. Ensuite, la liste est dépilée et chaque requête est placée dans le plan à la meilleure position possible pour elle.

– Le **glouton stochastique itéré** (Frank *et al*., 2001) : cette version consiste, partant d'un état donné du plan qui est dit « de référence » (qui peut être éventuellement vide), à prendre aléatoirement les requêtes et à essayer de les placer. Le nouveau plan est par la suite évalué.

Les méthodes de **recherches locales** (Tangpattanakul *et al*., 2015) telles que le **recuit simulé** ((Wu *et al*., 2014), (Globus *et al*., 2004)) ou les **recherches tabou** ((Bianchessi *et al*., 2007), (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. Pour produire la meilleure solution possible, beaucoup de paramètres doivent être adaptés. Ainsi, l'algorithme doit être retravaillé pour toute modification du problème. Enfin, les **algorithmes génétiques** tels que ceux présentés par (Yuan *et al*., 2014), (Mansour, Dessouky, 2010) ou encore (Globus *et al*., 2003) sont aussi étudiés, mais la modélisation est difficile et ces algorithmes sont assez lents (plusieurs heures), ce qui est dommageable pour planifier des missions rapidement en temps réel.

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 : la dynamique est ainsi mal gérée. 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 relatives à plusieurs satellites. Le tableau 1 note chaque méthode présentée ici par rapport aux critères de comparaison 1, 2 et 3. Les notes sont à interpréter entre - - (le pire cas) et + + (le meilleur cas). Au vu des limites des approches présentées et de l'évolution du domaine, il est donc nécessaire d'étudier et de concevoir de nouvelles méthodes de planification amenant plus de flexibilité et une meilleure gestion de la dynamique.

Tableau 1. Comparaison des méthodes

Méthode	Passage à l'échelle	Dynamique	Rp temps réel
Résolution CSP	-	-	-
Progr. Dynamique	- -	-	-
Séparation et Évaluation	-	-	-
Algorithme Glouton	- -	-	+
Recherches Locale	-	+	+
Algorithme Génétique	-	-	+

Parmi ces derniers, de nombreuses autres méthodes permettent de résoudre des problèmes d'optimisation dynamique. Par exemple, nous pouvons citer des méthodes classiques de résolution de **DCOP** (*Distributed Constraint Optimization Problem*) telle que **ADOPT** (*Asynchronous Distributed Constraint Optimization*) (Modi *et al.*, 2005), des algorithmes à **essaims particulaires** (Jordehi, Jasni, 2015) ou à base de **colonies de fourmis** (Mavrovouniotis *et al.*, 2015). Cependant, aucune n'ayant été utilisée dans le domaine de la planification satellite, nous avons fait le choix d'étudier les systèmes multi-agents et de les comparer à des algorithmes utilisés dans les systèmes opérationnels.

Les systèmes multi-agents ont montré leurs avantages en termes de flexibilité et de gestion de la dynamique, et ce dans divers domaines d'applications ((O Ramos *et al.*, 2013), (Boes *et al.*, 2015)). Cependant, ces systèmes n'ont jamais été utilisés pour la planification au **sol** de missions de satellites. De tels systèmes ont été étudiés afin 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 pour maximiser le nombre de demandes satisfaites. A cause de 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.*) limitant son déploiement.

2.3. La théorie des 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. Cette approche permet de prendre en compte naturellement un grand nombre d'entités et de contraintes et fournit des mécanismes d'auto-organisation qui permettent de s'adapter à la dynamique du problème. Dans la théorie **AMAS** (pour *Adaptive Multi-Agent Systems*) (Gleizes, 2012), les agents sont définis comme des entités autonomes, adaptatives et coopératives qui possèdent leur propre objectif local à atteindre en ayant une connaissance partielle de leur environnement. 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 qualifiée d'*adéquate*. La coopération est définie comme l'attitude 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 l'organisation du système.

Les problèmes complexes d'optimisation sous contraintes sont formalisés comme un ensemble d'entités régi 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.

La méthodologie **ADELFE** (*Atelier de Développement de Logiciels à Fonctionnalité Émergente*) (Bonjean *et al*., 2014) aide à la conception de logiciels à fonctionnalité émergente, tels que les AMAS. Cette méthodologie est basée sur des outils de l'ingénierie des systèmes tels que UML (*Unified Modelling Language*) et RUP (*Rational Unified Process*). Cinq phases la composent, allant des besoins préliminaires aux développement. De plus, chaque phase est décomposée en un ensemble d'activités. Ces activités aident le développeur à concevoir son système mais reste à un certain niveau d'abstraction, ADELFE étant très générique. Cette généricité est une force, la méthode pouvant être appliquée à un grand nombre d'applications, mais aussi une limite car elle requiert un effort considérable afin d'être adaptée pour un domaine en particulier. Ainsi, 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 requièrent 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 avec l'aide des agents sous le rôle « service », la solution peut être atteinte. Dans un problème d'optimisation, différents agents sous le rôle « contraint » peuvent demander le service du même agent « service ». En tant qu'agent coopératif, ce dernier aide 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

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 section 2.1, et en se basant sur la méthode ADELFE 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 est détaillé ci-après),

– **trois types d'entités actives** : la couverture, l'éphéméride solaire et les stations de télé-déchargement. Ces entités n'ont pas de but à satisfaire mais modifient de manière autonome leur état, ce dernier pouvant influencer la décision des agents coopératifs.

– **trois types d'entités passives** (ressources du système, qui influencent la résolution mais n'ont pas d'autonomie) : la mémoire des satellites, leur batterie et un module de calcul de trajectoire et d'attitudes orbitales. Par exemple, le satellite impacte sur la mémoire s'il réalise une acquisition ou non.

L'utilisation du modèle AMAS4Opt permet de définir le comportement et les interactions des agents : les agents Requête et Maille portent les contraintes du problème et doivent être satisfaits, ils ont le rôle « contraint » et les agents Satellite prennent le rôle « service » car ils aident à la résolution. 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 détaille les agents.

Tableau 2. Les agents coopératifs

Agent	Satellite	Maille
Rôle	« Service »	« Contraint »
Objectif	Acquérir des prises de vues	Être planifié
Négociations	Agents Maille	Agents Satellite
Interactions	Module de gestion trajectoire Couverture nuageuse Batterie et mémoire Éphémérides	Éphémérides

Dans le système ATLAS, la décomposition des agents Requête en agents Maille n'est pas prise en compte, par manque de données réelles. **Ainsi les interactions entre ces deux types d'agents ne sont pas implémentées**. Cette absence n'entrave pas la résolution du problème car les agents Requête n'influencent qu'une partie de la criticité des agents Maille et pas leur comportement.

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 est guidée par deux indicateurs : **la criticité et le coût**. Cette coopération permet de produire une planification respectant les critères suivants : un maximum de requêtes planifiées et un équilibre au sein de la constellation (les satellites doivent tendre vers la même charge de travail). Le fonctionnement du système ATLAS est basé sur la répétition par les agents de cycles « Perception - Décision - Action ». La phase de décision est l'étape centrale. En fonction de ses perceptions partielles et de son état, l'agent décide quelle action réaliser. Ces interactions sont présentées dans le tableau 3.

Le comportement coopératif des agents est décrit via les algorithmes 1 et 2. L'agent Maille cherche un satellite pouvant le satisfaire, mais comme il ne sait pas lequel choisir, le choix est fait sur la base d'une estimation de la difficulté que cette planification représente pour chaque satellite, information qui est donnée par l'agent Satellite à l'agent Maille. Les agents Satellite traite les demandes de service et en cas de conflit, l'agent Maille le plus critique est privilégié.

Comme précédemment mentionné, les requêtes clients n'arrivent pas toutes en même temps. ATLAS est capable de gérer cette dynamique, contrairement aux approches actuelles. En effet, les agents Satellite privilégient les agents Maille les plus « critiques ». Ainsi, si un agent Maille de faible priorité était planifié par un agent Satellite et qu'un nouvel agent Maille très urgent arrive, l'agent Satellite déplanifie le moins critique. Celui-ci cherche une nouvelle place (ligne 1 de l'algorithme 1). Enfin, les interactions entre les 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 modifications (ajout de requêtes par exemple) entraînent des perturbations locales qui ne remettent pas en question l'ensemble du plan.

Le diagramme de séquences présenté la figure 2 décrit un exemple d'interactions entre un agent Maille et deux agents Satellite qui peuvent le planifier. Le diagramme peut être décrit comme suit :

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

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 exists 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     goto line 1;
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             select the more critical mesh;
12             plan it;
13             inform cancel the other;
14         end
15     end
16 end
```

– à la réception de ce message, les agents Satellite estiment 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 choisit **coopérativement** le Satellite ayant le coût le plus faible et lui envoie un message 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 sélectionne un autre agent Satellite *askForConfirmation()* (action 6).

– Une confirmation est émise (action 7).

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

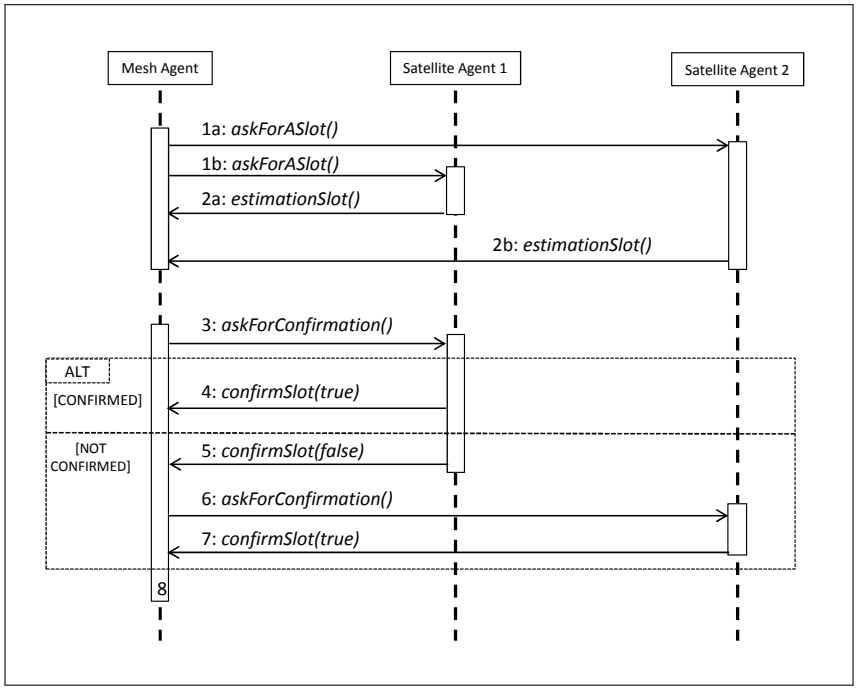


Figure 2. Diagramme de séquences

3.3. La criticité et le coût

3.3.1. La criticité

Dans ATLAS, le degré de non satisfaction de l'agent Maille est représenté par sa 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 Satellite sont donc écrites pour favoriser les agents Maille les plus *critiques*. Cette criticité, qui indique 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 : $Cm = \langle P, Ra \rangle$.

Le premier critère P (pour *priority*) concerne la priorité de planification de la maille. Celle-ci est définie par le client au moment de sa commande. En cas d'égalité de ce critère, les agents Satellite utilisent comme second critère le nombre d'accès de visibilité restants : Ra (pour *remaining access*). Ainsi, un agent Maille qui n'est visible qu'une seule fois par les satellites sera prioritaire par rapport aux autres agents Maille 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 donc 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 définie dans le modèle AMAS4Opt permet une coopération *partielle* entre les agents : elle est seulement assurée par les agents ayant le rôle « contraint ». En effet, seuls les agents ayant le rôle « service » connaissent quels agents « 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 pallier ce manque, et rendre les agents ayant le rôle « contraint » plus coopératifs.

Ainsi 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 reçoit un coût pour chaque demande émise aux agents Satellite. La coopération est assurée car un agent Maille privilégie l'agent Satellite lui répondant avec le coût le plus faible. Le coût est représenté par le triplet $C = \langle Cm, Ml, D \rangle$.

– Cm représente la nécessité d'adapter le plan, en déplaçant des éléments planifiés,

– Ml représente la charge mémoire,

– D représente le nombre de messages reçus de demandes de planification.

4. Résultats et discussions

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 missions, 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 au moins une solution correspondant au plan complet. En plus de l'accès de visibilité donné par le plan complet, le générateur ajoute à chaque maille un nombre aléatoire d'accès de visibilité vers d'autres satellites et/ou à d'autres créneaux temporels. Ce nombre est pondéré par un facteur d'accessibilité sur lequel l'utilisateur peut influer, et ainsi produire différents *scenarii*, chacun contenant une liste de mailles ayant toutes N accès de visibilité. La solution complète est donc difficile à obtenir parmi le grand nombre de solutions possibles. La figure 3 illustre les étapes du générateur : la création d'un plan complet, l'ajout de créneaux temporels sur le même satellite et vers d'autre satellites, puis la production de la liste qui sera donnée aux systèmes de planification.

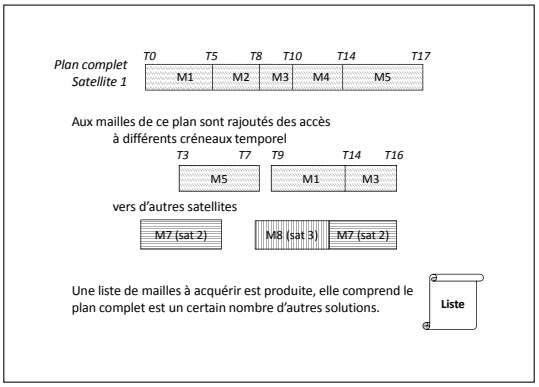


Figure 3. Construction d'une liste de requêtes par le générateur

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 présentées, nous avons utilisé neuf *scenarii* présentés dans le tableau 4. Dans chaque scénario, un satellite supplémentaire est rajouté. De plus, le facteur d'accessibilité des requêtes par les satellites est augmenté. Ainsi, plus la constellation est grande, plus le nombre de possibilités d'accessibilité pour chaque maille est grand. Cela permet de complexifier la résolution : en augmen-

tant le nombre d'accès possibles, le nombre de solutions possibles augmente. Nous définissons un indicateur sur la performance de 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 les systèmes de planification : ATLAS et ChronoG, l'algorithme glouton chronologique.

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 pas 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. Enfin, les demandes clients n'ont pas la même priorité, cette dernière pouvant être « Routine » (*la plus faible*), « Normale » ou « Urgente » (*la plus élevée*). Nous considérons deux critères pour comparer les systèmes : le pourcentage de chaque type de priorité d'agents Maille planifiés et l'évolution de la criticité globale du système durant la résolution, utilisée uniquement pour l'évaluation et pas dans le comportement des agents. Enfin, nous analyserons l'évolution du nombre de messages échangés ainsi que le nombre de cycles de décision en fonction de la taille des problèmes à résoudre. Un cycle d'exécution correspond au moment où tous les agents ont terminés leur cycle de « Perception - Décision - Action » (présenté dans la section 3.2).

4.3. Apport d'un comportement dynamique

Dans cette première comparaison, ATLAS 1 et 2 sont testés sur le scénario 4 du tableau 5. Ce scénario a été généré via le générateur présenté dans la section 4.1, une solution *complète* existe mais elle est difficile à obtenir. De plus, afin de souligner l'apport des mécanismes d'auto-adaptation, l'arrivée de nouvelles mailles dans les deux

Tableau 5. Pourcentage de requêtes acquises par ATLAS 1 et 2

	ATLAS 1	ATLAS 2
Stabilité	17	24
% Routine	68	91
% Normale	89	95
% Urgente	88	96
% Total	86	94

systèmes est contrôlée. En effet, toutes les mailles de priorité « normale » et « routine », soit 70 % de l'ensemble des mailles, sont disponibles au début de l'exécution. Les « urgentes » (les 30 % restants) sont insérées au cycle numéro 10. Le tableau 5 présente les résultats. ATLAS 2 planifie plus de mailles « urgentes » (96 %) que ATLAS 1 (seulement 88 %) et le pourcentage final de mailles planifiées est aussi plus important : 94 % contre 86 %. Quand ATLAS ne peut plus rien planifier, la stabilité est atteinte. Pour ATLAS 1 elle est atteinte au cycle 17 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

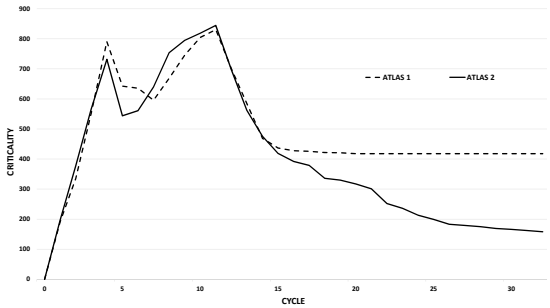


Figure 4. Évolution de la criticité globale

Nous nous intéressons ici à l'évolution de la criticité globale pour l'exécution des deux systèmes. Dans les systèmes opérationnels les demandes clients peuvent arriver à tout moment, nous introduisons à des moments aléatoires l'ensemble des mailles de trois priorités clients différentes. Cette criticité globale n'est qu'un indicateur pour observer le comportement du système : elle n'est pas utilisée durant la résolution. Elle est

calculée en additionnant les priorités client des agents Maille non-planifiés à chaque cycle d'exécution. Nous attribuons une valeur de 3 pour une priorité « urgente », 2 pour une priorité « normale » et 1 pour une priorité « routine ». Mathématiquement, cette criticité globale s'écrit : $GC_i = \sum_{j \in M^i} P_j$, où i est le cycle, M^i l'ensemble des agents Maille non planifiées au cycle i et P_j est la priorité du client de l'agent de Maille j.

La figure 4 compare l'évolution de la criticité globale. 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 élevée parce que le système déplanifie des agents Maille et donc libère de l'espace pour des agents Maille 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 avec ses comportement d'auto-adaptation est plus efficace : les agents Satellite adaptent leur plan et ainsi planifient un maximum d'agents Maille permettant ainsi de faire diminuer la criticité globale du système.

4.3.2. Communications et cycles de résolution

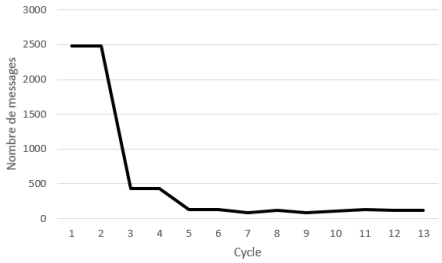


Figure 5. Évolution du nombre de messages par cycle

La figure 5 permet de voir l'évolution du nombre de messages au cours des cycles de décision, et ce pour la résolution du scénario 5 avec ATLAS 2. Deux fortes décroissances sont visibles entre les cycles 2 - 3 et 4 - 5. Les agents Maille initialisent le système, puis, ayant tous émis des requêtes vers les agents Satellite au cycle 1, ce sont ces derniers qui traitent les demandes lors du cycle 2, les agents Maille étant en attente de réponses. Cinq fois moins de messages sont envoyés au cycle 3, en effet, à ce stade ce sont les agents Maille qui traitent les réponses à leurs requêtes et demandent confirmation au satellite choisi. Le même phénomène, un palier puis une forte diminution aux cycles 3, 4 et 5, est visible mais avec moins de messages car davantage d'agents Maille sont déjà dans un état « planifiés », ils ne communiquent donc plus. En convergeant rapidement vers une solution, les agents n'encombrent pas le système

par un excès de communication. Le tableau 6 permet de visualiser le nombre de messages émis par les agents Maille et le nombre de cycles d'exécution nécessaires pour des constellations de plus en plus grandes. On constate que même si le nombre de messages augmente, le système converge toujours vers une solution de bonne qualité en un faible nombre de cycles et donc en un temps de calcul restreint.

Tableau 6. Nombre de messages échangés et nombre de cycles de résolutions

Satellites	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

Nous comparons la version d'ATLAS avec tous les mécanismes d'adaptation (nommée ATLAS 2) à la solution standard actuellement utilisée par les références du spatial : l'algorithme glouton chronologique, nommé ici **ChronoG**. ChronoG traite la constellation en considérant un satellite à la fois.

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.

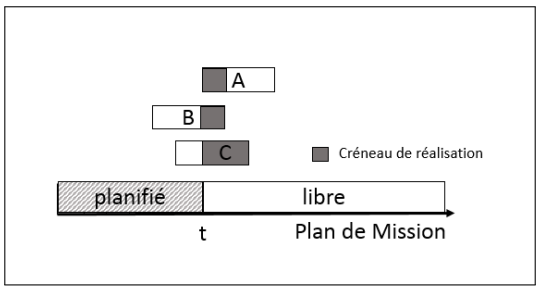
– Si une maille est planifiée, ChronoG passe à $t+1$, sinon, ChronoG vérifie si une maille peut être placée à cet instant en déterminant si la maille en question possède un accès de visibilité qui englobe la date courante t.

– Si plusieurs mailles sont possibles, ChronoG utilise l'heuristique suivante 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.

La figure 6 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.



Mailles	Accès Restant(s)
M1	5 (dont A)
M2	2 (dont B)
M3	1 (dont C)

Figure 6. Heuristique de choix et exemple d'application sur trois accès

Comme mentionné dans la partie 2.2, il existe plusieurs types d'algorithmes Glouton utilisés opérationnellement pour la planification de missions d'observation se différenciant par la manière dont sont parcourues les requêtes des clients. Nous reprendrons ici les trois exemples décrits dans la partie 2.2 : le Glouton Chronologique (celui décrit ci-dessus), le Glouton « 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 ayant une intersection temporelle, mais le principe est néanmoins globalement celui qui a été décrit ci-dessus.

Le glouton hiérarchique s'appuie lui aussi sur une heuristique constituée de la méthode utilisée pour classer les requêtes dans un ordre absolu. Ensuite, l'algorithme de planification parcourt la liste dans l'ordre décroissant 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.

Cette note de hiérarchisation peut être utilisée par le Glouton Chronologique. Lorsque ce dernier cherche l'acquisition à réaliser à la date courante de traitement, il est confronté à des choix similaires effectués par 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, la combinatoire gérée est moins importante car la période à planifier est plus courte. Cependant, ce dernier tente de compenser cette *courte vue* par de multiples tentatives censées améliorer suffisamment la solution obtenue. Comme déjà mentionné, ce désavantage est contrebalancé par le

faible temps d'exécution (et donc la faible puissance de calcul) nécessaire pour produire des solutions pertinentes.

Par conséquent, le glouton chronologique développé et utilisé pour nos comparaisons est assez réaliste et comparable aux algorithmes utilisés dans les systèmes de planification actuels.

4.4.2. Taux de planification

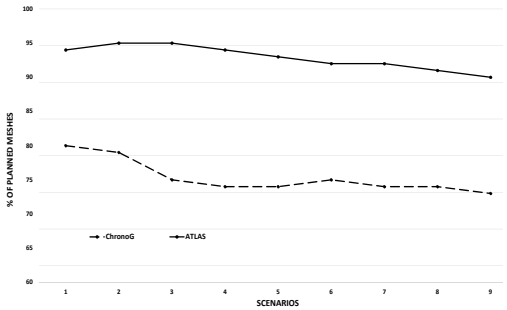


Figure 7. Comparaison des résultats entre ATLAS et ChronoG

La figure 7 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 7. 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	61 ms

Le tableau 7 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. Les temps moyens d'exécution sont assez proches entre les deux systèmes. ChronoG est plus rapide sur les *scenarii* simples car il ne planifie que très peu de mailles. Néanmoins, nous pouvons noter que la complexité d'ATLAS n'entraîne pas une augmentation drastique du temps de résolution. ChronoG est plus lent sur des *scenarii* complexes. En effet, un nombre important de conflits entraînent de nombreux appels à l'heuristique ce qui ralentit ChronoG. A l'inverse, parce que les agents du système ATLAS traitent localement ces conflits, cela ne ralentit pas la résolution.

4.4.3. Appel à la fonction de guidage

L'opération la plus coûteuse dans un système de planification 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 8 compare le nombre d'appels à cette fonction pour ATLAS (avec tous les mécanismes d'adaptation) et ChronoG, dans la planification des 9 *scenarii*. Ces derniers étant basés 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.

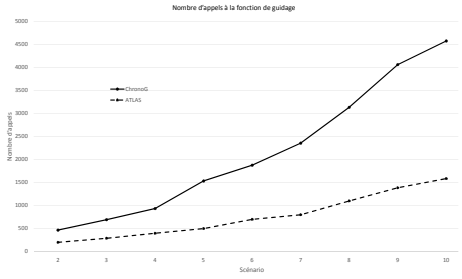


Figure 8. Comparaison du nombre d'appel à la fonction de guidage

La figure 8 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 plus fois d'appels). Cette différence s'explique car ChronoG est un algorithme glouton : il essaye toujours 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 font leur choix et demande confirmation (demande qui entraîne l'appel à la fonction de guidage). Cette « réflexion » que permet le coût entraîne donc un choix plus « intelligent » des agents. Bien entendu, la difficulté ici

est d'avoir le coût le plus juste possible pour ne pas causer des re-planifications (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

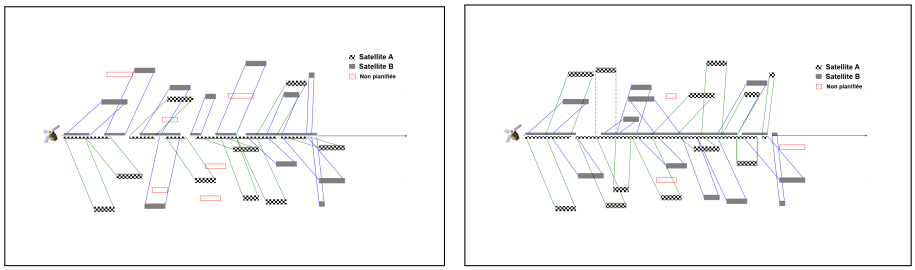


Figure 9. a et b. Comparaison de missions planifiées avec ChronoG (a) et ATLAS (b)

Pour illustrer le partage des acquisitions entre les satellites, nous avons généré un scénario particulier. Dans ce dernier, deux satellites *agiles* identiques (que nous nommerons A et B) se suivent, cette agilité leur permettant 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 9 a et b sont des représentations des tâches que les satellites ont effectué 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 9a), le satellite A n'a que seulement neuf tâches de planifiées et B en compte onze. En comparaison, la mission planifiée par ATLAS comporte vingt-quatre prises de vues. 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 confrontées 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 traitées 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 s'auto-adaptent et modifient 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 distribuée entre les agents qui ont de simples tâches à effectuer. Les interactions locales sont très simples et permettent ainsi de traiter au mieux ce problème complexe.

Enfin, la distributivité des interactions permet de proposer des solutions temps réel. Si le système ATLAS est arrêté, une solution existe et elle concerne l'ensemble de la période de planification. De plus, cette solution est équilibrée et privilégie les demandes clients urgentes. Au contraire, les approches gloutonnes construisent le plan de façon itérative (chronologique ou par priorité). Ainsi, tant que l'ensemble des requêtes clients 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 Maille, et par le coût indiqué par les agents Satellite. Ces deux moteurs de la coopération permettent ainsi d'assurer qu'un nombre important de requêtes soit planifié de façon équilibrée entre les satellites de la constellation. De plus, les requêtes peuvent être prises 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 obtient 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 très encourageants et prouvent l'intérêt d'une approche par système multi-agent adaptatif pour planifier efficacement une constellation de satellites.

Nos futurs travaux concernent l'ajout d'un nouveau niveau d'adaptation. Les créneaux réservés dans les plans de mission pourront se déplacer (en respectant leurs contraintes) et ainsi faire de la place pour de nouvelles acquisitions sans avoir à se dé-

planifier. Ceci permettra d'atteindre de meilleures solutions. Le comportement manquant des agents Requête sera aussi implémenté. La mesure de criticité des agents Maille tiendra compte de cette influence. De plus, ATLAS sera comparé à d'autres approches d'optimisation pour valider notre système. Enfin, ATLAS sera aussi testé sur des cas d'utilisations opérationnels. Ainsi, nous pourrons comparer nos résultats avec des plans de mission développés et élaborés par le CNES (*Centre National d'Études Spatiales*) pour les besoins des missions d'observations de la Terre ou Airbus D&S-Geo, deux organismes qui utilisent et 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 *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.

Boes J., Nigon J., Verstaevel N., Gleizes M.-P., Migeon F. (2015). The Self-Adaptive Context Learning Pattern: Overview and Proposal. In *CONTEXT 2015*. Cyprus.

Bonjean N., Mefteh W., Gleizes M.-P., Maurel C., Migeon F. (2014). Adelfe 2.0. In *Handbook on agent-oriented design processes*, p. 19–63.

Bonnet G. (2008). *Coopération au sein d'une constellation de satellites.* These de Doctorat.

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 *JFPC, 2014.*

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 *JFSMA, 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*. These de Doctorat.

Frank J., Ari J., Morris R., Smith D. E., Field M. (2001). Planning and Scheduling for Fleets of Earth Observing Satellites. In *International Symposium on Artificial Intelligence, Robotics, Automation and Space.*

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., Flipo A., Verfaillie G. (2011). Planning and replanning for a constellation of agile Earth observation satellites. *International Conference on Automated Planning and Scheduling.*

Jordehi A. R., Jasni J. (2015). Particle swarm optimisation for discrete optimisation problems: a review. *Artificial Intelligence Review*, vol. 43, n° 2, p. 243–258.

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

Lemaître M., Verfaillie G., Jouhaud F., Lachiver J. M., Bataille N. (2002). Selecting and scheduling observations of agile satellites. In *Aerospace Science and Technology vol 6*.

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.

Mavrovouniotis M., Müller F. M., Yang S. (2015). An ant colony optimization based memetic algorithm for the dynamic travelling salesman problem. In *Proceedings of the 2015 on genetic and evolutionary computation conference*, p. 49–56.

Modi P. J., Shen W.-M., Tambe M., Yokoo M. (2005). Adopt: Asynchronous distributed constraint optimization with quality guarantees. In *Artificial Intelligence vol 161.*

O Ramos G. de, Rial J. C. B., Bazzan A. L. (2013). Self-adapting coalition formation among electric vehicles in smart grids. In *Self-adaptive and self-organizing systems 2013*

Tangpattanakul P., Jozefowiez N., Lopez P. (2015). A multi-objective local search heuristic for scheduling earth observations taken by an agile satellite. *European Journal of Operational Research*, vol. 245, n° 2, p. 542–554.

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.

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.

Yuan Z., Chen Y., He R. (2014). Agile earth observing satellites mission planning using genetic algorithm based on high quality initial solutions. In *IEEE Congress on Evolutionary Computation*.