Dynamic meshing of large area coverage using adaptive multi-agent systems

Timothée Jammot

RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

Université de Toulouse

Doctorat de l'Université de Toulouse

préparé à l'Université Toulouse III - Paul Sabatier

Maillage dynamique pour l'acquisition de grandes couvertures par systèmes multi-agents adaptatifs

---

Thèse présentée et soutenue, le 18 juin 2024 par

Timothée JAMMOT

École doctorale
EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse

Spécialité
Informatique et Télécommunications

Unité de recherche
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Elsy KADDOUM et Emmanuel RACHELSON

Composition du jury
M. René MANDIAU, Rapporteur, Université Polytechnique des Hauts-de-France
M. Vincent CHEVRIER, Rapporteur, Université de Lorraine
M. Jean-Pierre GEORGÉ, Examinateur, Université Toulouse III - Paul Sabatier
Mme Nadia KABACHI, Examinatrice, Université Claude Bernard Lyon 1
Mme Elsy KADDOUM, Directrice de thèse, Université Toulouse - Jean Jaurès
M. Emmanuel RACHELSON, Co-directeur de thèse, ISAE-SUPAERO

Membres invités
M. Serge Rainjonneau, Thalès Alenia Space
M. Mathieu Picard, Airbus Defense and Space

Timothée Jammot

DYNAMIC MESHING OF LARGE AREA COVERAGE USING ADAPTIVE MULTI-AGENT SYSTEMS

Directrice de thèse :
Elsy Kaddoum, MCF, UT2J
Université Paul Sabatier

Résumé

Le domaine de l'observation de la Terre regroupe différents types d'appareils pour la prise de vue ou acquisition d'images terrestres. Les satellites d'observation en orbite basse sont conçus pour prendre des photographies haute résolution depuis l'espace. Ils sont utilisés pour des missions d'observation urgentes mais aussi pour des missions long terme dites grandes couvertures. Dans ce cadre une grande zone est photographiée progressivement à chaque passage de satellites qui acquièrent des zones au sol dites mailles. Au terme de la mission les images résultant des mailles sont regroupées pour former la grande couverture.

En raison de la taille de la zone à acquérir les missions de grandes couvertures font appel à plusieurs satellites. La zone doit être acquise par ces satellites de façon à minimiser le temps de complétion de la mission et à maîtriser l'utilisation des ressources satellitaires. Le problème d'optimisation défini est complexe. De nombreux facteurs rendent la résolution en temps raisonnable difficile, comme la taille de la zone, l'hétérogénéité des caractéristiques des satellites et les prévisions météorologique. L'utilisation d'un algorithme d'optimisation en méthode approchée est ainsi nécessaire.

Le grand nombre d'entités du problème et le besoin d'adaptabilité justifient l'utilisation d'un algorithme par raisonnement décentralisé. Dans cette thèse nous adressons la résolution du problème des grandes couvertures par systèmes multi-agents adaptatifs (AMAS). Nous proposons d'abord une nouvelle méthode de maillage, le maillage dynamique, qui permet un positionnement précis des mailles sur la zone entraînant ainsi une nouvelle formalisation du problème des grandes couvertures. Pour résoudre ce problème nous présentons le système AMAS Glimpse basé sur le modèle agent AMAS4Opt. Glimpse est ensuite validé sur divers scénarios simulés.

Une comparaison avec un algorithme glouton illustre les apports de notre méthode pour l'optimisation des métriques du problème. Glimpse réduit l'utilisation de ressources satellitaires en évitant les acquisitions superflues tout en minimisant le temps de complétion requis pour traiter de très grandes couvertures. Les résultats obtenus ouvrent de nouvelles directions de recherche dans les domaines des AMAS et des grandes couvertures.

Institut de Recherche en Informatique de Toulouse - UMR 5505
Université Paul Sabatier, 118 route de Narbonne, 31062 TOULOUSE cedex 4

Timothée Jammot

DYNAMIC MESHING OF LARGE AREA COVERAGE USING ADAPTIVE MULTI-AGENT SYSTEMS

Supervisor :
Elsy Kaddoum, MCF, UT2J
Université Paul Sabatier

Abstract

In the Earth observation field a variety of devices are used to take pictures or acquire ground images. Low Earth orbit satellites are designed to acquire high resolution images from space. They are used for urgent missions but also long term ones called Large Area Coverage. In this context a large area is acquired gradually for each pass of satellites that acquire small surface areas called meshes. At the end of the mission the images acquired from the meshes are regrouped to form the large coverage.

Due to the size of the area to acquire large area coverage missions require the use of several satellites. The area needs to be acquired by these satellites so as to minimize the time to mission completion and optimize the use of satellite resources. The resulting optimization problem is complex. Many parameters render the solve in reasonable time difficult, such as the size of the area, the heterogeneity of the satellite characteristics and the weather forecasts. The use of an approximate optimization method is necessary.

The high number of problem entities and the need for adaptability justify the use of a decentralized reasoning algorithm. In this thesis we address the solve of the large area coverage problem by adaptive multi-agent systems (AMAS). We first propose a new meshing technique, dynamic meshing, that allows for a precise positioning of meshes on the area and thus introduces a new formalization of the Large Area Coverage problem. To solve this problem we present the Glimpse AMAS based on the AMAS4Opt agent model. Glimpse is then validated on different simulated scenarios.

A comparison with a greedy algorithm illustrates the contributions of our method for optimizing the problem metrics. Glimpse reduces the use of satellite resources by avoiding superfluous acquisitions while minimizing the time required to acquire very large areas. The results open new research directions in the AMAS and large area coverage fields.

Institut de Recherche en Informatique de Toulouse - UMR 5505
Université Paul Sabatier, 118 route de Narbonne, 31062 TOULOUSE cedex 4

Remerciements

JE voudrais conclure ces années de thèses par des remerciements pour tous ceux qui m'ont encadré, m'ont aidé dans mes travaux ou étaient tout simplement là pour me soutenir.

Merci d'abord à mes rapporteurs, Pr. René Mandiau et Pr. Vincent Chevrier. Vos retours très positifs, notamment au niveau de la pédagogie du manuscrit, viennent valider de longues heures de travail. Merci également à mes examinateurs, Dr. Jean-Pierre Georgé et Dr. Nadia Kabachi d'avoir accepté de relire le manuscrit et participé à mon jury et aux discussions.

Je voudrais aussi remercier mon équipe d'encadrement. Elsy, Emmanuel, Mathieu, Serge, j'ai apprécié travailler avec chacun d'entre vous tout au long de la thèse. Une des difficultés de cette thèse était l'encadrement par quatre organismes différents mais vous m'avez toujours bien conseillé et j'ai beaucoup appris de votre cohésion dans mon encadrement.

Je souhaite notamment adresser un très grand merci à Elsy pour m'avoir accompagné et soutenu dans les moments difficiles. J'espère avoir ton assiduité et ta persévérance un jour. Merci pour tous les efforts que tu as fait pour moi, je souhaite sincèrement à tout doctorant d'avoir une directrice de thèse telle que toi.

Je veux aussi remercier l'IRT Saint-Exupéry pour le financement de cette thèse. Merci à Grégory de m'avoir accueilli dans l'équipe OCE, et merci à Guillaume d'avoir fait de même dans l'équipe Synapse. J'ai été ravi de pouvoir continuer sur la lancée de mes stages avec cette thèse et même de pouvoir la présenter à l'étranger, merci de m'avoir accordé votre confiance.

J'ai reçu beaucoup d'aide durant le développement de Glimpse à l'IRT, merci à Adrien, Clément, Rémi, Mikaël et Hugo pour votre bonne humeur et vos conseils. Adrien et Clément la présentation de nos articles à Berkeley reste une fantastique expérience pour moi, merci pour ces bons moments partagés.

Merci à toute l'équipe SMAC de m'avoir accueilli pendant mes stages de master et la thèse. Frédéric, Marie-Pierre, vous m'avez écouté quand j'avais besoin de vous parler et cela m'a sans aucun doute beaucoup aidé. Je sais que tu préfères ne pas être nommé mais je vais continuer la tradition et te remercier, Pierre, pour tes discussions sur les AMAS, toujours intéressantes et venant bousculer mes conceptions souvent trop figées.

Je voulais aussi remercier mes camarades doctorants de « promotion » SMAC, Bruno, Davide, Guilhem, Jean-Baptiste, Kristell, Maroun, Maxime et Walid. J'ai toujours pu partager mes problèmes de thèse avec vous et ce fut un plaisir de vous aider en retour. Vous m'avez aussi aidé à combattre ma timidité de début de thèse et je pense avoir beaucoup gagné en maturité notamment grâce à vous.

Je finis cette thèse en tant qu'ingénieur logiciel chez Spacebel et je tiens à remercier Pierre et Fabien de m'avoir accordé le temps de clore cette page dans de bonnes conditions. La fin de thèse est une période difficile mais grâce à vous j'ai pu bénéficier d'une certaine tranquillité d'esprit dans la dernière ligne droite.

Merci aussi à mes amis, Anselme, Benjamin, Coralie, Elisabeth, Elodie, Guillaume, Lucie, Rémi, d'avoir été là pour écouter mes avancées dans la thèse et de toujours y avoir répondu

avec le sourire.

Un grand merci également à mes parents et mes sœurs. Vous avez été là pour discuter de mes projets et pour m'encourager, quelle que soit la piste que je prenais. Je n'aurais sûrement pas pu aller aussi loin sans votre soutien, merci pour tout.

Enfin, merci Anaïs. Tu connais les écueils de la rédaction et tu m'as toujours encouragé avec toute la gentillesse et la bienveillance du monde. Merci d'avoir été à mes côtés jusqu'au bout de cette aventure.

Contents

Contents (detailed)

Acronyms

ADELFE	Atelier de Développement de Logiciels à Fonctionnalité Emergente
AMAS	Adaptive Multi-Agent System
AMAS4Opt	AMAS for Optimisation
AOI	Area Of Interest
ALNS	Adaptive Large Neighborhood Search
COP	Constraint Optimization Problem
CSP	Constraint Satisfaction Problem
DRL	Deep Reinforcement Learning
EA	Evolutionary Algorithm
EP	Evolutionary Programming
ES	Evolution Strategy
GA	Genetic Algorithm
GP	Genetic Programming
GUI	Graphical User Interface
ILP	Integer Linear Programming
IoT	Internet of Things
LAC	Large Area Coverage
LARM	Large Area Request Manager
LEO	Low Earth Orbit
MBDE	Multi-objective Binary-encoding Differential Evolution
MCTS	Monte-Carlo Tree Search
MDP	Markov Decision Process
MPF	Mission Planning Facility
NCS	Non Cooperative Situation
OOP	Object Oriented Programming
SA	Simulated Annealing

Introduction

As part of Earth observation missions, satellites are used to acquire areas on the Earth's surface. Large area coverage missions for example are large-scale acquisition projects. They require the use of a set of satellites that make several passes over a long acquisition period. Splitting an area of interest into sub-areas called meshes that can be acquired by satellites during their passes is a difficult optimization problem.

In this thesis, we present the **Glimpse** system (self-adaptive and learnin**G** mu**L**t**I**-agent model for **M**eshing and **P**lanning **S**ystems in dynamic **E**nvironments) based on adaptive multi-agent systems for the dynamic splitting and positioning of meshes on a grid of elementary sub-areas. The experiments carried out and the comparison with an approach that is part of the operational state of the art show that a decentralised and dynamic approach to solve this problem is adequate.

Context and Motivation

Earth observation from space is a relatively recent topic with varied applications including topography, oceanography, mapping. Short term and small scale events such as disaster control need to be visualized as soon as possible while other long term missions may be completed after weeks or months of sequential acquisitions. Planning the acquisitions of observation satellites is a field that has seen increasing complexity in recent years in correlation with satellite availability and on-board technology. The need for higher resolution and faster imaging lead to the development of agile satellites and constellations able to acquire any zone on the surface of the Earth within hours. Building and launching these satellites is costly and **each acquisition performed needs optimized methods to obtain clear images as early as possible**.

Optimization of small scale imaging missions for a single or a few satellites is a topic that has seen interest in the literature. Meanwhile, the optimization of Large Area Coverage missions covering large surfaces, spanning over months and using satellites with differing characteristics is a less known and researched problem. The complexity of the problem increases with the number of acquisitions required to complete the missions and the number of satellites available to perform them. **Satellites coordination is necessary to minimize the number of redundant tasks being performed by each satellite.** Issues common to short

term acquisitions remain, including the importance of weather forecast with over half of performed acquisitions leading to blurred images due to partial cloud coverage.

The increased complexity of the Large Area Coverage problem led us to consider **resolution algorithms able to navigate a large research space** with additional information during the resolution. In this thesis we focus on solving the problem with an **Adaptive Multi-Agent System**. This type of algorithm uses elementary code components called agents to cooperatively solve the problem. Each agent tries to satisfy simple goals by communicating with a local neighborhood of agents. The emerging behaviour of the algorithm leads to the cooperation between satellites to acquire different surfaces with high probability of successful acquisition leading to clear images.

Manuscript organisation

- The first chapter proposes a summary of the work presented in this thesis in french. We briefly present the Large Area Coverage problem, our contributions, and the main results obtained. A short conclusion then proposes an overview of our approach and the potential research continuations to this work.
- The second chapter describes the **components of Earth observation missions** and introduce the process of acquisition from client requests to reception of clear images of requested areas. We also present the Large Area Coverage optimization problem and estimate resolution complexity.
- The third chapter gives **metrics** to evaluate the quality of obtained solutions to the Large Area Coverage problem and describes the **operational methods** to address the problem in the industry. We introduce our **theoretical contribution** with a state of the art to discuss the operational methods used and possible resolution algorithms with their application to similar problems in the literature.
- The fourth chapter presents the theory behind **Adaptive Multi-Agent Systems** and agent cooperation. We then describe the tools used during the thesis to help with the conception and development of a multi-agent system.
- The fifth chapter introduces our practical contribution with the multi-agent system **Glimpse**. We first describe its agents with their available actions and goals, and then the cooperative behaviors between agents and their motivations.
- The sixth chapter presents the structures of the scenarios built to **evaluate** the quality of solutions found using Glimpse. We build a list of scenarios designed to test different criteria including the scalability and robustness of resolution algorithms. We then discuss obtained results relative to the goal of each experiment.
- The **conclusion** presents the **main results** about the approach including dynamic meshing and using an Adaptive Multi-Agent System and describes the **improvements** available to Glimpse for the optimization of the Large Area Coverage problem or similar problems.

- The annex proposes the improved version of the greedy algorithm for the solving of the Large Area Coverage problem used to benchmark the results of Glimpse. This hybrid greedy and Simulated Annealing algorithm provides a better comparison for future performance benchmarks of Glimpse.

1 Glimpse : maillage de grandes couvertures par systèmes multi-agents

Les satellites d'observation en orbite basse ont pour objectif de prendre des photos de zones d'intérêt à la surface de la Terre. Les missions de grandes couvertures ou Large Area Coverage (LAC) concernent des demandes d'acquisition de grandes zones, à l'échelle d'un pays ou d'un continent. Les missions LAC fournissent des images utilisées dans des domaines tels que la cartographie et la surveillance. Ces missions sont coûteuses en termes de ressources satellitaires et de temps. L'utilisation d'un seul satellite pour effectuer les missions LAC est inefficace et peut entraîner des décalages temporels entre les images. C'est pourquoi un ensemble de satellites est utilisé pendant une longue période pour assurer la couverture.

Ces satellites peuvent être hétérogènes, avec des caractéristiques et des instruments d'acquisition différents, et exploités par des centres de planification indépendants et non communicants. L'un des défis des missions actuelles d'observation par satellite est la coordination des centres de planification pour une résolution réussie des missions LAC par un ensemble de satellites hétérogènes. Une meilleure coordination permettra une meilleure utilisation des satellites pendant leur durée de vie limitée et des images de couverture plus facilement accessibles.

Cette thèse (rédigée en anglais à partir du chapitre 2) a pour objectif de proposer une approche novatrice à la résolution du problème d'observation satellitaire des **grandes couvertures**. Ce chapitre (rédigé en français) présente un résumé général du travail de recherche effectué. Il s'organise comme suit. La section 1.1 décrit le problème des grandes couvertures. La section 1.2 propose un état de l'art des algorithmes de résolution et techniques de maillage appliqués à ce problème. La section 1.3 présente nos contributions avec le maillage dynamique, une technique de maillage innovante, et Glimpse, un algorithme de résolution du problème des grandes couvertures par systèmes multi-agents adaptatifs. La section 1.4 décrit les résultats d'expérience obtenus pour l'évaluation des performances et de la robustesse de Glimpse. Finalement la section 1.5 conclut et présente diverses suites possibles à cette étude.

1.1 Le problème des grandes couvertures

Le problème LAC est l'optimisation de l'acquisition d'une zone d'intérêt définie par la demande d'un client à l'aide d'un ensemble de satellites exploités par le centre de planification ou Mission Planning Facility (MPF) qui reçoit la demande du client.

1.1.1 Traitement d'une requête d'acquisition

Un **satellite** est un outil d'acquisition suivant une orbite polaire et exploité par un MPF. En fonction de son orbite et de la rotation de la terre, le satellite survole des zones terrestres à des intervalles de temps appelés **passages**. Lors d'un passage donné, une zone géographique appelée **corridor** est accessible par le satellite. La largeur d'un couloir dépend de l'agilité des satellites, c'est-à-dire de leurs capacités d'orientation, et de la résolution minimale des images obtenues par acquisition. Cette terminologie liée aux satellites d'observation est présentée en figure 1.1.

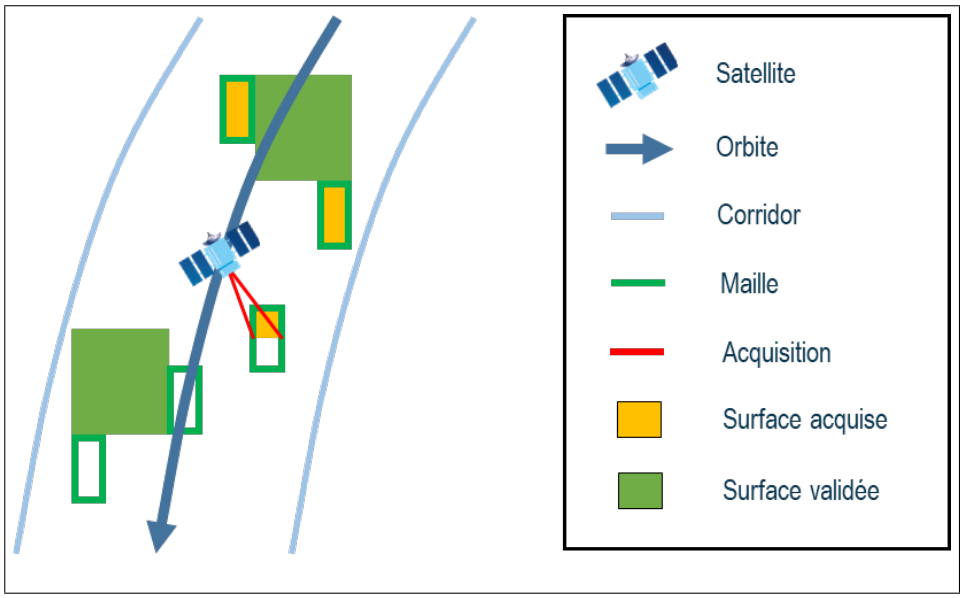

Figure 1.1: Terminologie des satellites d'observation de la Terre

Une première phase en amont du problème LAC consiste à établir l'ensemble des satellites et leurs passages qui peuvent être utilisés pour acquérir la zone d'intérêt. Ensuite, pour chaque passage d'un satellite au-dessus de la zone d'intérêt, le MPF en charge de la demande du client suit un processus d'acquisition divisé en 5 phases présentées dans la figure 1.2 :

1. **Requête d'acquisition.** Réception de la requête d'acquisition client par le MPF. Les paramètres de la requête comprennent la zone à acquérir, l'urgence d'acquisition et le délai maximum avant péremption de la requête.

2. **Maillage et planification.** Placement d'une ou plusieurs mailles qui couvrent la zone d'intérêt. Ces mailles font l'objet d'acquisitions dans un plan d'acquisition transmis au

satellite.

3. **Acquisition.** Réalisation des acquisitions par le satellite lors de l'exécution de son plan d'acquisition. Cette exécution peut être espacée de la réception du plan de plusieurs heures. La météo prévue peut donc être différente de celle observée lors de la réalisation des acquisitions.

4. **Validation ou rejet.** Réception des images acquises par le MPF. Les images sont ensuites jugées utilisables ou non suivant la nébulosité constatée. Les images utilisables sont conservées tandis que les zones nuageuses doivent être réintégrées dans le processus d'acquisition depuis l'étape de maillage et planification.

5. **Images validées.** Les images validées forment un ensemble qui couvrent totalement la zone d'intérêt donnée dans la requête d'acquisition client. Ces images sont renvoyées vers le client pour terminer le processus d'acquisition.

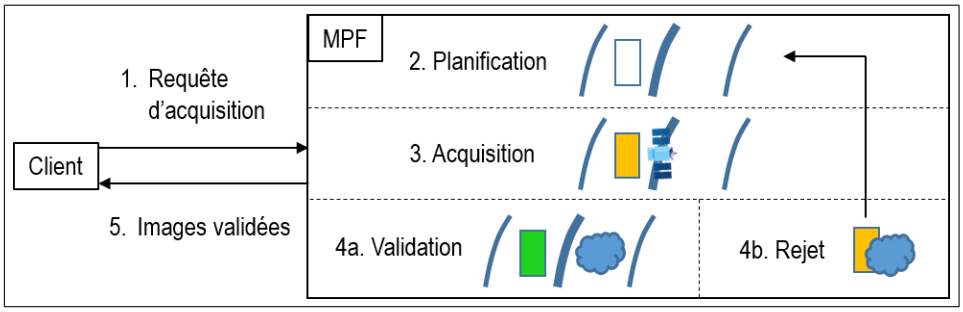

Figure 1.2: Processus de traitement d'une requête d'acquisition par un MPF

1.1.2 Maillage de grandes couvertures

La résolution d'une mission LAC implique le maillage et la transmission des demandes d'acquisition aux MPF qui exploitent le satellite. Les MPF ont chacun son processus de planification qui est considéré comme une boîte noire pendant la résolution. Les demandes d'acquisition plus urgentes que celles émises par la mission LAC peuvent être traitées en priorité par les MPF. Les demandes d'acquisition de mailles envoyées aux MPF ne sont donc pas garanties d'aboutir à l'acquisition de toutes ces mailles.

En outre, la réussite de l'acquisition d'une maille par un satellite dépend des conditions météorologiques au moment de l'acquisition. L'image obtenue par une acquisition sera considérée comme claire ou floue en fonction du taux de nébulosité observé. L'acquisition d'une maille par un MPF peut donc échouer en raison de différences entre les prévisions météorologiques faites lors de l'envoi du plan d'acquisition au satellite et les conditions réelles au moment du passage. Pour chaque maille, trois états sont à distinguer :

1. **Active** Maille en attente de la planification du passage sur lequel elle est positionée.

2. **Acquise.** Maille incluse dans un plan d'acquisition du satellite et qui sera acquise.

3. **Validée.** Maillage dont l'acquisition est réussie et pour laquelle une image sans nuage est disponible.

Le problème LAC consiste à trouver un ensemble de mailles à acquérir pour chaque passage de satellite qui minimise à la fois le **temps de complétion** et les **ressources satellitaires** nécessaires à l'obtention d'images sans nuage de l'ensemble de la zone d'intérêt. Le temps est mesuré par le nombre de passages des satellites depuis le début de la mission LAC ou l'équivalent en nombre de jours. Les ressources satellitaires utilisées jusqu'à la fin de la mission LAC sont mesurées par une métrique appelée **gâchis**, définie comme l'acquisition excédentaire de zones au cours de la mission. Par exemple, le gâchis d'acquisition minimal de 0% est atteint si toutes les acquisitions validées de la grande couverture concernent des zones qui n'ont été acquises qu'une seule fois.

L'entité chargée d'assurer la coordination des MPFs pour couvrir une grande couverture est nommée Large Area Request Manager (LARM). Un LARM dispose des informations relatives aux MPFs, aux caractéristiques de leurs satellites respectifs et de leurs passage sur la zone d'intérêt de la grande couverture. Un LARM suit un processus d'acquisition parallèle à celui des MPFs qui permet d'acquérir et de valider la totalité des images relatives à la zone d'intérêt. Ce processus est constitué des étapes suivantes aussi présentées en figure 1.3 :

1. **Maillage.** Découpage du couloir de passage en mailles.
2. **Requêtes d'acquisition.** Envoi au MPF responsable de la planification du satellite d'une ensemble de requêtes d'acquisition concernant les mailles sélectionnées.
3. **Réception d'images.** Collecte des retours du MPF et enregistrement des images ayant pu être acquise dans la grande couverture.

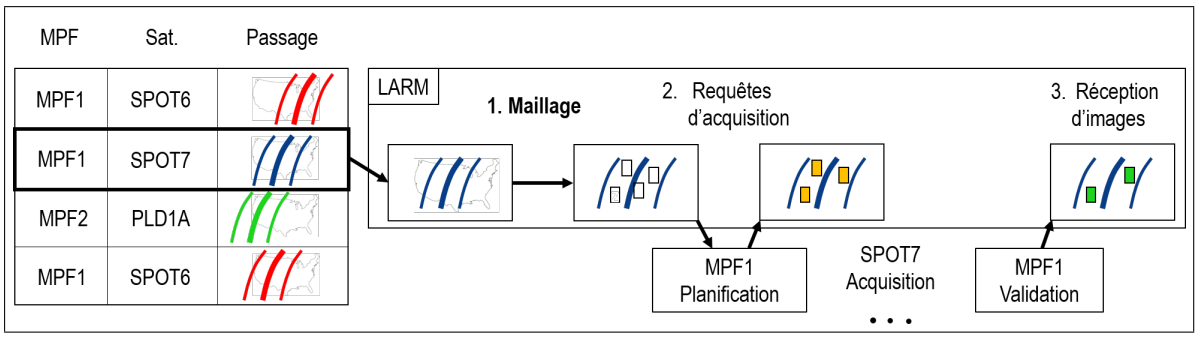

Figure 1.3: Processus entre LARM et MPFs pour l'acquisition de mailles sur le passage d'un satellite

1.2 Techniques de maillage

Le problème LAC tel qu'il est présenté dans la section précédente n'est pas traité dans son ensemble dans l'industrie. **Une première approche**, couramment rencontrée dans l'industrie, consiste à appliquer le prétraitement suivant :

1. Division de la zone d'intérêt en plusieurs sous-zones,
2. Répartition exclusive des sous-zones entre les MPF,
3. Acquisition par les MPF de leurs sous-zones respectives.

Ce prétraitement permet d'éviter de submerger les MPF de demandes d'acquisition sur l'ensemble de la zone d'intérêt. La coordination entre les MPF n'est pas nécessaire car chaque sous-zone est traitée séparément. La répartition entre les MPF permet ainsi d'éviter la complexité du problème LAC sur l'ensemble de la zone d'intérêt. Cependant, cette approche implique que certains passages de satellites seront ignorés pour les sous-zones car une autre sous-zone leur a été attribuée. De nombreuses opportunités d'acquisition sont ignorées pour simplifier le problème et le temps d'achèvement de la mission augmente de manière significative. Le problème LAC, considéré dans sa globalité, présente une complexité accrue mais permet la coordination des MPF pour des missions à la fois plus rapides et moins coûteuses en termes de ressources satellitaires.

Une autre technique que nous pouvons souligner dans le contexte opérationnel est le **maillage statique**, soit l'utilisation d'une grille de maillage. Dans ce cas une grille de maillage est établie par le MPF une fois qu'une sous-zone de la zone d'intérêt lui a été attribuée. Cette technique consiste à construire pour chaque passage du satellite un ensemble d'emplacements de mailles qui peuvent être acquis. Une grille de mailles jointives formant un pavage régulier sur le couloir du passage est calculée à la réception de la demande du client. Les mailles obtenues par maillage statique sont fixes et de taille constante dans le même passage. Cette taille est définie en fonction du maximum des capacités d'acquisition du satellite pour une seule maille. Le positionnement et les dimensions fixes des mailles ne permettent pas de les adapter aux besoins d'acquisition géographique au cours de la mission. Ces contraintes peuvent entraîner des acquisitions supplémentaires de sous-zones validées.

Pour surmonter cette limite, nous introduisons une nouvelle technique de maillage appelée **maillage dynamique**. Une grille de sous-zones élémentaires appelées **cellules** est construite sur la zone d'intérêt. Cette discrétisation permet un positionnement plus précis des mailles dans les couloirs des passages, et en particulier sur leurs bords. La position des mailles et leurs dimensions peuvent également être adaptées sur la grille de cellules pour une meilleure utilisation des ressources satellitaires. Les sous-parties de couloirs validées ou nuageuses peuvent être ignorées au profit de sous-parties non validées et dont la probabilité de validation est plus élevée. L'exemple de la figure 1.4 illustre le placement d'un ensemble de mailles sur des emplacements de mailles, dans le cas du maillage statique, et sur une grille de cellule, dans le cas du maillage dynamique.

La distribution de mailles de tailles variées sur une grille de cellules pour chaque passage de satellite est une tâche complexe. Le problème LAC est un problème d'optimisation combinatoire **NP-Hard** comme un sous-cas des problèmes de planification de couverture [Lemaître *et al.*, 2000]. Nous avons choisi d'implémenter un système multi-agent pour optimiser le problème LAC par maillage dynamique. Le système présenté dans cet article est basé sur la théorie AMAS [Gleizes *et al.*, 2008; Georgé *et al.*, 2003; Glize, 2001; Gleizes *et al.*, 1999]. Les systèmes multi-agents adaptatifs ont prouvé leur efficacité dans la résolution de

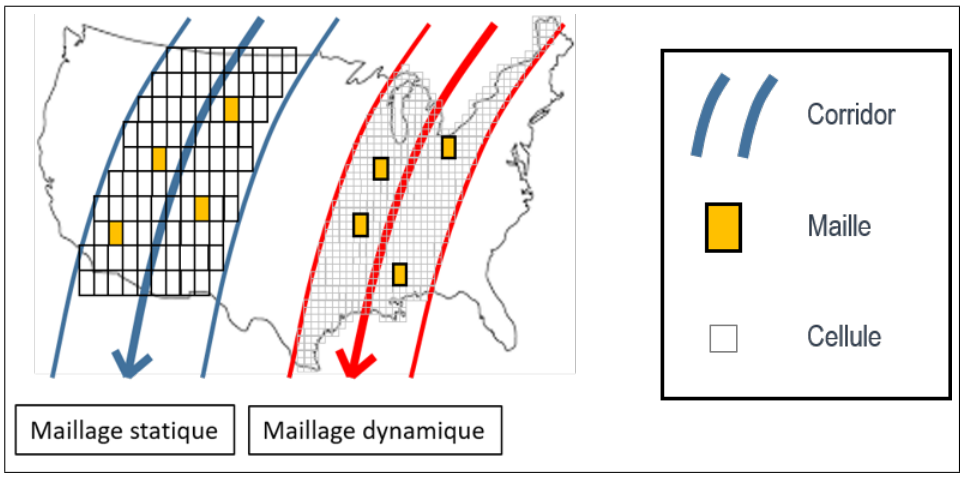

Figure 1.4: Illustration du placement de mailles suivant la technique de maillage, statique ou dynamique

problèmes d'optimisation hautement combinatoires comprenant un grand nombre d'entités par l'émergence de fonctions globales de bonne qualité grâce à la coopération d'agents distribués ayant des comportements locaux simples [Perles, 2017; Brax, 2013; Bourjot *et al.*, 2003; Verstaevel *et al.*, 2016; Guastella, 2020].

1.3 Le système Glimpse

Glimpse compte trois types d'agents : les agents Cellule, Maille et Passage. Ces agents suivent un cycle de vie constitué de phases séquentielles :

1. **Perception**, réception des messages et mise à jour des données internes,
2. **Décision**, traitement des données et choix d'actions à effectuer,
3. **Action**, envoi de messages et modification de l'environnement de l'agent.

Dans cette section, nous décrivons les agents du système par leurs objectifs, leurs interactions et leurs comportements représentés par des algorithmes de décision.

Le maillage statique couramment utilisé est une heuristique de placement des mailles au sein des passages qui présente l'avantage de ne pas autoriser de chevauchements entre mailles. Le maillage dynamique proposé dans cette étude apporte à Glimpse un degré de liberté supplémentaire dans le maillage. Les mailles peuvent être placées sur n'importe quelle coordonnée (i, j) appartenant au corridor de leur passage. Le premier objectif des comportements des agents dans Glimpse est de trouver pour chaque agent Maille un placement coopératif prenant en compte la criticité des agents Cellule qui seront acquis avec l'agent Maille.

1.3.1 L'agent Cellule

Dans Glimpse, la **criticité des agents Cellule** représente l'importance d'acquisition de la zone couverte par la cellule pour un passage donné. Les criticités de plusieurs cellules peu-

vent être comparées pour déterminer leur ordre d'importance d'acquisition sur un passage. La comparaison suit un ordre lexicographique : les critères sont normalisés entre eux et comparés dans un ordre fixe jusqu'à trouver un critère différenciateur qui indique l'agent Cellule le plus critique. La comparaison de critères issus des données du problème implique la définition d'une importance relative de ces données.

Dans le cadre de ce travail, l'importance des critères est issue d'analyses d'experts du domaine. La criticité des agents Cellule dans un passage donné est basée sur les critères suivants, dans leur ordre lexicographique :

1. **Reste à acquérir.** La criticité d'une cellule déjà validée est nulle.
2. **Probabilité de validation.** Probabilité de validation de la cellule si acquise dans ce passage. Pour une cellule acquise dans des passages antérieurs, elle est pondérée par les probabilités de validation dans ces passages.
3. **Probabilité de validation future.** Probabilité de validation de la cellule dans les passages restants sur l'horizon de planification pour lesquels le corridor couvre cette cellule.
4. **Proximité de cellules validées.** Ratio de cellules voisines déjà validées sur des passages antérieurs.
5. **Opportunités d'acquisition.** Ratio de passages restants sur l'horizon de planification pour lesquels le corridor couvre cette cellule.

Les agents Cellule ont pour objectif d'être acquis puis validés sur n'importe quel passage sur l'horizon de planification qui contient la cellule dans son corridor. Le comportement d'un agent Cellule se base sur ses connaissances locales, notamment les agents Maille visibles par l'agent Cellule dans chaque passage pour maximiser la probabilité d'acquisition de la cellule. La visibilité des agents Cellule est limitée aux mailles qui les recouvrent ou sont directement adjacentes sur la grille de cellules.

La résolution de l'insatisfaction des agents Cellule se traduit par une notion d'adhésion à des mailles sur les passages connus. L'adhésion d'une cellule à une maille décrit la volonté de faire partie des cellules couvertes par cette maille dans le passage correspondant. Une cellule ne peut adhérer qu'à une seule maille par passage, soit la maille qu'elle préfère et qui a accepté sa requête d'adhésion parmi les mailles qu'elle connaît dans ce passage.

L'objectif de l'adhésion est de promouvoir la répartition géographique des mailles dans le corridor des passages. Cette répartition est nécessaire en maillage dynamique pour éviter le regroupement et chevauchement des mailles d'un même passage sur les cellules les plus critiques dans ce passage. Les agents Cellule choisissent la maille à laquelle ils préfèrent adhérer suivant un tri par criticité des agents Maille visibles dans leur voisinage.

1.3.2 L'agent Maille

Les agents Maille ont pour objectif d'être acquis puis validés dans leur passage. Pour ce faire, ils utilisent les requêtes d'adhésion des agents Cellule dans leur voisinage pour altérer

leur géométrie et leur placement dans le corridor de façon à maximiser leur probabilité d'acquisition et de validation. Une requête d'adhésion traitée par un agent Maille peut faire l'objet de trois types d'altération de sa géométrie sur la grille de cellules :

- **Déplacement.** Les coordonnées d'origine (i_0, j_0) de la maille sont modifiées.
- **Extension.** Le facteur f de largeur ou le facteur g de longueur de la maille est étendu dans un sens donné.
- **Réduction.** Le facteur f de largeur ou le facteur g de longueur de la maille est réduit dans un sens donné.

L'inclusion d'une cellule fait l'objet d'une extension de la maille tant qu'elle n'a pas encore atteint sa largeur maximale f_k^{max} ou longueur maximale g_k^{max} en nombre de cellules, suivant la direction relative de la cellule qui émet une requête d'adhésion. Lorsque la largeur ou longueur maximale est atteinte un déplacement peut être effectué. Ce déplacement causerait l'inclusion à la maille d'un ensemble de cellules et l'exclusion de cellules sur le bord opposé au sens du déplacement.

Dans ce cas une comparaison entre les criticités des cellules à inclure suite au déplacement potentiel et les criticités des cellules à exclure est simulée au sein de l'agent Maille. Le déplacement est effectué si la criticité des cellules qui adhéreront à la maille suite au déplacement est supérieure car cette altération augmente la criticité de la maille elle-même. La **criticité des agents Maille** dépend des criticités des agents Cellule qui adhèrent à elle, triées par ordre décroissant.

La géométrie libre des mailles permet aussi la réduction des bords de la maille qui ne comprennent pas de cellules en adhésion. Ces réductions et l'adhésion des cellules aux mailles les plus critiques dans leur voisinage favorisent la formation de mailles juxtaposées. Ces juxtapositions diminuent le gâchis dû aux chevauchements de mailles et aux acquisitions surnuméraires de mêmes cellules au sein d'un passage.

1.3.3 L'agent Passage

Les comportements des agents Cellule et Maille ont pour objectif de placer les mailles dans leur environnement local pour maximiser leur probabilité d'être acquises et validées. Les altérations et déplacements des mailles sont basés sur leur voisinage de cellules qui émettent des requêtes d'adhésion. Ces comportements ne suffisent pas pour garantir la couverture des cellules les plus critiques à l'échelle du corridor d'un passage. Des sous-parties du corridor peuvent ainsi piéger les mailles dans un optimum de criticité local. La création de mailles par les agents Passage dans leur corridor correspond à un saut dans l'espace de recherche similaire à une augmentation de température d'une optimisation par recuit simulé [Kirkpatrick *et al.*, 1983], ou une mutation d'un algorithme génétique [McCall, 2005].

Les agents Passage ont pour objectif de maximiser la couverture des cellules les plus critiques au sein du corridor d'un passage. Leur comportement est lié au placement des mailles placées sur le corridor et doit permettre aux cellules les plus critiques d'être couvertes de

façon dynamique, par exemple suite à une mise à jour des données météorologiques qui vient modifier la criticité des cellules.

Afin de laisser le placement local des mailles s'effectuer par les interactions entre agents Cellule et Maille, le comportement des agents Passage n'est effectué qu'après la stabilisation des mailles du passage. Les mailles d'un passage sont stabilisées quand toutes les cellules couvertes par des mailles adhèrent à l'une d'elles, et suite à un délai suffisant pour garantir qu'aucun message, notamment des requêtes d'adhésion, n'est en transit ou en cours de traitement. Suite à la stabilisation des mailles, les cellules appartenant au corridor du passage sont parcourues jusqu'à trouver une cellule non couverte ayant une criticité supérieure au point de référence de criticité de la première maille candidate au remplacement, soit la cellule la plus critique de la maille la moins critique.

Le remplacement d'une maille est nécessaire pour garantir un nombre de mailles dans le passage inférieur au seuil maximal défini comme le nombre de mailles suffisant pour couvrir entièrement le corridor du passage. Ce seuil maximal a pour objectif d'éviter la surcharge de requêtes d'acquisition de mailles au centre de planification.

1.4 Résultats et discussion

Cette section présente l'évaluation et les performances du système Glimpse pour l'optimisation de problèmes de grandes couvertures simulés. Les résultats d'expérimentations sont suivis par une discussion et nos perspectives d'amélioration du système.

1.4.1 Méthodologie d'expérimentation

Les scénarios de mission considérés comportent les entrées nécessaires au traitement du problème par un LARM, soit les éléments suivants :

- La requête de programmation client contenant :
  - La zone d'intérêt à acquérir,
  - Les centres de planification,
  - Les satellites et leurs caractéristiques,
  - Les passages des satellites sur la zone.
- Les données météorologiques pour les cellules de chaque passage,
- La taille des cellules de la grille,
- La taille de l'horizon de planification.

Afin de comparer les résultats de Glimpse nous avons choisi de les comparer à un LARM pouvant être utilisé dans un contexte opérationnel. Un algorithme de type glouton hiérarchique est donc choisi. Cet algorithme a un comportement simple de sélection de la meilleure maille à court terme, sans aucune remise en question des choix effectués.

Ce glouton hiérarchique emploie un maillage statique pour placer les mailles dans les corridors des satellites au cours de la résolution de la grande couverture. La technique maillage statique utilisée place des emplacements de mailles de taille maximale autorisée juxtaposées suivant un pavage régulier de sorte à couvrir la zone d'intérêt.

De plus nous considérons des scénarios de missions dont les satellites sont caractérisés par les dimensions des mailles pouvant être acquises. Des scénarios seront décrits comme comportant des satellites **homogènes** si les mailles qu'ils acquièrent sont de mêmes dimensions, et **hétérogènes** sinon.

1.4.2 Performances

Pour chaque expérimentation les métriques observées sont, par ordre d'importance :

1. Le temps de complétion à 95% de la grande couverture, en nombre de passages.
2. Le gâchis, surplus d'acquisition de surfaces déjà acquises au cours de la grande couverture, en pourcentage.

La première expérimentation propose un scénario de mission avec les caractéristiques suivantes :

- L'Australie en zone d'intérêt,
- Des cellules de bord 30km,
- 4 satellites **homogènes** aux mailles de largeur 60km et longueur 120km,
- Un horizon de planification de 10 passages.

Les résultats obtenus montrent des taux de complétion similaires pour les deux algorithmes évalués (Glimpse avec le maillage dynamique et un glouton hiérarchique avec le maillage statique), et des taux de gâchis égaux à 95% de complétion de la grande couverture. Une différence observable est la progression du gâchis plus importante en début de mission pour Glimpse mais qui ralentit par la suite. La progression du gâchis du glouton hiérarchique suit une courbe plus linéaire.

La deuxième expérimentation propose de réutiliser ce scénario en modifiant les dimensions de mailles acquérables par les satellites afin de les rendre **hétérogènes**. Les 4 satellites hétérogènes de ce nouveau scénario peuvent respectivement acquérir des mailles qui ont les dimensions suivantes : 60km/90km, 60km/120km, 90km/150km et 90km/180km.

Les résultats obtenus montrent dans ce cas une amélioration du temps de complétion et du gâchis de l'ordre de 10% en faveur de Glimpse. De façon similaire à la première expérimentation, le gâchis accumulé par Glimpse est supérieur en début de mission mais stagne par la suite. Le gâchis accumulé par le glouton continue de croître au cours de la mission jusqu'à dépasser celui de Glimpse. Ces résultats démontrent une tendance suivant laquelle Glimpse positionne mieux les mailles quand l'espace est plus contraint (zones déjà validées, mauvaise météo) et réduit le gâchis dû aux réacquisitions de mailles dans les phases avancées de la mission LAC.

1.4.3 Robustesse

Les résultats obtenus par Glimpse dans la section précédente ont été obtenus sur quelques cas d'utilisation spécifiques. Afin de développer ces résultats et d'attester de la robustesse de Glimpse dans d'autres cas d'utilisation, nous proposons quatre critères définis comme paramètres de robustesse pour un LARM.

Pour chaque critère, la robustesse de Glimpse est validée par comparaison de Glimpse utilisant un maillage dynamique avec un algorithme glouton sélectionnant des mailles à partir d'une grille de maillage statique. Les résultats soulignent la robustesse des algorithmes et des techniques de maillage dans différents scénarios simulés. Les quatres critères étudiés ainsi que les scénarios définis pour chaque critères sont les suivants :

1. **Taille de la zone.** Taille de la zone d'intérêt.
2. **Forme de la zone.** Forme de la zone d'intérêt.
3. **Météo.** Type de météo sur une zone d'intérêt donnée.
4. **Nombre de satellites.** Nombre de satellites disponibles pour la grande couverture.

Pour tous les scénarios décrits, les paramètres globaux suivants sont appliqués :

1. Taille des cellules : 20x20km,
2. La taille des mailles varie entre 20x40 km et 60x120 km en fonction du cas d'utilisation,
3. Satellites hétérogènes (toutes les fauchées différentes),
4. Longueur maximale des mailles fixe,
5. Délai de 24 heures entre l'acquisition des satellites et l'acceptation ou le rejet des images.

Les résultats du critère de la taille de la zone valident l'adaptation de Glimpse. Le temps d'exécution et le gâchis sont considérablement réduits dans le cas de très grandes zones qui sont plus susceptibles d'être au centre des demandes des clients. Le critère de la forme de la zone permet de vérifier la capacité d'auto-adaptation des agents. En exploitant les avantages du maillage dynamique, les agents ont pu placer les mailles de manière appropriée pour s'adapter aux limites du couloir et à la forme de la zone.

L'anticipation des conditions météorologiques a également été confirmée par le critère correspondant. Glimpse a réussi à gérer la dynamique et les éventuelles modifications météorologiques au cours de la planification. Enfin, le critère du nombre de satellites a révélé que Glimpse peut servir d'outil aux opérateurs pour choisir les meilleures constellations satellitaires. La réduction du gâchis par rapport à l'algorithme glouton a été observée quel que soit le nombre de satellites, la différence étant plus importante lorsque le nombre de satellites est faible.

La stabilité de ces observations valide la robustesse de Glimpse pour gérer une grande variété de cas que l'on peut rencontrer dans le domaine des grandes couvertures. Glimpse

montre de meilleures performances pour les zones d'intérêt nuageuses de forme irrégulière, ce qui en fait un bon algorithme pour la résolution des cas les plus complexes du problème des grandes couvertures.

1.4.4 Analyse de sensibilité

La criticité des agents Cellule dans Glimpse peut être modifiée en changeant à la fois les critères qui la composent et l'ordre dans lequel ils apparaissent. Comme les critères sont traités dans un ordre lexicographique, les critères venant en premier ont plus d'impact car toute comparaison décisive coupe la comparaison des critères suivants dans l'ordre.

Dans cette section, un **variant** de criticité représente une liste ordonnée de critères de criticité de l'agent cellule. Un **variant** est nommé en utilisant les différents critères qui le composent dans l'ordre lexicographique. Cette section propose une étude de sensibilité des critères de criticité des agents Cellule de Glimpse en comparant les résultats obtenus suivant le variant de criticité utilisé. Les métriques considérées sont le temps de complétion et le gâchis. Un algorithme glouton hiérarchique avec maillage statique est utilisé comme algorithme de référence.

40 variants différents sont identifiés avec une liste de critères de criticité d'une taille allant de 1 à 4 critères. Des résultats de temps de complétion et gâchis pour chaque critère sont obtenus sur un ensemble de 10 scénarios. Ces scénarios varient de par leur zone d'intérêt, le nombre de satellite utilisés, le type de satellites (hétérogènes ou homogènes) ou encore la date choisie qui influence la météo observée. Les résultats obtenus sont moyennés sur une série d'expérimentations afin de réduire la variance dûe aux effets aléatoires tels que les différences entre météo prévue et observée.

Parmi les variants sélectionnés un d'entre eux obtient des résultats favorables à la fois en terme de temps de complétion et gâchis par rapport aux autre variants. Ce variant est composé, dans l'ordre, d'un critère de validation des cellules, d'un critère évaluant la météo du passage courant, d'un critère sur le nombre d'opportunité d'acquisition restante et enfin d'un critère sur la météo des les passages futurs sur l'horizon de planification. Ce variant est proche de celui choisi pour effectuer les expérimentations de performances et de robustesse, et qui avait été choisi en suivant les avis d'experts du domaine.

1.4.5 Discussion

Les résultats obtenus montrent les avantages du maillage dynamique par Glimpse par rapport à un maillage statique traditionnel employé par un algorithme glouton hiérarchique.

Les scénarios avec satellites homogènes montrent une amélioration en faveur de Glimspe pour la métrique de gâchis de ressources satellitaires. Dans le cas des satellites hétérogènes, des améliorations en terme de temps de complétion sont également constatées. Ces observations s'expliquent par de meilleurs placements des mailles par maillage dynamique. En effet les zones déjà validées, les chevauchements de mailles inter-passages et les zones de faible criticité sont évitées grâce au placement plus précis et adaptatif des mailles de Glimpse.

L'étude de la robustesse de Glimpse confirme ces observations. Les résultats montrent que les performances de Glimpse sont notamment supérieures à celles du glouton hiérarchique dans les cas difficiles du problème des grandes couvertures. Les grandes zones, les zones nuageuses et les zones de forme particulière sont mieux traitées par Glimpse, ce qui valide la robustesse du système.

Enfin l'analyse de sensibilité des critères de criticité des agents Cellule de Glimpse montre l'adéquation du critère utilisé sur les expériences de performances et de robustesse. Les résultats soulignent la possibilité de sélectionner un variant de Glimpse spécifique pour différents type de scénarios de grandes couvertures.

1.5 Conclusion

1.5.1 Synthèse

Le problème LAC est un problème d'optimisation de l'utilisation de satellites pour la réalisation efficace d'acquisitions de zones à la surface de la Terre. Dans ce problème, une entité dite LARM reçoit une demande de client pour l'acquisition d'une grande zone d'intérêt. Le LARM envoie des demandes d'acquisition aux centres de missions ou MPFs lors des passages successifs du satellite. Les demandes d'acquisition peuvent échouer en raison de différences entre les prévisions météorologiques et les conditions météorologiques observées. Une fois des images acquises et validées par le MPF, le LARM accumule ces images jusqu'à pouvoir renvoyer au client la collection d'images acquises, soit le produit de la mission LAC.

Dans cette thèse, nous nous concentrons sur la résolution du problème LAC à l'aide de systèmes multi-agents adaptatifs. Ces systèmes sont composés d'agents, entités élémentaires qui s'auto-organisent pour atteindre des objectifs locaux tout en respectant des règles de coopération. Nous proposons d'utiliser les systèmes multi-agents pour résoudre le problème LAC avec une technique de maillage dynamique. Cette technique introduit des cellules, des sous-zones élémentaires qui permettent un placement plus précis du maillage et une adaptation aux caractéristiques de la mission. Le système multi-agent Glimpse exploite ce maillage dynamique pour résoudre le problème LAC. Les trois types d'agents du système Glimpse, Cellule, Maillage et Passage, sont décrits. Leurs interactions coopératives permettent de faire émerger une fonction d'optimisation du maillage au cours de la résolution d'une mission LAC.

L'évaluation de Glimpse est composée de trois axes. Dans ces trois axes un algorithme de type glouton hiérarchique utilisant le maillage statique est utilisé en tant que LARM pouvant être rencontré en contexte opérationnel. Les résultats concernant les performances de Glimpse démontrent sa capacité de placement des mailles sous contrainte et particulièrement dans le cas de satellites hétérogènes. Des exéprimentations supplémentaires de robustesse montrent la capacité de Glimpse à mieux traiter de grandes zones d'intérêt avec de mauvaises météos, soit les cas difficiles du problème LAC. Enfin une étude de sensibilité révèle la possibilité de déterminer des variants de Glimpse suivant les caractéristiques de la mission.

Ces résultats mettent en avant l'adéquation des choix effectués dans la production des

contributions scientifiques et opérationnelles de cette thèse. L'utilisation de systèmes multi-agents représente une contribution dans le domaine de l'optimisation des missions satellitaires. L'analyse des relations entre critères de criticité d'agents coopératifs et résultats participe à la recherche et au développement des systèmes multi-agents. Côté industriel, Glimpse est un potentiel outil d'optimisation de missions LAC et d'estimation de la taille de constellation à employer pour ces missions. Enfin une proposition est faite d'un type de maillage innovant, le maillage dynamique, permettant un placement plus fin des mailles et une adaptation aux contraintes en cours de résolution.

1.5.2 Perspectives

Cette thèse ouvre plusieurs axes de recherche possibles. Une étude comparant les résultats de Glimpse et d'autres algorithmes candidats mettrait davantage en évidence les forces et les faiblesses de Glimpse et de la théorie AMAS. Ces résultats pourraient être étendus pour considérer d'autres types de problèmes d'optimisation similaires au problème LAC. Le sujet de l'amélioration des résultats de Glimpse pourrait émerger de ces études. Une possibilité réside dans l'emploi de techniques d'apprentissage par renforcement pour une meilleure détermination des variants de Glimpse suivant les missions LAC.

Le problème LAC lui-même pourrait également être étudié avec une représentation plus complète. Une technique de découpage d'image par les MPFs, la validation partielle pourrait notamment être utilisée. Le maillage dynamique semble particulièrement adapté à cette technique avec laquelle les cellules sont validées individuellement au lieu d'être groupée par maillage. Des caractéristiques des satellites considérées en opérationnel pourraient également être ajoutées. Une représentation des manoeuvres des satellites, de la consommation d'énergie à bord ou encore d'espace de stockage utilisé par les images acquises sont autant de pistes améliorant la représentation du problème, et autant de contraintes devant potentiellement être considérées dans la conception du comportement des agents de Glimpse.

2 Programming of Earth Observation Missions

2.1 Introduction

In this thesis we investigate the *Large Area Coverage (LAC)* problem. This problem is one of the different steps of the Earth observation problem. In this chapter, we give a brief introduction to the programming of Earth observation missions by first defining the different elements of Earth observation, then we introduce the full process of Earth observation missions and present the important terms to define the *LAC* optimization problem.

2.2 Earth Observation Components

The Earth observation domain encompasses different kind of fields such as on-site photography, aerial acquisition from drones or airplanes, and acquisition from specialized satellites. In this thesis, we focus on Earth observation missions performed by a constellation of **Earth observation satellites**.

A satellite is defined as an artificial object orbiting around the Earth. Satellites can be differentiated by their altitude and orbit. First, **geostationary satellites** operate at a height of approximately 36.000 kilometers. Their orbit maintains them at a fixed point above the Earth and they can be used for telecommunication and weather tracking for example. **Earth observation satellites operate at *Low Earth Orbit (LEO)***, often around 400 kilometers. Their speed is approximately 27.000 km/h or 7.5 km/s, allowing them to complete a full rotation of the Earth in less than 2 hours. Earth observation satellites orbit around a north/south pole axis, meaning that after a full rotation around the Earth a new Earth surface can be acquired as Earth rotates around itself. The low orbits improve the precision of acquired images, with higher resolution and better quality. Different types of satellites including *LEO* satellites are presented in figure 2.1.

The mission applications for Earth observation satellites range from oceanography to forest control, mapping, disaster control and more. Depending on the need, the satellites can use either optical or radar acquisition technologies. In this thesis, we focus on optical satellites and their applications. These satellites have an **optical sensor** pointing towards the Earth that covers a width or **swath** and is able to acquire Earth surfaces. The sensor is

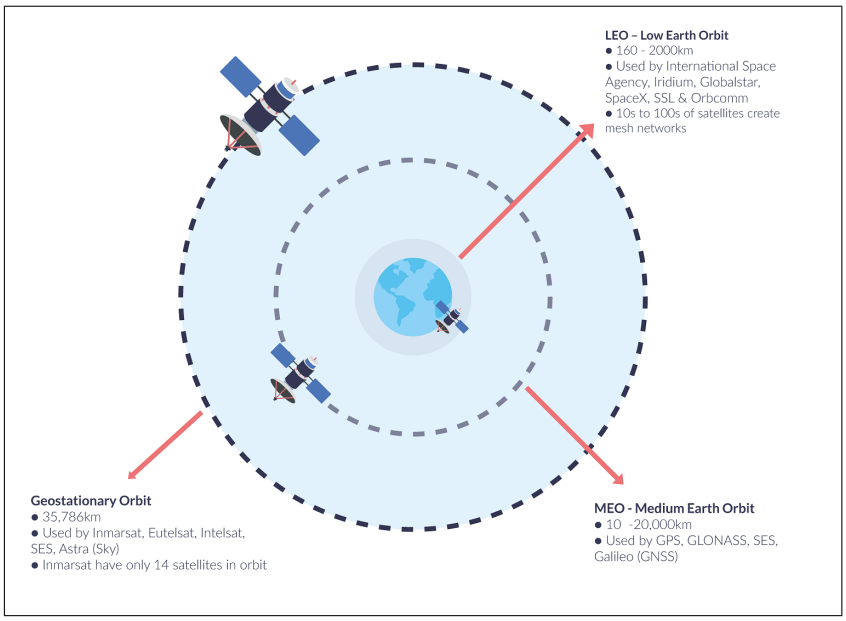

Figure 2.1: Types of satellites by type and operating height

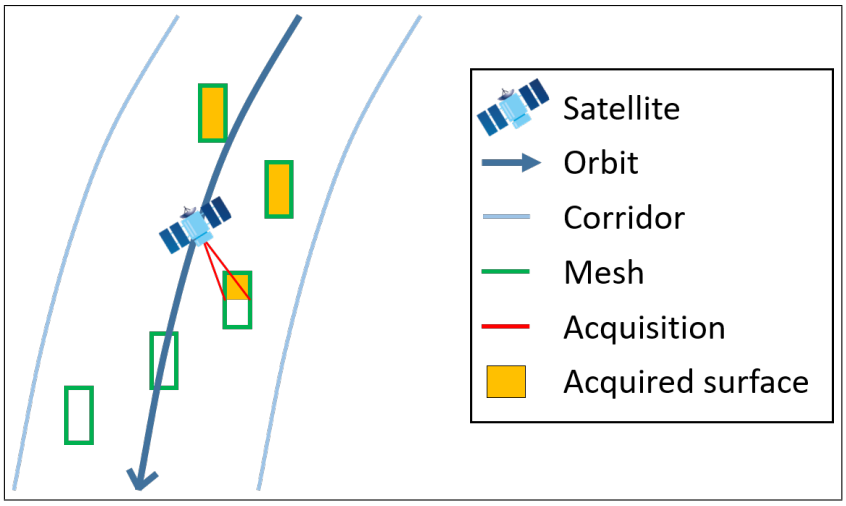

Figure 2.2: Schematic elements of the acquisition of meshes by an *LEO* satellite

switched on at the start of an acquisition and turned off after a few seconds. The resulting image is obtained from a **mesh**, a rectangle of the Earth's surface of width relative to the sensor swath and of length proportional to the elapsed time during which the optical sensor was switched on. A mesh is a geometrical representation of a possible **acquisition** by a satellite. A mesh can partially or fully cover the ***Area Of Interest (AOI)***, the zone that needs to be acquired completely to complete the observation mission. Figure 2.2 illustrates the schematic representation of elements including the acquisition of several meshes by an *LEO* observation satellite.

Acquisitions were originally only performed directly under the satellite at a point called **nadir**. To improve the flexibility of the acquisition process *LEO* observation satellites have since been modified to be **agile**, meaning they can rotate on 3 different axis. These rotations

of pitch, yaw and roll are presented in figure 2.3.

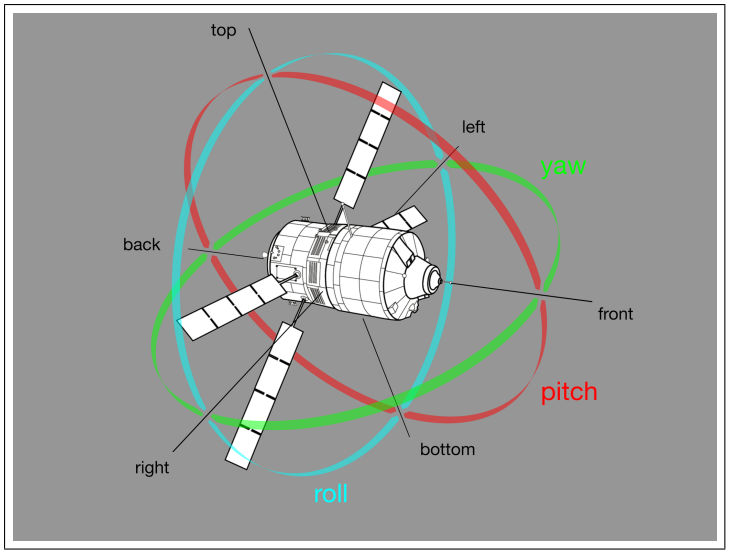

Figure 2.3: Angles of rotation of agile satellites

The goal is for the satellite to acquire meshes that are away from nadir relative to its trajectory. Agile satellites perform orientation maneuvers to place themselves towards specific meshes, a process usually taking a few seconds to complete. Acquiring meshes that are off nadir lowers the resulting image resolution but greatly reduce the time required to perform the acquisitions compared to the time imposed by non-agile satellites having to be on an orbit directly above the mesh.

Indeed over a given orbit agile satellites are able to cover a larger surface of the Earth during a **pass**. A pass is defined as the time lapse during which a satellite flies over a zone called **corridor** of the pass. A corridor intersects the *AOI* and defines the limit where meshes can be acquired relative to the maximum inclination of the satellite using its agile rotational properties. The difference in acquisition capabilities between acquisitions at nadir for non-agile satellites and acquisitions in a corridor for agile satellites is illustrated in figure 2.4. Note that, by increasing the covered surface, the search space including the meshes to be acquired by satellites is also increased. While orbiting around the Earth, satellites communicate with **ground stations**.

A ground station communicates with the satellite by radio or light messages, either to send the next instructions in a mission plan to the satellite, a process called **uplink**, or to receive acquired images from the satellite, a process called **download**. This uplink-acquisition-download process is illustrated in figure 2.5

Commercial Earth observation satellites are often operated by companies lending their services to other companies or individuals, allowing clients to request the acquisition of a specific zone. An organism responsible for client request treatments is called a ***Mission Planning Facility (MPF)***[Nejad, 2022]. The task of *MPFs* can be summarized as the assignment of acquisitions to different satellites such as to maximize the use of satellites resources and minimize the time between request reception and treatment while considering the request importance. Indeed, **requests can vary in client time requirements**. An acquisition

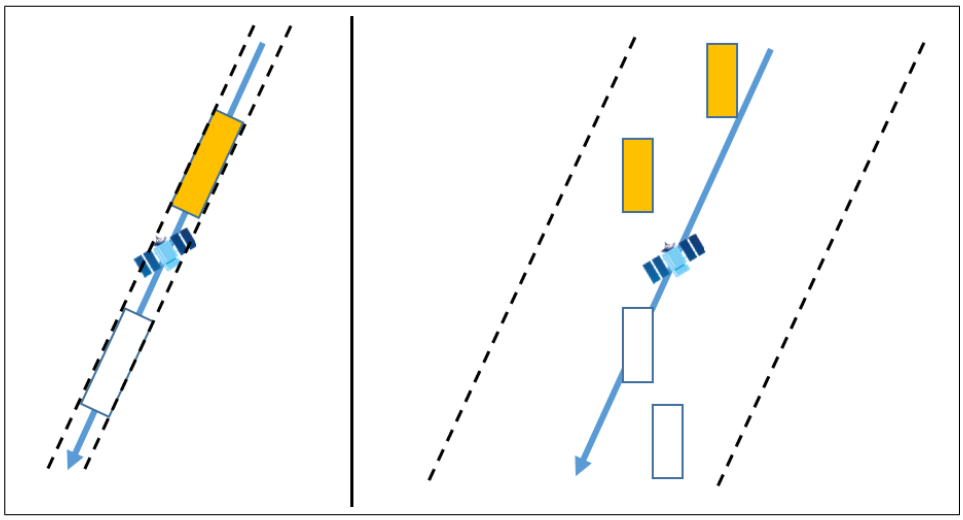

Figure 2.4: Comparison of acquisition abilities between non-agile (left) and agile (right) satellites

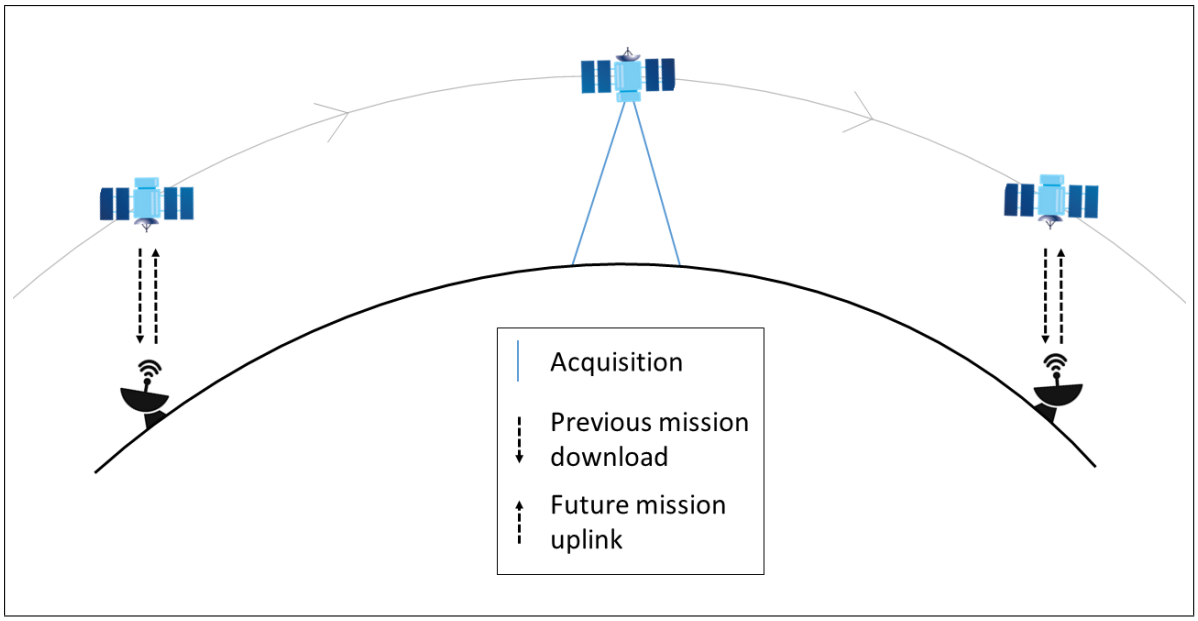

Figure 2.5: Satellite cycle of an *LEO* observation satellite consisting of uplink, acquisition and download

request relative to a forest fire is usually more urgent than a routine task, for example the acquisition of an image for a longer scientific study. Such concepts and their impact on client request treatment are detailed in section 2.3.

2.3 Earth Observation Steps

This section presents step-by-step the acquisition process followed by an *MPF*, from the reception of an acquisition request to image validation by the client. Figure 2.6 presents the full process with the following steps:

1. **Acquisition request.** The acquisition request received by an *MPF* specifies different

Figure 2.6: Illustration of an observation request by a client directed to an *MPF*

criteria guiding the acquisition. First the ***AOI*** for acquisition, defines a zone that can be covered by either a single mesh or a few meshes depending on the optical sensors characteristics of the satellites operated by the *MPF*. Then the **time** before request obsolescence, a period varying from a few hours to several months depending on the client need is defined. A desired minimum **resolution** can also be specified. Finally different kind of contracts define a **cost** relative to the desired request importance.

Based on the listed criteria, the urgency of the acquisition request is established. A hierarchy of requests is maintained, with tiers that can be simplified as **urgent**, **normal** and **routine**. This hierarchy impacts the treatment of requests, as presented in the next steps.

2. **Meshing and planning.** In this step, the requested zone is split in one or several meshes. Each mesh is associated to a satellite, with the characteristics of the optical sensor dictating the swath and maximum length of the mesh. The meshes cover the whole zone but may be acquired at different times and the resulting images may require reconstitution later in the acquisition process. The most common way to split the zone into meshes is to divide the zone using a precomputed grid of meshes and later selecting successive mesh locations as meshes to acquire. We call this process **static meshing**. An example of a static meshing grid over France is presented in figure 2.7. We describe and compare static meshing to a novel way of splitting the zone called **dynamic meshing** in chapter 5.

   For each satellite, an acquisition plan must be computed. This plan is composed of sequential instructions that the satellite follows in chronological order. Instructions include acquisitions, orientation maneuvers or sending back acquired image to a ground station using the **download** process. Current acquisition satellites cannot acquire two meshes at the same time, a strict order of acquisitions and maneuvers respecting the satellites characteristics must be defined. While respecting material constraints an *MPF* tries to maximize the acquisition of important meshes defined by the hierarchy built in step 1.

   An *MPF* may operate a constellation of satellites and build plans to avoid acquisitions of the same *AOI* by different satellites. As meshes are acquired, plans are continually updated on ground and sent to the satellites when their orbit intersects with the visibility of a station able to communicate with them. Another constraint is introduced at this step: planning around weather issues. **Half of the images acquired operationally are deemed unusable due to clouds being apparent**. Indeed, given the position of ground

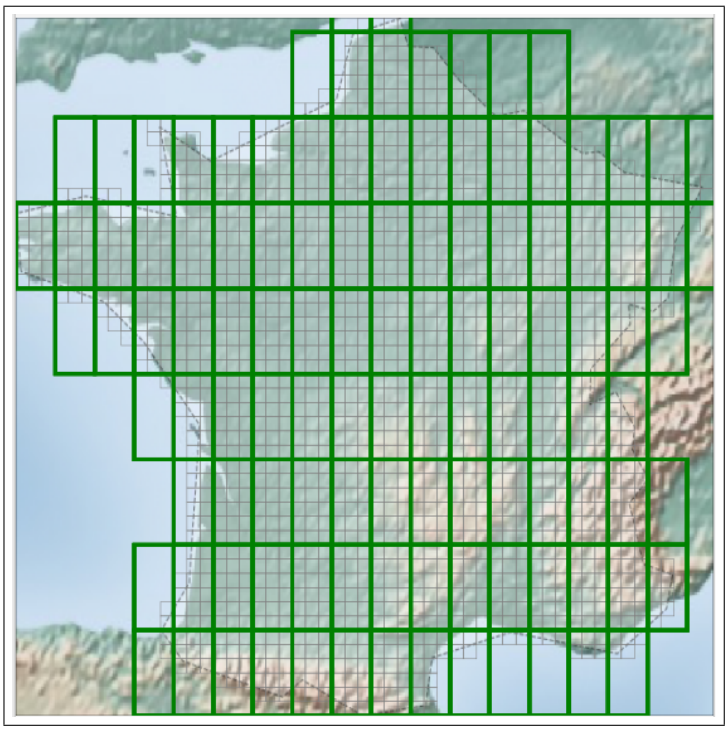

Figure 2.7: Example of a static meshing grid for a single satellite with France as *AOI*

station able to communicate with the satellites and the time required to compute and validate an acquisition plans, plans are computed ahead using uncertain weather conditions. Thus, a large number of images is rejected due to cloudiness.

3. **Acquisition.** The satellite receives the acquisition plan and performs the acquisitions and maneuvers during the set time frame. As described in step 2, the acquisition step may happen several hours after the plan is sent from an emission station depending on the distance between the station emitting the plan to the satellite and the *AOI*. Due to this gap **the weather forecast at the time of plan emission may be inaccurate during acquisition** and captured images may be clouded. The satellite then sends the acquired images back to a ground reception station.

4. **Validation or rejection.** At this step received images of the meshes may need to be reconstructed to form a single or several images expected by the client. Cloudiness is evaluated and the images are sent to the client. If the images are evaluated as satisfactory by the client, the acquisition process ends. Otherwise the images are rejected and the process starts again beginning from step 2.

5. **Request completion.** Obtained images that have been validated by the *MPF* as conform to the client prerequisites are the final product of the process. After confirmation with the client the acquisition request is completed.

A single *MPF* treating an acquisition request is usually sufficient but as we present in section 2.4 some large scale requests called *LAC* requests may require the involvement of several *MPFs*.

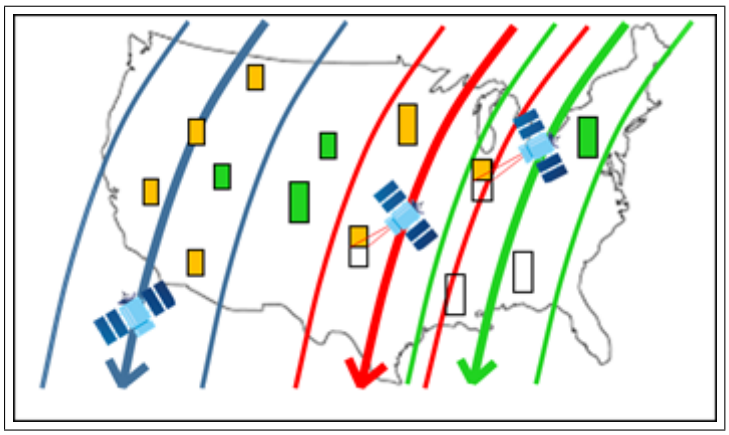

Figure 2.8: Representation of multiple satellites acquiring over the same *AOI*

2.4 The *LAC* Problem

2.4.1 Context

Planning for satellite missions is a broad subject of research and development. As presented in chapter 2, many different sources treat the optimization problem summarized in step 2 of section 2.3. In the following, we refer to this problem as the daily single satellite scheduling problem. In this thesis, we focus on another specific type of client request aiming on the acquisition of **large areas** (i.e. country or continental scale). The acquisition of such zones could be performed by a single *MPF* but due to the low number of available satellites, the resulting images would be acquired after several orbits and the *AOI* coverage span over a very large time period. This would lead to disparities between images due to varying weather conditions or seasons. To solve this issue different *MPFs* and thus operated satellites are used as illustrated in figure 2.8.

In the context of large scale requests, the clients only specifies the *AOI* to acquire, a cloudiness rate threshold ad a deadline to latest completion. The acquisition coverage should then be performed as quickly as possible. To minimize the time required to fully acquire the requested zone, **we consider the large area request treatment to be performed by a single super *MPF*** dispatching client requests to a subset of satellites operated by different *MPFs*. This super *MPF* called a ***Large Area Request Manager (LARM)*** is trying to solve the *LAC* problem. The structure between *LARM*, *MPFs* and their operated satellites is described in figure 2.9.

2.4.2 Acquisition Process

The steps followed traditionally by the *LARM* to complete the *LAC* mission are similar to those followed by a traditional *MPF* described in section 2.3:

1. **Request reception.** The *LARM* receives the *LAC* client request. Its components include the *AOI*, the list of *MPFs*, their available satellites and the lists of passes performed by all the satellites over the *AOI* over a large time period.

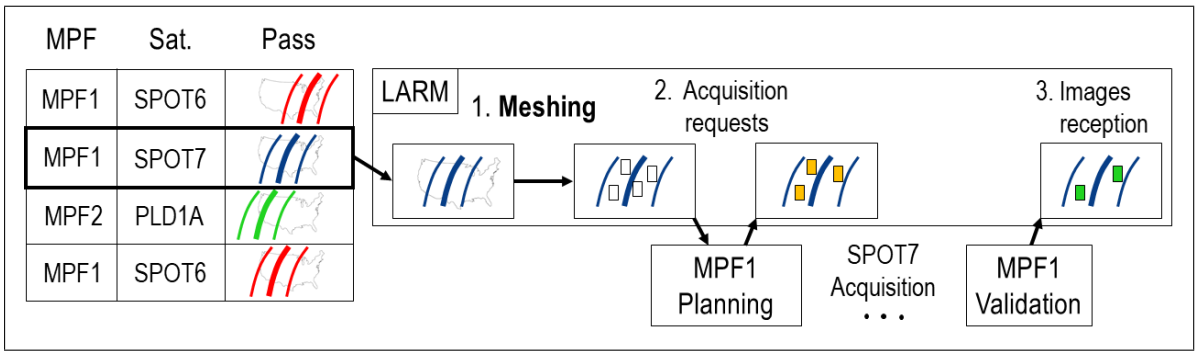

Figure 2.9: Process between *LARM* and *MPFs* for the acquisition of meshes on a satellite pass

2. **Meshing.** The *LARM* divides the *AOI* into meshes to acquire. The resulting mesh grid covers the whole area so that each mesh needs to be acquired at least once to complete the *LAC* mission.

After these steps are performed the *LARM* follows the list of satellite passes in chronological order and performs the following steps for each pass:

1. **Mesh selection.** The list of acquirable meshes is built from the corridor of the pass. An acquirable mesh must be fully included in the corridor. A subset of meshes is then built from acquirable meshes based on the likelihood of being successfully acquired. The selected meshes are sent to the *MPF* operating the satellite performing this pass as acquisition requests.

2. **Validation.** The *MPF* either denies or accepts each acquisition request and the satellite perform acquisition and download for accepted acquisition requests. Images from these requests are sent to the *LARM* for cloud coverage estimation. Images with a cloud coverage rate above the client approved threshold are discarded and corresponding meshes need to be acquired at a later pass. Images with a cloud coverage under the threshold are validated and saved as part of the *LAC* request answer.

For each satellite pass over the *AOI* in phase 3 the *LARM* builds an acquisition plan. However the *LARM* does not operate the satellites and needs to send acquisition requests for each mesh to the *MPF* operating the satellites. A mesh for the *LARM* is then considered an *AOI* by the *MPF* receiving the relative acquisition request. The *MPF* then follows its own acquisition process and either sends the resulting images or responds negatively if the satellite could not acquire the mesh during the pass due to higher priority acquisitions being performed.

2.4.3 Waste

Satellite resource **waste** is the under-utilization of satellite time by performing useless tasks. In the context of *LAC*, waste can either be caused by:

- differences between weather forecast and observed weather causing acquisitions of cloudy areas,

- ▷ acquisition of meshes which cover at least partly surfaces which have already been acquired and validated.

Weather forecast issues are hard to solve in the context of *LAC* missions. The only leverage is to select meshes with high weather forecast confidence for good weather. Better weather forecasts and shorter gaps between plan transmission and acquisition help with minimizing waste from this source but technical improvements to the parameters of the problem are considered beyond the scope of this thesis.

Acquiring the same surfaces several times is a problem encountered due to the use of satellites operated by different *MPFs* and a justification for the *LARM*. **Random acquisitions by independent non-communicating *MPFs* would frequently produce overlaps of the acquired surfaces**. Even with the coordination of an *LARM* re-acquisitions can happen. Satellite with different on-board acquisition tools and characteristics may cause overlaps between meshes and waste satellites resources. This is described further in section 3.4.1. Requesting the acquisition of the same surface by different satellites may also be necessary in order to obtain the corresponding image quickly and finish the *LAC* mission in a shorter time frame.

Minimizing waste is desirable as each discarded image is non-profitable for the *MPF* operating the satellite and by extension the requesting *LARM*. **Waste may also indirectly extend the time to completion of the *LAC* mission** as discarded images represent missed acquisition opportunities on satellites passes.

2.4.4 Conclusion

In this chapter the context of Earth observation missions performed by *LEO* satellites is presented. Individual satellites follow a scheduling pattern from client request given to *MPFs* to image acquisition and validation. Requests regarding large areas require the use of several *MPFs* and satellites that are coordinated by an *LARM*. For each satellites pass over the *AOI*, a small number of meshes need to be selected for acquisition from a large pool of candidate meshes. The resulting *LAC* problem is a difficult multi-objective optimization problem which has seen growing interest in the Earth observation literature. In chapter 3 we present a formalization of the *LAC* problem and its solving metrics. We then propose a state of the art of approaches for solving the *LAC* problem encountered in an operational context and in the research literature.

3 State of the Art

3.1 Introduction

The relevance of the *Large Area Coverage (LAC)* problem is proportionally growing with the number of *Low Earth Orbit (LEO)* observation satellites available. As many different satellites can be matched to perform a single *LAC* task, the difficulty of distributing the acquisitions to each satellite grows.

As seen in section 2.4, the *LAC* problem is considered a multi-objective optimisation problem. This chapter first starts by defining the metrics used to evaluate the solving approaches and the obtained solutions. Then, it presents the operational method used by *Mission Planning Facility (MPF)* centers. Afterwards, this chapter focuses on different theoretical approaches explored to solve the studied problem, analyses their advantages and limitations and discusses the adequacy of multi-agent systems to solve this type of problems.

3.2 Problem Formalization

We define the *LAC* problem as the coverage of the requested *Area Of Interest (AOI)* in a minimal time while minimizing waste. The components of the *LAC* problem can be formalized as follows:

- a *Large Area Request Manager (LARM)* L receiving a large area coverage acquisition request of an *AOI* noted Z,
- a set of *MPFs* $F = f_1, \ldots, f_i, \ldots, f_I$ to which the LARM can send acquisition requests,
- for each *MPF* f_i a set of operated satellites $S = s_{i1}, \ldots, s_{ij}, \ldots, s_{iJ}$,
- for each satellite s_{ij} a set of passes $P = p_{ij1}, \ldots, p_{ijk}, \ldots, p_{ijK}$ over the *AOI* with the first and last available time of acquisition for pass p_{ijk} respectively $t_{start_{ijk}}$ and $t_{end_{ijk}}$ and a corridor $c_{ijk} \in Z$.

The problem consists in finding for each satellite pass p_{ijk} a set of N requested meshes to acquire $M = M_{ijk1}, \ldots, M_{ijkn}, \ldots, M_{ijkN}$ with $M_{ijkn} \in c_{ijk}$ which minimizes time to completion and waste.

3.3 Evaluation Metrics

3.3.1 Problem Metrics

As described previously the *LAC* problem implies the use of several *LEO* satellites during the course of weeks or months. Indeed the *LAC* client request is usually treated as a background task and often relegated to later satellite passes because of more urgent requests. However **the time taken to complete the *LAC* mission should be minimized** to fulfill the client's request at the earliest. Completing parts of the acquisition process early also allows the *LARM* to send partial updates to the client. **The time to completion is the first metric considered** and its minimization is an objective of the LAC problem.

Due to the length and high-scale profile of *LAC* missions the use of satellite resources is another important factor. *LEO* satellites have a lifespan of approximately 10 years in orbit as the atmospheric drag deteriorates them over time. Because of this lifespan and the high cost of building and launching the satellites, each acquisition should be optimized to maximize its likelihood of resulting in a clear image and to acquire as much non-acquired *AOI* surface as possible. The **waste** of a *LAC* mission defines **the rate of wasted satellite resources** over the whole acquisition period. Acquisitions resulting in useless clouded image and surfaces of the *AOI* being acquired several times over different satellite passes increase the waste of the mission. **The waste is the second metric considered** and its minimization is an objective of the LAC problem.

On the one hand the optimization of the time to completion metric benefits the client emitting a *LAC* request to the *LARM*. On the other hand reducing waste, benefits the *LARM* as the number of requests sent to each *MPF* during the mission is reduced proportionally to how well satellite resources are used. Both objectives should be optimized during a *LAC* mission but due to this difference, the time to completion is considered to be of a higher priority than waste. **The *LAC* problem is a multi-objective optimization problem** with a strict order of importance affected to these two objectives. In the following, when comparing results of a *LAC* mission simulation, the following hierarchy of criteria in descending order of importance is used to evaluate the quality of each solution:

1. **minimization of time to completion**,
2. **minimization of acquisition waste**.

3.3.2 Solving Metrics

Problem metrics define the quality of a solution for an example of the *LAC* problem. To evaluate the performance of the algorithm that led to a solution we consider other metrics linked to the quality of the problem solving itself. Criteria in this case are dependant on the type of algorithm used to find the solution. Their relative importance is not discussed here as criteria may be ranked differently depending on the need of the *LARM* and the type of *LAC* request. In no particular order the solving metrics we identified include:

- **Calculation time**, the time required for the algorithm to find a solution in seconds. This

metric is calculated from the launch of the algorithm at *LAC* request reception and up to the validation of the last mesh finishing the acquisition process of the whole *AOI*, marking mission completion.

- **Algorithm robustness**, the ability of the algorithm to demonstrate good performances for different types of *LAC* request. Differences in requests could include discrepancies in weather, *AOI* shapes, amount of satellites, mesh sizes, etc...
- **Algorithm scalability**, the ability of the algorithm to find solutions of consistent quality for ever increasing sizes of *AOI*. Solutions should also be found in reasonable calculation time as a plan obtained with a very high calculation time will be ready too late for it to be transferred to the satellite.
- **Local minima sensitivity**, the ability of the algorithm to avoid being trapped in local minima and explore the search space for improved solutions.

Both problem and solving metrics are used in the next sections to evaluate different approaches to the *LAC* problem. Later in chapter 4 these metrics are used to discuss the merits of using an *Adaptive Multi-Agent System (AMAS)* to solve the *LAC* problem with a focus on the robustness of the system to different experimental conditions.

3.4 State of the Art of Solving Approaches

3.4.1 Operational Technique

The decisions behind satellite affectations and acquisitions in the industry have historically been made by experts in this domain. The **estimation of acquisition success** for a given *AOI* and acquisition time is **performed by an expert** using recently measured weather data and knowledge of the region's climate. Similarly this expertise is used in solving the *LAC* problem operationally. The team supervising this study includes domain experts which shared information regarding common operational practice. The technique used **splits the *LAC* request into several sub-areas** that can be acquired by a single *MPF* and affects the adequate *MPF* to each obtained sub-area. The following process is applied:

1. division of the area of interest into several sub-areas as presented in figure 3.1,
2. exclusive distribution of sub-areas among *MPFs*
3. acquisition by the *MPFs* of their respective sub-areas,
4. merge of the acquired sub-area images in a single request answer.

This process simplifies the *LAC* problem by reducing the waste and avoiding the problem complexity produced by coordinating several satellites each controlled by different *MPFs*. Indeed with each *MPF* treating the coverage of its sub-area the characteristics of its satellites can be similar. Similar satellites can be used with precomputed grid of mesh

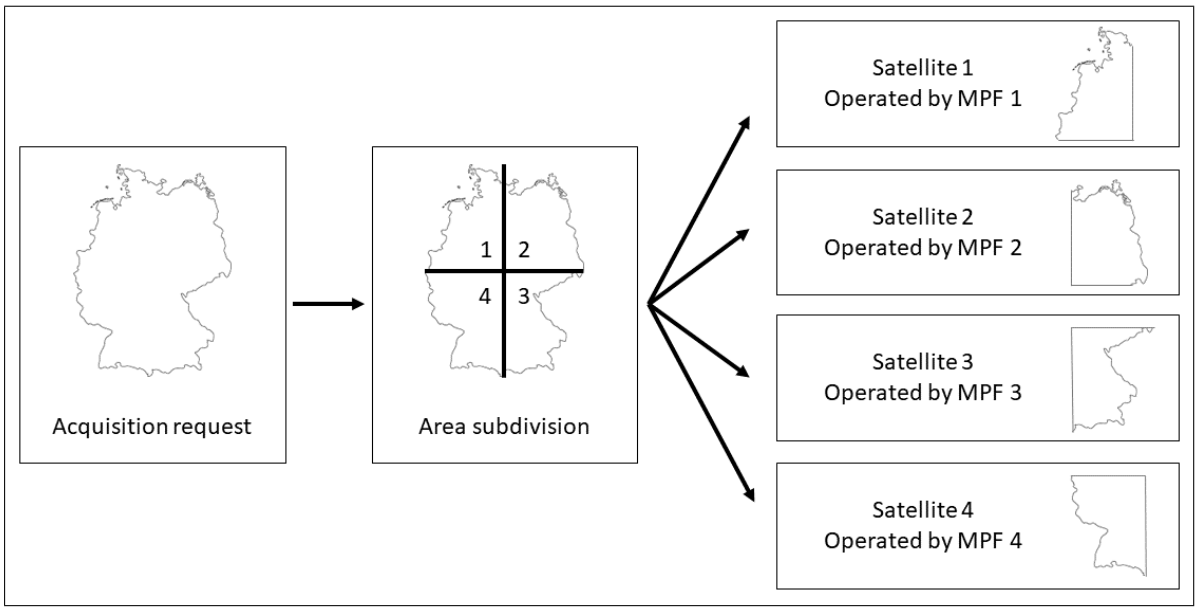

Figure 3.1: Operational division of a requested *AOI* and distribution to non-conflicting satellites

placements to acquire over the course of the *LAC* mission following a technique called **static meshing**.

Static meshing efficiently uses meshes of equal dimensions so that their acquisitions do not overlap which reduces waste due to re-acquisition. Using separated sub-areas also avoids the *LAC* problem complexity caused by the large pool of satellites to coordinate during the mission. With a reduced number of satellites for each sub-area and a single *MPF* to manage them no *LARM* is required. A simple merging treatment is required when each *MPF* has finished its coverage mission.

As a simple solution to the *LAC* problem, this approach has drawbacks when considering the time to completion metric. **The distribution of satellites in sub-areas restricts their use** in the parts of their passes that overlap other sub-areas of the *AOI*. An example of this restriction is presented in 3.1. Each pass restricted in this way may be underused in number of meshes acquired or be used to acquire meshes with lower probability of clear weather than what an unrestricted pass of the same satellite would allow. In other terms **the reduction in the search tree of the *LAC* problem cuts potentially optimal nodes** for every satellite path. We can conclude that acquisition opportunities lost due to these artificial restrictions increase the overall time needed to acquire the *AOI*.

Additional constraints imposed on the problem such as the use of strips (presented in section 3.4.2) instead of meshes may be used operationally to further simplify the *LAC* problem with static meshing. Lifting these restrictions increases the complexity of the *LAC* problem by introducing the need to coordinate satellites that may present different characteristics and to accurately select meshes in the corridors but it allows the full use of each satellite pass to better optimize the time to completion. **In the following we consider the *LAC* problem without subdivision** formalized in section 3.2 to lead to better overall completion time and waste than those obtained with the operational technique presented in this section.

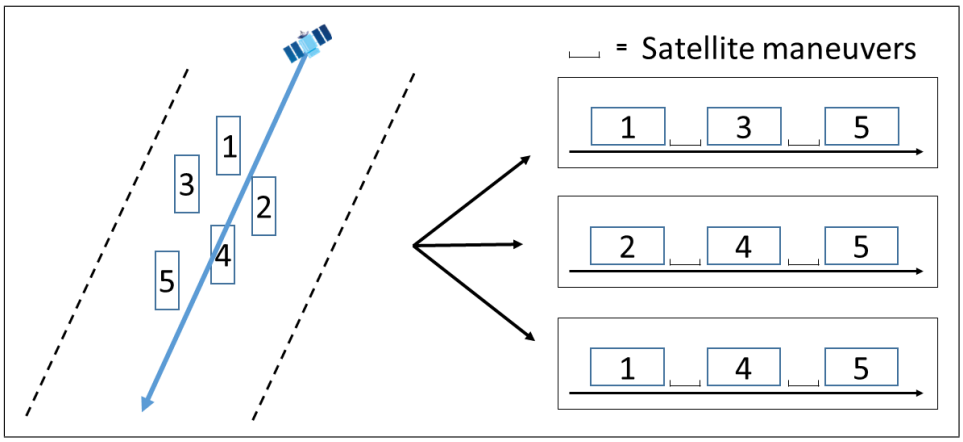

Figure 3.2: Single satellite acquisition problem with possible solutions

3.4.2 Approaches to the *LAC* Problem

The optimization of *LEO* satellites observation missions is a wide domain that encompasses different problems.

Single Satellite Problem. A common example is the optimization by an *MPF* of an acquisition plan for a single satellite over a set time frame or until a planning horizon. In this context we consider a single *MPF* and a set of small scales acquisitions requests also called **spots**. Each spot request can be covered by a single mesh of a satellite operated by the *MPF*. A number of instructions must be sent to the satellite in advance of a time frame called **planning horizon**. The planning horizon is at minimum the time period between two windows of transmission from a ground station to the satellite. Indeed no other communications can be made to the satellite and the satellite executes its acquisition plan without additional input possible in this time frame as seen in figure 2.2. For each request a mesh that includes the *AOI* is computed and represents a possible acquisition. **Acquisition planning for a planning horizon of a single satellite consists in selecting different meshes and building a sequence of acquisitions and orientation maneuvers**.

An example of a mission planning problem for a single satellite by a *MPF* is depicted in figure 3.2. To select a sequence of meshes to acquire different metrics are used. These metrics include the importance of the corresponding request that is rated relative to other requests waiting to be completed, the time left before the corresponding request expires and the weather forecast at the *AOI* at the predicted time of acquisition.

As is the case in the *LAC* problem for several heterogeneous satellites, the acquisition plan of a single satellite can be modified to optimize the metrics mentioned above. This optimization problem is multi-objective as both the importance of fulfilled requests and the overall number of fulfilled requests must be maximized. This is a known problem in the literature and has been characterized by several sources including [Bensana *et al.*, 1999]. The authors formalize the problem as a set of constraints to satisfy using a subset of possible acquisitions and maneuvers from the search space that maximizes the weight or importance of acquired images. The problem is judged similar to a **Multi-Knapsack** problem and able

to be represented by a *Constraint Satisfaction Problem (CSP)* and solved by an *Integer Linear Programming (ILP)* solver like **CPLEX**.

[Frank *et al.*, 2001] expands on the realism of the problem by including on-board constraints. An interval based model is built to represent the problem and an heuristic biased stochastic greedy algorithm is suggested to solve it. [Globus *et al.*, 2004] builds a benchmark for this type of problem with different realistic use cases for one to three satellites. The compared algorithms include hill climbing, simulated annealing, tournament selection genetic algorithm and elitist genetic algorithm with three different mutation operators enabling permutations between acquisitions and maneuvers. Simulated annealing was found to show better results than hill climbing which in turn outperformed genetic algorithms. Based on this work, **[Bonnet, 2017] proposes a multi-agent system** adapted to the processing of new acquisition requests for the optimization of a set of acquisition plans. The system developed is able to integrate new acquisitions requests to existing plans without recomputing and is highly scalable. Better results are obtained with this multi-agent system when compared to a state-of-the-art hierarchical greedy algorithm.

The optimization of a single satellite plan is a complex problem as is demonstrated in chapter 1 section 5 of [Bonnet, 2017]. [Lemaître *et al.*, 2000] finds the problem to be **NP-Hard**: a given solution to this problem can be verified in polynomial time but not found quickly. [Hall and Magazine, 1994] describe it as the "Space Mission problem" and also categorize it as NP-Hard. Solutions to NP-Hard problems are often solved through heuristics and approximate methods such as the algorithms listed above. We then consider **the *LAC* problem as a planning problem of several satellites over a planning horizon** defined by their passes over a single *AOI*. In this case the relative importance of meshes is not a relevant metric as a single large acquisition request is considered.

However the waste as defined in chapter 2.4.3 is a new metric to optimize that appears with the introduction of a pool of satellites and meshes that can overlap. Another complexity layer is added as the *MPF* responsible for the *LARM* task must emit acquisition requests to other *MPFs* operating with a hidden internal planning logic. This is another stochastic element of the problem to add to the potential differences between weather forecast and observed weather. **From these observations we suggest the *LAC* problem is similar to the Space Mission problem in complexity**. Both problems require selection of sequential tasks with uncertainty through weather forecast with additional constraints in the case of the *LAC* problem with the introduction of *MPF* answers.

LAC Problem. When compared to the Space Mission problem, the *LAC* problem has fewer examples available in the literature. [Fei and Zhi, 2011] considers the observation of an area of China large enough to require observation by several satellites. The authors define strips instead of meshes as illustrated in figure 3.3. The authors also describe a strip grid and a greedy algorithm for a small set of satellites to acquire the target strips in minimal time. [Xu *et al.*, 2018] proposes a formalization of the *LAC* problem. A discretization of the search space with a set of point coordinates to acquire in order to complete the observation mission is described. The problem is solved by a genetic algorithm optimizing the selection of strips and showing good results when compared to a greedy-based heuristic algorithm. [Holvoet

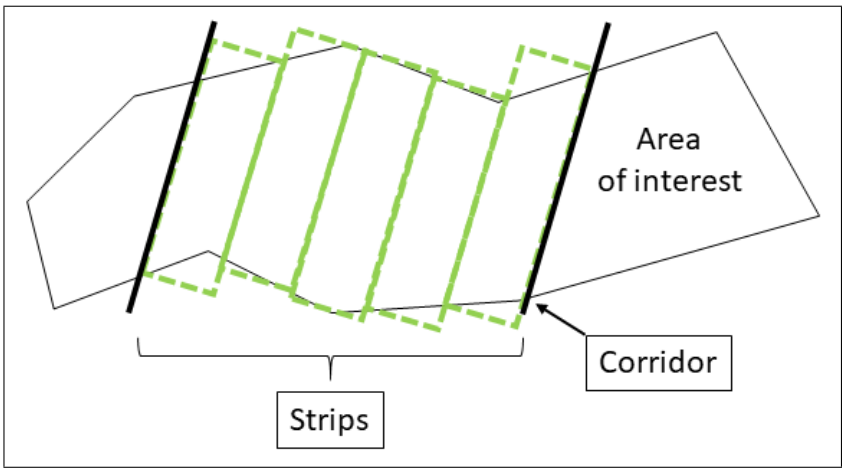

Figure 3.3: Illustration of an *AOI* subdivided in strips

et al., 2018] suggests a similar point based discretization for disaster observation missions. The authors describe the strip building process and solve the problem with a simulated annealing algorithm. [Niu *et al.*, 2018b] adapted a multi-objective genetic algorithm to solve the *LAC* with strips. The authors obtained better results than the greedy algorithm used as benchmark.

Strips are commonly used in large area observation missions. The use of strips simplifies the problem and search space to finding a set of observation angles for a set of satellite passes over the *AOI*. Acquisition with strips also allows the simplification of weather constraints. The observation angle of a satellite for a pass is fixed without possible maneuvers and as a result weather is not considered as an element of the problem. Using strips helps with addressing problem complexity but it has several drawbacks. First, **strips underuse the agility of the considered satellite**. Acquisition of strips only requires the roll angle out of the three rotation angles available to agile satellites.

Ignoring weather forecasts also causes increased waste on average. Weather may be favorable for acquisition on parts of the strip and unfavorable on some other parts. Finally strips are harder to fit in an existing acquisition plan. *LAC* mission are often low priority in the hierarchical list of requests managed by the *MPFs*. Satellites may be available for acquisition on parts of the strip and unavailable on other parts causing the strip to not be acquired and adding delay to the mission. **We choose to consider meshes** as they allow **more precise targeting** of favorable weather areas, **fully use the agility of satellites** and are **easier to integrate in an existing plan** due to their short acquisition time requirements.

The *LAC* problem with meshes is to our knowledge not treated in the literature. The geometric problem of large area coverage by strips is already difficult. The addition of weather forecasts increases the size of the search space as each acquisition may result in acceptation or rejection due to cloudiness of the acquired image. To solve the *LAC* problem with meshes another solving approach is explored at the IRT Saint-Exupéry and was presented in [Hadj-Salah *et al.*, 2019]. The authors consider a mesh grid using static meshing and a problem formalization similar to the one presented in section 3.2. The problem is described as an *MPF* and *Deep Reinforcement Learning (DRL)* is used to solve it. Results

obtained with *DRL* after training outperformed a greedy heuristic on a large coverage use case of France. Several simplifications are used to allow the use of *DRL*: a single mesh is acquired on each satellite path and all satellites share similar acquisition characteristics.

In this thesis, **we allow multiple acquisition per pass but choose to ignore maneuver restrictions** for several reasons. First the maneuver computation tools are specific to each *MPF* and ignored by the *LARM* when deciding which mesh acquisition to request. The acquisition plan of each satellite is also not known by the *LARM* when requesting acquisitions from *MPFs*. Other client requests may cause rejection of requests or impossible maneuvers if planned by the *LARM*. We also choose to consider satellites with differing acquisition characteristics as presented in the following.

3.4.3 Optimization Overview

Several types of optimization methods can be used to solve the *LAC* problem. Different approaches can be evaluated using the solving metrics presented in section 3.3.2. The choice of which algorithm to use in order to optimize the *LAC* problem is discussed in this section. First we present the groups of suitable optimization algorithms and their characteristics. We then draw a comparison considering the previously mentioned solving metrics with each optimization algorithm. Finally we justify the choice of a multi-agent system to solve the *LAC* problem using a new meshing approach that we call **dynamic meshing** and present in section 5.2.1.

Optimization is the search of solutions of ever-increasing quality through the search space of a problem. Search spaces can be enumerable or represented by equations or models. In the case of the *LAC* problem the search space is considered large enough that **Exact methods** are not applicable. Exact methods search the entire search space to find the optimal solution. As mentioned in section 3.4.2 the *LAC* problem falls in the NP-Hard category of optimization problem difficulty. Exact methods could eventually find the optimal solutions for such problems but the time required is considered impractical. **Approximate methods** explore the search space using cuts and **heuristics** to guide them towards solutions of good quality. [Lacour, 2014] presents an overview of different exact and approximate approaches for combinatorial optimization problems using the weight spanning tree problem as example. Heuristics are functions evaluating the best path to follow at each step of the search tree and as such are judged based on the quality of solutions found using them. In the following several approximate optimization algorithms using different types of heuristics are presented.

3.4.3.1 Greedy Algorithm

A greedy algorithm tries to find good solutions using only vertical search through the search tree. At each decision node the algorithm selects the best evaluated path and cuts the rest from the search. This approach prioritizes **exploitation** which refines existing solutions to improve them over **exploration** which finds other solutions to expand on available options [Macready and Wolpert, 1998]. **Greedy algorithms cannot backtrack** once a node is chosen and rely on short-term heuristics to guide the solving through the search tree. As such they

are easy to implement and require very little computation time. However constraints in solving can lead the algorithm to be trapped in local minima and to cut long-term quality solutions that appear worse on short-term evaluations.

An example of a greedy algorithm developed to solve the *LAC* problem is presented in [Xu *et al.*, 2020a]. The authors abstract weathers constraints from the problem and use strips to cover the *AOI*. **This approach may lead to local minima trapping** as the decision of selecting a strip is never challenged afterwards and a single path through the search tree is explored. Similarly, positioning meshes on a pass leads to local minima as only one configuration of collective mesh placements is considered.

3.4.3.2 Constrained Optimization

Constrained optimization, presented in [Box, 1965], places constraints on the variables of the objective function to optimize solutions found. These constraints can either be hard meaning they must be respected to find a valid solution or soft meaning the evaluation suffers a penalty if the constraint is not respected. To perform constrained optimization the problem must be formalized as a *Constraint Optimization Problem (COP)* with its constraints and then solved typically by a branch-and-bound algorithm [Narendra and Fukunaga, 1977]. These algorithms store the best solution found at any point and use backtracking to skip parts of the search that are evaluated as worse than the best solution.

[Kearfott, 1992] and [Demeulemeester and Herroelen, 1992] use branch and bounds algorithms to solve different constrained optimization problems. [Lawler and Wood, 1966] presents a survey of different branch and bound methods. Branch and bound algorithms have also been used to solve space planning problems as seen in [Chu *et al.*, 2017] and [Madakat *et al.*, 2018]. Constraints can be found to convert the *LAC* problem in a *COP* but **exploring the search tree through node exploration proves challenging due the uncertainties of the problem**. Weather observations may lead to many different states and mesh status which renders lateral exploration and cutting difficult.

3.4.3.3 *Monte-Carlo Tree Search (MCTS)*

MCTS presents similar issues in solving the *LAC* problem. *MCTS* requires formulating the problem as a *Markov Decision Process (MDP)*, a 4-tuple of (S, A, P_a, R_a) with S the states, A the available actions, P_a the probability that action a in state s at time t will lead to state s' at time $t+1$ and R_a the expected reward received after transitioning from state s to state s' after taking action a. *MCTS* uses simulations called roll-outs to explore the search tree vertically and put weights on expanded nodes based on an evaluation of the state reached. The most promising nodes are expanded in priority to cut the search of paths with worse evaluations.

The best-case problem for *MCTS* is deterministic as each node expansion leads to a set list of children nodes to explore. Let us consider the *LAC* problem solved by *MCTS* with a set of mesh placements in a pass as a node of the search tree. Each node expansion leads to many different children nodes based on the difference between predicted and observed weather for each selected mesh of the parent node. **This greatly increases the search space and**

makes *MCTS* impractical to solve the *LAC* problem in reasonable time without significant problem-dependent modifications [Świechowski *et al.*, 2023].

3.4.3.4 *Genetic Algorithm (GA)*

GAs have also been used to solve space planning mission previously. A *GA* formulates individual solves of a problem as individuals in a population. Each individual has a set of properties called genome which can be altered to change decision processes in the problem. A fitness function is used to measure the value of solutions found by individuals of the population. After each solving by the population a subset is created with the fittest individuals. Mutations may be applied to the genome of some of the selected individuals to create new solutions, a process similar to a random jump in a tree searching method. A new solving is then performed using the subset population of individuals. This process continues until certain conditions are met, which can include reaching time or fitness thresholds. Some researchers have explored *GAs* as potential algorithms to solve satellite related problems.

[Mansour and Dessouky, 2010] presents a *GA* to solve the daily single satellite scheduling problem. [Niu *et al.*, 2018a] uses NSGA-II[Deb *et al.*, 2002], a *GA* that creates a mating pool composed of parent and offspring populations, selecting the best solutions with respect to fitness and spread. [Xu *et al.*, 2020b] developed a genetic algorithm to solve the *LAC* problem using agile satellites and strips. This study proposes a genome that uses 0 to signal that a specific strip was not scheduled for acquisition and 1 otherwise. The study shows improved result when compared to greedy algorithms, however **cloud coverage is not considered and as a consequence selected strips may result in wasted acquisitions regularly**.

3.4.3.5 *Evolutionary Algorithm (EA)*

EAs represent a broad family of optimization algorithms inspired by Darwinian evolution concepts. An *EA* simulates solutions as individuals of a population and use different functions such as reproduction, crossovers and mutations to search through the population individuals with high fitness. *GAs* are a subtype of *EAs* that encodes the data of individuals as genes and specifically uses crossovers and mutations. Examples of other *EAs* cited in [Bartz-Beielstein *et al.*, 2014] include *Evolutionary Programming (EP)*, *Evolution Strategy (ES)* and *Genetic Programming (GP)*.

EP only uses mutations as operator and selects individuals based on a probabilistic fitness function. *ES* use small global variable changes to parents and mutated offsprings to select the fittest individuals. *GP* uses similar strategies as *GAs* but the individuals are computer programs that are refined to perform a user specified task. While *GAs* are the most commonly used *EAs* to solve observation satellites planning problems, other studies include approaches using other types of *EAs*.

[Chang *et al.*, 2022] combines NSGA-II with an *Adaptive Large Neighborhood Search (ALNS)* algorithm and adds "Destroy" and "Repair" operators to the evolutionary cycle. [Li and Li, 2019] proposes a *Multi-objective Binary-encoding Differential Evolution (MBDE)* to solve the daily satellite scheduling problem without weather uncertainty. The authors use discrete

values to simulate the start and end of acquisition windows and achieve better overall results when compared to the NSGA-II algorithm. As for *LAC* related research, [Li *et al.*, 2023] proposes an improved NSGA-III algorithm called ESP-NSGA-III for the *LAC* problem without weather uncertainty and using strips. The proposed algorithms introduces extreme solution preservation to keep extreme individuals with specific regional solutions in the evolution cycle of the *GA* and thus improve the diversity of solutions available for a mission.

While these studies all abstract cloud coverage as a simplification of both the daily scheduling and the *LAC* problem, **they improve the use of traditional *GAs*** as was presented previously. The modification or addition of operators allows for more specific approaches to each problem with improved results in performance or diversity of solutions.

3.4.3.6 *Simulated Annealing (SA)*

SA is an optimization method based on the physical process of materials where the temperature is adjusted gradually. *SA* uses a state with neighbours at each iteration of the algorithm. The algorithms tries to solve the problem by reaching an approximated global optimum represented by a low energy state. A change of state can worsen the current solution which reduces the risk of being trapped in a local minimum. An acceptance probability function regulates the probability of transitioning to a neighbouring state. The algorithm starts with a temperature T as a high value (or infinity) that represents the energy left in the system. For high T values, the acceptance probability function allows for transitions to higher energy neighbours (lower quality solutions) more easily. For low T values, transitions can only be made towards a state of lower energy than the current one.

This process enforces different rates of exploration, starting with a large exploration and narrowing to an exploitative process towards the end of the annealing. The algorithm stops when a certain quality threshold is reached or when a computing budget is exhausted. The main advantages of *SA* and similar local search algorithms are their **resistance to being trapped in local minima** and their **ability to search through large-dimension state spaces** where population based algorithms would encounter memory issues. *SA* is one of the most used and compared to algorithm in the field of satellite optimization problems for its simplicity of implementation and due to the scale of the problems.

[Liu *et al.*, 2022] considers the multi-objective multi-satellite downlink problem and optimizes both profit and efficiency to satisfactory results. [Wu *et al.*, 2017] proposes an adaptive *SA* for the daily satellite scheduling problem that uses clusters of scheduling tasks to reduce satellite maneuvers and limit the search space. [Han *et al.*, 2022] solves the daily satellite scheduling problem with weather uncertainty using a *SA* with a fast insertion strategy to include new acquisition requests. The authors compare the results with other algorithms such as a *GA* and *ALNS* to obtain positive efficiency and effectiveness results. [Waiming *et al.*, 2019] uses a hybrid *GA* and *SA* algorithm for a version of the daily satellite scheduling problem that includes transmission operations with ground stations. The authors show the feasibility of a two-phase algorithm that finds high quality candidate solutions with a *GA* before switching to a *SA* and solving the issue of local optima trapping.

SA tends to perform well to solve problems similar to the *LAC* problem such as the daily satellite scheduling problem. However ***SA* can lead to precocious solves** as the energy of the system decreases which leads to local optimum trapping. **The uncertainty of the states is often not considered in the cited studies** which could be an issue due to a variable decrease in temperature of the states depending on the outcome of the uncertainty parameters.

3.4.4 Analysis and Comparison

The literature on satellite scheduling problems is comprised of problems similar but not exactly applicable to the *LAC* problem as we define it in this study. The daily satellite scheduling problem is the most encountered, with different hypotheses including the agility, number and characteristics (*i.e.* optical or radar lenses) of satellites. **For studies considering these problems, the weather uncertainties are usually abstracted** to reduce the search space to deterministic states involving the scheduling and switching of observation tasks. This approach to the daily satellite scheduling problem is common as it can then be categorized as a multi-dimensional knapsack problem as seen in [Bensana *et al.*, 1996], which helps with benchmark and comparison to other algorithms in the literature.

The insertion and modification of observation tasks in visibility windows is the focus of these studies. Works on the *LAC* problem often abstract this process by restricting the agility of the satellites used. In these studies the *AOI* is made discrete through a number of points placed on the *AOI* that all need to be acquired to complete the mission. These points are acquired by locking the satellites' rolling ability and acquiring continuous meshes as long as the *AOI* called "strips". **Using strips simplifies the *LAC* problem** to a problem of selecting which strip to acquire in which satellite pass. It reduces the need for satellite coordination as the acquisitions are strictly distinct without overlap. Weather uncertainties are also often abstracted since an acquisition resulting in a clouded image will be repeated without change on another pass.

However **strips cause a number of issues when considering the *LAC* problem** as defined in section 3.2. The restriction to a single roll angle for satellites may lead to acquisition of parts of the *AOI* where the weather was predicted poor. The acquisition of meshes instead of strips allows for the change in roll angle and acquisition of other parts with higher likelihood of desirable weather. As such **strips cause waste, and in most cases lead to an increase in completion time**. Each strip that fails to be acquired due to clouded images will be re-acquired without change to respect the strip grid created over the *AOI*.

In the case of meshes being used, a clouded area can be re-acquired using a mesh positioned differently to cover both the mentioned area and other areas that are still to-be-acquired. This flexibility makes them easier to fit around constrained areas than strips which are always as long as the *AOI* permits. **Meshes also allow *MPFs* to integrate urgent requests** to the satellite plans and still continue the acquisition of the large area as a background task, whereas strips need to be cancelled and re-acquired on a later pass in the case of urgent request integration.

This overview presented approaches to problems similar to the one detailed in section 3.2. The daily satellite scheduling problem treats meshes but considers a smaller scale due

Algorithms	Computing time	Robustness	Scalability	Local minima sensitivity
Greedy	++	-	- -	-
Constrained	++	- -	++	- -
MCTS	- -	+	- -	+
GA	- -	+	+	++
EA	- -	+	+	+
SA	-	+	++	+

Table 3.1: Evaluation of the *LAC* problem solving metrics per optimization algorithm graded from - - (low performance) to ++ (high performance) for each category

to the point of view of a single *MPF*. The *LAC* problem using strips takes an *LARM* point of view but restricts satellite agility and ignores weather to limit the complexity of the problem. Due to the increase in the search space caused by the introduction of meshes and the difference in metrics evaluated we need to consider the **adequacy of the optimization algorithms** when considering the ***LAC* problem with weather uncertainty and meshes**. Based on previous observations, we propose in table 3.1 a comparison of the characteristics of previously mentioned algorithms when applied to the *LAC* problem. The evaluation for each criterion scales from - - (low performance) to - (below average performance), + (above average performance) and ++ (high performance).

3.4.5 Conclusion

In this chapter we presented the *LAC* problem with meshes and its formalization with relevant problem and solving metrics. Based on these definitions, we presented the operational solving of the problem and an array of possible optimization algorithms used to solve similar problems. The optimization algorithm used to solve the *LAC* problem needs to be robust to the different operational conditions met in observation missions. Due to the size of the *AOI* sent to the *LARM* for acquisition, the scalability of the algorithm is also important. The solutions must be found in reasonable computation time and need to avoid local minima that are present in the *LAC* problem.

From these observations, we propose the use of *AMAS* as an optimization method to solve the *LAC* problem. We suggest that using *AMAS* answers several difficult issues in the solving of the *LAC* problem:

- ▷ a global evaluation function is not needed, which in the case of such optimization problems is often difficult to find and requires significant computation time,
- ▷ the parallelization of the system is inherent due to the distributed agents processing simple personal algorithms,
- ▷ the cooperative behavior is adequate to the integration of environment updates during the solve, for example weather updates in the case of the *LAC* problem.

In this thesis we focus on the optimization of the scale of the *LAC* problem as the central issue to its efficient solving. For this reason *AMAS* are studied as good optimization

candidates over the existing examples present in the *LAC* literature.

In chapter 4 we present the *AMAS* theory, the methodology used to apply it to the *LAC* problem and show the adequacy of *AMAS* due to their characteristics related to the ease of treatment of both scalability and robustness of the problem.

4 Theory and Tools

4.1 Introduction

As stated in the previous chapter, in this thesis we propose a cooperative multi-agent system for the solving of the the *LAC* problem. The proposed system was designed using the ***AMAS* theory** and the ***AMAS for Optimisation (AMAS4Opt)* agent model** presented in section 4.2 and 4.3 respectively. The multi-agent system was developed as a research contribution to the **Synapse project** of the IRT Saint-Exupéry, a project which focused on mission planning for Earth observation satellites. The tools used within the Synapse project to develop the multi-agent system are presented in section 4.5.

4.2 The *AMAS* Theory: Cooperative Self-Organization

The *AMAS* theory [Gleizes *et al.*, 2008; Georgé *et al.*, 2003; Glize, 2001; Gleizes *et al.*, 1999] is based on a **bottom-up approach** starting from the **autonomous entities** of the problem and describing their **cooperative interactions** in order to design an **adaptive multi-agent system** from which the solution of the problem **emerges**.

It is well-known in computer science that different algorithms can be functionally adequate for a given problem, i.e. the proposed algorithm is able to solve the problem for which it is designed. The *AMAS* Theory considers that *for any functionally adequate system, there exists at least one cooperative internal medium system that fulfills an equivalent function in the same environment* [Glize, 2001]. In other terms, by focusing on (and hence understanding) a subset of particular systems (those with cooperative internal mediums) it is possible to obtain a functionally adequate system in a given environment. We concentrate on a particular class of such systems, those with the following properties [Gleizes *et al.*, 1999]:

- **the system is cooperative and functionally adequate** with respect to its environment. Its parts do not 'know' the global function the system has to achieve *via* adaptation;
- **the system does not have an explicitly defined goal**, rather it acts using its perceptions of the environment as a feedback in order to adapt the global function to be adequate. The mechanism of adaptation results in the agents trying to maintain cooperation using their skills, representations of themselves, other agents and the environment;

- ▷ **each agent only evaluates whether the changes taking place are cooperative from its point of view** – it does not know if these changes are dependent on its own past actions;
- ▷ basically, the idea is that **it is easier and more efficient to design systems by focusing on the agents and the cooperative self-organising mechanisms which will result in the adequate system**, rather than trying to produce the right global system directly.

Applications. Several multi-agents systems based on this theory have been designed to efficiently solve a wide variety of applications. [Perles, 2017] proposes an *AMAS* based framework for the solving of common problems encountered in power systems such as autonomous electrical networks. The study shows the adequation of *AMAS* systems for voltage regulation in such highly evolving environments. [Brax, 2013] detects anomalies in maritime surveys using an *AMAS*. The author justifies the possibility of instantiating the proposed model to other surveillance systems due to its robustness and genericity. [Bourjot *et al.*, 2003] draws from the behavior of social spiders to propose a reactive multi-agent system. The resulting swarm mechanism is able to integrate non-local information during the processing in order to make the system more adaptable to change.

[Verstaevel *et al.*, 2016] introduces an *AMAS* able to learn from the demonstrations of a human tutor and reuse contexts in its own use cases. Results on ambient robotics experiments show behaviors that are user satisfying despite the highly combinatorial observation space of the problem. [Guastella, 2020] estimates missing environmental information in large scale *Internet of Things (IoT)* urban environments. The *AMAS* developed in this study is very adaptable to changing urban contexts and able to handle a large number of heterogeneous devices.

The results of these studies contribute to our interest of an *AMAS* application to the *LAC* problem. The **size of the search space**, **possibility of alterations in the environment** and **possible heterogeneity of the problem elements** all match the description of the challenges met with the *LAC* problem.

4.2.1 Cooperation

Coordination is a central issue in the field of distributed multi-agent systems. Interactions between agents need to help them perform their respective activities and achieve their goals. The coordination of agents is a known problem in the multi-agent literature. [Mandiau *et al.*, 2008] describes a coordination mechanism based on decision matrices for the solving of simulated traffic problems. [Beer *et al.*, 1999] presents different agent interactions based on negotiation such as auctions, contract nets and argumentation. [Terán *et al.*, 2013] proposes mathematical models for the different types of coordination mechanisms present in the multi-agent literature. In this thesis we focus on **cooperation** as the mechanism of coordination between agents.

Cooperation is defined as the ability agents have to work together in order to realize a common global goal. It implies that the activities of the agents are supplementary, and dependency links and solidarity exist between them. They have a cooperative attitude that satisfies four properties:

1. **Sincerity**: If an agent knows that a proposition p is true, it cannot say anything different to others;
2. **Willingness**: Agents try to satisfy a request if it is coherent with their own skills and the current state of the world, and if no prejudice results from the action, either to the acting agent or to another. If there is a resulting prejudice, refer to property three;
3. **Fairness**: They always try to satisfy, when it is possible, agents with the higher level of difficulty or criticality;
4. **Reciprocity**: Each agent of the same society knows that it and the others verify these three main properties.

The objective is to design systems where agents:

- ▷ do the best they can when they encounter difficulties. These difficulties can be viewed as exceptions in traditional programming. From an agent point of view, we call them *Non Cooperative Situation (NCS)*. An agent locally tries to detect such failures and try to repair them.
- ▷ **anticipate** *NCS*. The agent always chooses the actions with minimal disturbance to the other agents it knows. It tries to anticipate [Doniec *et al.*, 2005] (for him and others) problems that can introduce *NCS*;
- ▷ are **cooperative** towards the system and other agents. The first point implies that agents not reaching their own goal can be seen as generating *NCS* that must be repaired. Being cooperative toward other agents implies that when detecting or anticipating *NCS* agents try to always help agents with the higher level of difficulty ;

To resume, the *AMAS* theory focuses on designing cooperative agents that act by reorganizing their acquaintances and interactions adequately with the others agents. The *NCS* are cooperation failures that an agent must anticipate or detect and repair. Their handling follows the agent life cycle : **perceive, decide and act**. Thus, an agent is in a *NCS* when:

- ▷ a **perceived** signal is not understood or is ambiguous;
- ▷ a perceived information does not **produce any new decision**;
- ▷ the consequences of its **actions** are not useful to others.

Given that, the *AMAS* Theory defines seven *NCS*.

- ▷ **Incomprehension**. The agent is not able to extract any understandable information from the received message in its perception life cycle step.
- ▷ **Ambiguity**. Related to the update of the local representation in the perception life cycle step, this *NCS* informs the agent that different interpretations are possible, and by that, an accurate representation update is not possible.

- ▷ **Incompetence**. The agent does not have the competence to treat received requests from its neighborhood (decision step).
- ▷ **Unproductivity**. During the decision life cycle step, the agent is unable to propose an action to do during the action life cycle step.
- ▷ **Conflict**. This *NCS* is detected when the agent considers that modifying the environment can prevent other agents from reaching their goals (i.e. when using a resource required by another agent).
- ▷ **Concurrence**. The agent is able to perform the actions but considers that there are other agents able to perform the same actions and reach the same state in the environment.
- ▷ **Uselessness**. This last *NCS* is detected when the agent considers its actions or itself not useful for the system or its environment.

To solve a *NCS*, three generic behaviors can be used by an agent:

- ▷ **Reorganization**: change its relation with other agents, meaning contact other agents, stop interacting with a given agent or change the importance allocated to existing relations.
- ▷ **Adjustment**: modify its behavior by adjusting its internal parameters.
- ▷ **Openness**: decide to create a new agent or to delete itself.

4.2.2 Achieving Self-Adaptation and Self-Organization

We consider that each part P_i of a system S achieves a partial function f_{P_i} of the global function f_S (figure 4.1). f_S is the result of the combination of the partial functions f_{P_i}, noted by the operator "$\circ$". The combination being determined by the current organization of the parts, we can deduce $f_S = f_{P_1} \circ f_{P_2} \circ ... \circ f_{P_n}$. As generally $f_{P_1} \circ f_{P_2} \neq f_{P_2} \circ f_{P_1}$, by transforming the organization, the combination of the partial functions is changed and therefore the global function f_S changes.

This is a powerful way to adapt the system to its environment. A pertinent technique to build these kinds of systems is to use adaptive MAS. As in Wooldridge's definition of **multi-agent systems** [Parsons and Wooldridge, 2002], we will be referring to systems constituted by several autonomous agents, plunged into a common environment and trying to collectively solve a common task.

4.2.3 AMAS Agents

Agent Life Cycle. The agents of an *AMAS* follow a life cycle including phases known as **perceive**, **decide** and **act**. To cooperatively optimize progress and waste, **agents are able to communicate using messages**. Each agent unpacks messages during the perceive phase, updates their own status and/or chooses possible messages to send during the decision phase, and emits the chosen messages during the act phase. These messages can be requests,

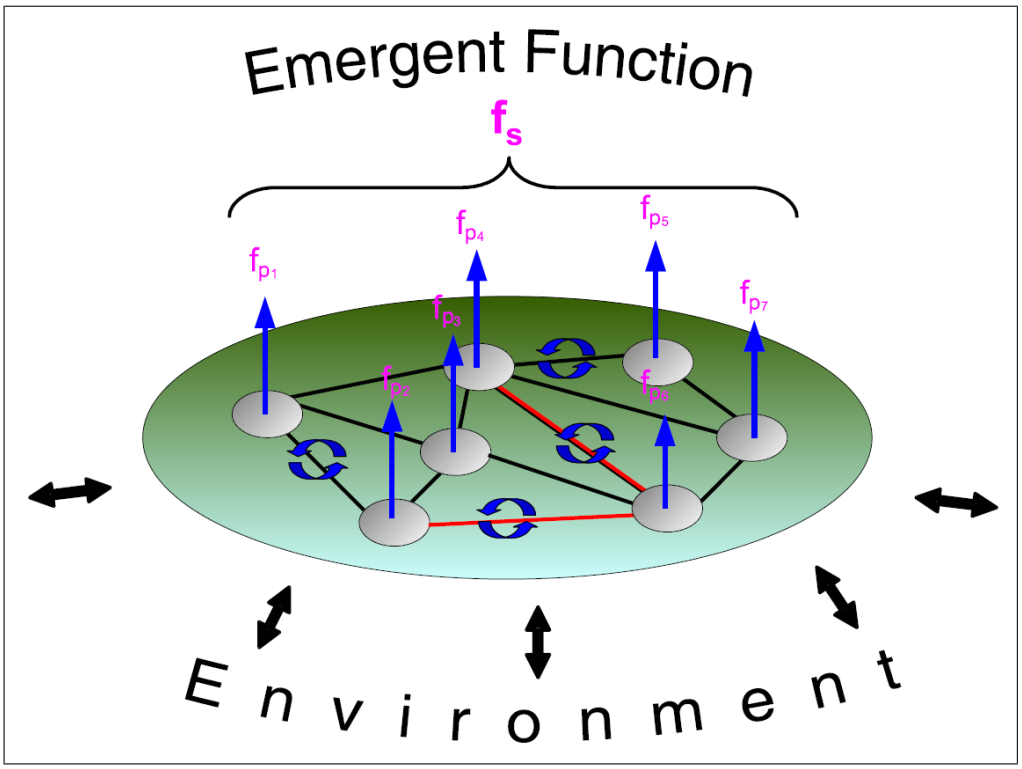

Figure 4.1: Adaptation: changing the function of the system by changing the organization

status updates or acknowledgments. Several messages can be emitted at once by an agent during its act phase but each message is addressed to a single agent only. Messages contain the identity of the emitting agent and the data required for the receiving agent to decide the action to perform.

Communication. Messages are used to ensure the coherence of the system and homogeneity of information between the agents of the system. **Agents run independently and asynchronously**, as a result the direct gathering of data about another agent is subject to errors as the information could have been modified either before or after the gathering during the current cycle of the agent. Messages are an answer to this issue but can only be used as representations of the state of the emitting agent in the previous agent cycle. Acknowledgments and updates are then required to confirm the expected actions were taken and had an impact on the environment. **Using messages can be avoided** when an information that is not subject to change during agent cycles is required by an agent. **The information can then be gathered directly** in order to reduce the message load. These direct exchanges of stable information will not be mentioned in the following for simplification reasons. For example the length of a mesh is an information subject to change which may be gathered through an exchange of messages between agents, while the satellite associated to a pass is not subject to change and may be gathered through direct observation.

4.2.4 *ADELFE*

Designing adaptive multi-agent systems requires a specific methodology different from the top down traditional methods. Indeed, in such systems, we concentrate on the different parts of the system and their interactions. The global function of the system emerges from these interactions. Thus, the ***Atelier de Développement de Logiciels à Fonctionnalité Emergente (ADELFE)*** methodology [Bernon *et al.*, 2002; Picard and Gleizes, 2004; Bernon *et al.*, 2005; Rougemaille *et al.*, 2008] was developed. ***ADELFE* is a toolkit to guide designers through the development phase** of complex, open and distributed systems based on the *AMAS* theory and the concept of emergence. It is based on some well-known tools and notations coming from the object-oriented software engineering: *UML* (Unified Modelling Language) and *RUP* (Rational Unified Process). It uses *AUML* (Agent-UML) to express interaction protocols between agents.

4.3 The *AMAS4Opt* Agent Model

AMAS4Opt*.** To specify the usage of the *ADELFE* methodology for solving optimization problems under constraints, the ***AMAS4Opt agent model was introduced in [Kaddoum, 2011]. The goal of the model is to **allow the development of *AMAS* applications for multi-disciplinary and multi-objective problems**. Generic behaviors and local interactions are defined to facilitate the solving of complex optimization problems. *AMAS4Opt* has been used to solve optimization problems in varied fields of research. [Degas, 2020] develops an *AMAS* with cooperative agents based on *AMAS4Opt* to solve multi-objective air traffic control problems. In a similar context, [Mykoniatis *et al.*, 2017] address delays in flight schedules with *AMAS4Opt* agents estimating probabilistic states of other agents. [Bonnet, 2017] uses *AMAS4Opt* for the daily satellite scheduling problem and injection of urgent requests in the mission plans of observation satellites.

Optimization. *AMAS4Opt* agents are specifically designed to solve complex optimization problems. As such **the *AMAS4Opt* model proposes two agent roles** based on their goals and resources: a ***Constrained role*** and a ***Service role***. Agents can have one or both roles depending on the context. A *Constrained* agent can use **request** messages directed to a *Service* agent to perform a specific service. The consequences of the service include an **impact on the environment**, **modifications in the internal representation of the *Service* agent** or the **gathering of information for the *Constrained* agent** through **answer** messages. Agents send requests depending on their internal algorithm and environment and may send multiple requests of different types to the agents in their neighborhood during a single life cycle. Requests are sent to contribute to the local goal of the agent or to solve an identified *NCS*. Thus, each request is associated to a measure of criticality, representing the requesting agent's difficulty or distance to reach its goal. In section 5.3 we present **Glimpse**, an *AMAS* for the solving of the *LAC* problem. Its agents are specified with their goals, roles and interactions as developed under the *ADELFE* toolkit and *AMAS4Opt* agent model.

4.4 *AMAS* for the *LAC* Problem

In section 3.3.2 the solving metrics of the *LAC* problem are defined. The calculation time, algorithm robustness, scalability and local minima sensitivity are evaluated when discussing the adequacy of an algorithm to the problem. In this chapter the properties of the multi-agents systems based on the *AMAS* theory are presented. Problems solved using *AMAS* include difficult characteristics such as high combinatorial, dynamic environment, missing information and heterogeneous problem elements. Recent studies regarding multi-objective optimization problems use the *AMAS4Opt* agent model to facilitate the development of agents and their communications for the solving of these problems. Good results are obtained in resource allocation problems such as the daily satellite scheduling and air traffic control problems. **Due to the similar nature of the *LAC* problem, we suggest *AMAS* based solution to be adequate**.

The **bottom-up approach** of *AMAS*, where simple elementary algorithms are designed for each agent instead of a complex overarching algorithm, matches with the scalability criterion. Indeed **the interactions between agents do not grow in complexity with the size of the search space**. The nature of **distributed complexity also helps with the calculation time**. The agents' life cycle can be allotted to different calculation units to perform parallel tasks and lower the time required for the system to progress to the next cycle. **The solving of *NCS* between cooperative agents help with the avoidance of local minima trapping**.

Finally the robustness of the system depends on the behavior of agents in different experimental environments and is able to be tuned during the design of the *AMAS*. From these observations we suggest that an *AMAS* is an adequate optimization algorithm for the solving of the *LAC* problem. In chapter 6 the results of Glimpse on a variety of *LAC* scenarios are presented. Resolution metrics are discussed and table 3.1 of section 3.4.4 is expanded to include *AMAS*. This classification provides an estimation of the advantages and drawbacks to using *AMAS* to solve the *LAC* problem as formalized in this study.

4.5 Synapse Project Tools

Programming Language. *AMAS* applications have well defined entities in the form of agents. Agents are always exactly the same in their definition, changing only by their internal representation. As such, ***Object Oriented Programming (OOP)***[Rentsch, 1982] languages are suited to the development of *AMAS*. We decided to use **Java** as programming language to develop Glimpse. The popularity of the language ensured easier access to independent functionalities and libraries. The automatic memory management is specifically desirable when using *AMAS* due to the cyclic process of deletion and creation of agents. Moreover the *AMAS* framework providing the structure of agents with their life cycle, basic communication functions and agent scheduler called ***AMAK***[Perles *et al.*, 2018] was also developed using Java. The interface between *AMAK* and Glimpse was simplified considerably by using the same development environment, programming language included.

Versioning. For the versioning of Glimpse, we used an internal framework based on Github. This allowed for the group development and review of the *AMAS*, its *Graphical User Interface (GUI)* and additional programs such as an *ILP(Integer Linear Programming)* algorithm (annex A). The collaborators included members of the scientific contribution, the **SMAC** team from the **IRIT** laboratory and the **ISAE-Supaéro**, and professional contribution represented by **Airbus** and **Thales Alenia Space**. The scientific contributions including **Glimpse** are discussed in chapter 5.

Weather Data. The data used for the experimentations of section 6 was also provided by Airbus and Thales Alenia Space. The **satellites' ephemeris**, meaning their position and velocity through time, **their characteristics** and **the relevant *MPF* attributions** were collected from operational data of previously performed observation missions. Similarly the **observed weather** at the time of the these past observation missions is also collected. The observations are used to simulate forecasts as explained in section 6.2.2. The experiments presented in this study alter the characteristics of the satellites and their operating *MPF* but the ephemeris and weather are kept identical so as to approach operational conditions of actual observation missions. Indeed *LARM* operators cannot modify the ephemeris of the satellites in order to move the corridors of the passes over the *AOI* as the satellite may be shared with other clients with their own acquisition requests.

4.6 Conclusion

In this chapter we present a type of multi-agents systems called AMAS. AMAS make use of distributed and cooperative agents. These agents can communicate and react to their environment to achieve local goals. Through the use of *Non Cooperative Situations* and *agent roles* the agents self-organize and an emergent function is produced. AMAS have been used successfully in the solving of difficult optimization problems with dynamic environments and large search spaces. As such, they appear as a candidate optimization method for the solving of the *LAC* problem. In chapter 5 we propose the contributions of the thesis including the design of Glimpse, an AMAS for the solving of the *LAC* problem.

5 Thesis Contribution

5.1 Introduction

Some of the methods and tools presented in chapter 4 help with the design of any multi-agent system. Meanwhile methods like *AMAS4Opt* help with the design of multi-agent systems developed to solve complex optimization problems. In this chapter the Glimpse algorithm is presented. Glimpse is a multi-agent system that uses agents based on AMAS4Opt components to solve the *LAC* problem. The solving is distributed in each agent using communication and cooperation tools. The next sections describe the agents of Glimpse and discuss relevant design choices.

5.2 Problem Formalization

5.2.1 Dynamic Meshing

In section 3.4.1 static meshing was introduced as a common technique used by an individual *MPF* to divide an *AOI* in mesh placements to be acquired by **homogeneous** satellites - sharing similar characteristics. **Identical swaths are the best case for static meshing** as each mesh acquired by different satellites can be placed on a single grid if mesh dimensions are shared. The lack of overlaps between meshes means that each mesh needs to only be acquired once. This guarantees that no re-acquisition due to acquisition overlaps occur, potentially using satellite resources optimally. However **when using static meshing acquired images that are discarded due to bad weather will need to be re-acquired entirely** at the corresponding mesh placement. Meshes are also locked in their placement to the precomputed grid to ensure that no overlaps are occurring and cannot be moved or altered.

Heterogeneous Satellites. Static meshing can also be used in the *LAC* context with several *MPFs* and over a larger *AOI*. If used in this context the best use case of static meshing is when every satellite is homogeneous as no overlap occurs between meshes. However operationally *LARMs* most often send acquisition requests to *MPFs* operating different types of **heterogeneous** satellites - presenting different characteristics. If a precomputed grid of mesh placements is built using heterogeneous satellites the resulting grid presents overlaps between mesh placements. Figure 5.1 illustrates this case using 2 heterogeneous satellites

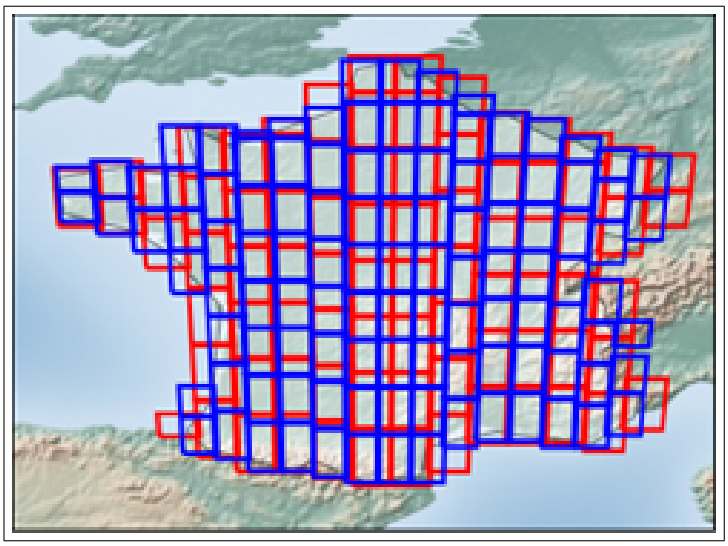

Figure 5.1: Example of overlapping mesh grids of heterogeneous satellites

and their respective mesh grids over the *AOI*. **Because of these overlaps acquisitions may partially re-acquire subparts of the *AOI* acquired by other satellites.** The guarantee of having no overlaps is only true for the different meshes of a single satellite, not with other heterogeneous satellites. Having forced overlaps between satellites increases waste but also problem complexity as the problem cannot be formalized as a single grid of enumerated meshes to acquire. Static meshing is less useful in this context as the rigidity of the method still exists while its benefits are reduced or nullified.

Dynamic Meshing. We propose a new meshing technique adapted to heterogeneous satellites called **dynamic meshing**. To answer the continuity of the search space we use a segmentation of the *AOI* that is common to all satellites. This segmentation is a precomputed grid of subareas called **cells**. Each cell represents an **elementary square zone to acquire** and has a length ranging from 1km to the greatest common divisor of all satellite swaths. The *AOI* is fully covered by cells and only a subset of cells is covered by the corridor of a given satellite pass. The **meshes inside the corridor are placed over this cell grid** that follows the same north/south axis. The placement and size of meshes may also change on the cell grid. The difference in mesh placement between static and dynamic in a sample corridor without considering optimality is highlighted in figure 5.2.

This discretization of the problem operated using a cell grid allows for a more flexible meshing. Indeed with dynamic meshing the **meshes can be placed and resized to fit geographical acquisition needs** during a satellite pass. A mesh can cover an area estimated to have clear weather more precisely by altering its position which is not a possibility with static meshing. The overlaps between meshes of different satellites discussed previously when using static meshing can be avoided using dynamic meshing by placing the meshes to border the already validated surfaces with as little overlap as possible. **These improvements cause meshes to be placed on areas that either contribute more to the global completion of the mission or that are of a higher probability of resulting in clear and usable images**. Meshes can also be placed on the border of passes or *AOI* more precisely as to

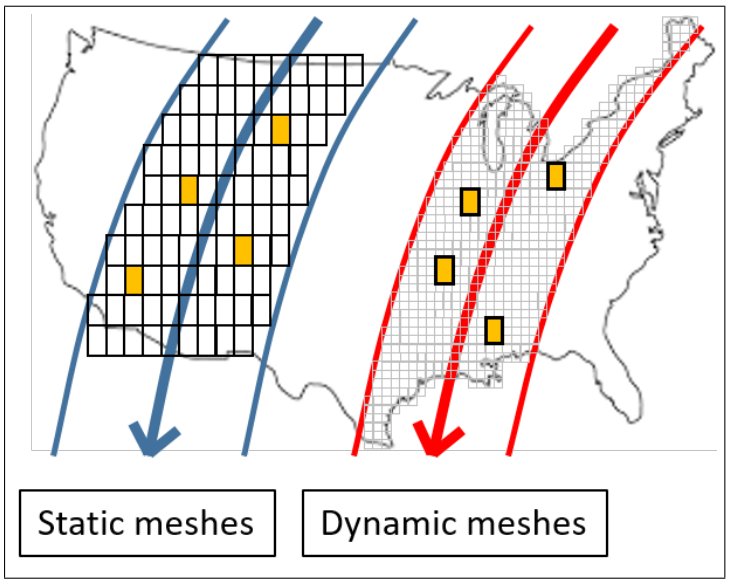

Figure 5.2: Comparison of static and dynamic meshes on the same corridor

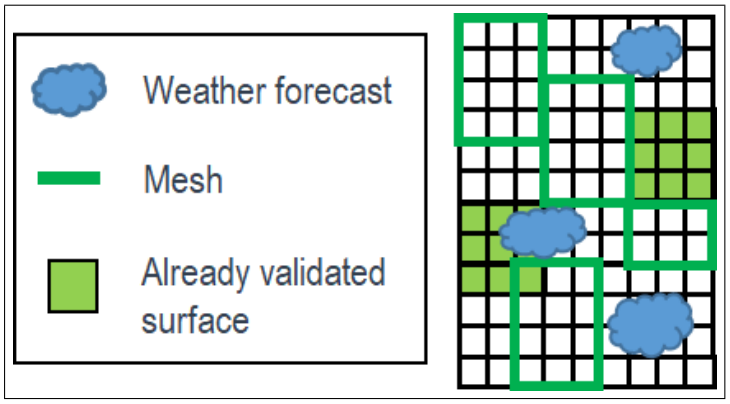

Figure 5.3: Dynamic meshes placed as to avoid cloudiness and already validated elementary surfaces

remain inside the corridor or *AOI* while targeting a specific sub-area.

This targeting cannot be performed using static meshing as dynamic meshes can be summarized as aggregates of cells, a list of elementary coordinates all included in an acquirable surface area. As a consequence dynamic meshes can be rearranged to fit around already validated cells and weather forecasts as seen in figure 5.3. This improves their likelihood of successful acquisition and optimizes the satellite resource usage for each mesh. In the case of heterogeneous satellites, the acquisition of static meshes often lead to re-acquisitions of parts of the meshes already acquired by another satellite. **Dynamic meshes can be placed to avoid re-acquisitions** and acquire more useful (non acquired and non validated) cells during each pass.

Partial Validation. Dynamic meshing also pairs well with **partial validation**, a technique where each cell is independently acquired and validated or rejected. This is favorable for dynamic meshing as meshes can be placed on future passes to precisely fill the gaps where cells were not validated whereas static meshes need to be re-acquired in the same position

due to the precomputed mesh grid. Partial validation is not used in the experiments of chapter 6 as its use is dependent of operational implementation in *MPFs*. We discuss it further as a future work perspective in chapter 7.

5.2.2 Problem Formalization

Using this new meshing technique of the *LAC* problem with dynamic meshing and its cell grid we propose the following formalization; given:

- A **zone** Z on which is placed a grid of width W and length L of cells $c_{11}, \ldots, c_{xy}, \ldots, c_{XY}$, with x and y the geographical coordinates of the lower left corner the cell c_{xy}. Z is divided into a set of cells $C = \{(x, y) \mid c_{xy} \cap Z \neq 0\}$ inside or partially inside the area. The cells are characterized by:
  - a **width** l_c and a **length** L_c per kilometer.
  - An **acquisition status** $status(c_{xy}, t) \in \{Active, Acquired, Validated\}$ indicating respectively if the cell c_{xy} is waiting for planning, included in an acquired mesh or included in a mesh validated at time t.
- The **set of satellites** $S = s_1, \ldots, s_j, \ldots, s_J$ of **minimum mesh acquisition width** ${f_j}^{min} * l_c$ and **maximum mesh acquisition width** ${f_j}^{max} * l_c$ with ${f_j}^{min}, {f_j}^{max} \in \mathbb{N}$; and **minimum mesh acquisition length** ${g_j}^{min} * L_c$ and **maximum mesh acquisition length** ${g_j}^{max} * L_c$ with ${g_j}^{min}, {g_j}^{max} \in \mathbb{N}$.
- The known passes $P = p_{j1}, \ldots, p_{jk}, \ldots, p_{jK}$ of satellite s_j. Passes are known up to a **planning horizon** H, in number of passes, to avoid weather prediction errors on distant passes. Passes are characterized by:
  - a **date of occurrence** $t_{p_{jk}}$
  - a **set of acquirable cells** $C_{p_{jk}} = \{(x, y) \mid c_{xy} \in Corridor_{p_{jk}}\}$
  - a **maximum mesh number threshold**
    $$|M_{p_{jk}}| < \frac{|C_{p_{jk}}|}{{f_j}^{max} * {g_j}^{max}}$$
    to avoid overloading the *MPFs* with mesh acquisition requests. This threshold is defined as the number of meshes of maximum size sufficient to cover all cells in the corridor of the pass.

Goal. The goal of the solving is to determine for each pass a **set of meshes** to include in an acquisition request defined as

$$M_{p_{jk}} = m_{p_{jk}1}, ..., m_{p_{jk}n}, ..., m_{p_{jk}N}$$
$$m_{p_{jk}n} = (x_n, y_n, f, g)$$

with:

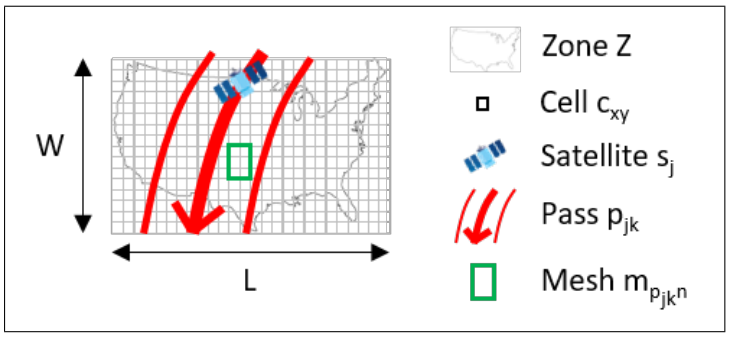

Figure 5.4: Visual representation of the formalized *LAC* problem with a cell grid and dynamic meshes

- x_n, y_n the coordinates of the lower left corner of the mesh,
- and $f_j^{min} \leq f \leq f_j^{max}$ and $g_j^{min} \leq g \leq g_j^{max}$.

Reaching this goal consists in aggregating the cells of a given pass to re-construct meshes considering the characteristics of the satellite performing the pass.

Figure 5.4 summarizes the graphical elements of this formalization with the example of a zone Z of width W and length L covered by a grid of cells c_{xy}, and a pass p_{jk} of the satellite s_j that contains in its corridor a mesh $m_{p_{jk}n}$.

Objectives. The main objective is to **minimize the time required** to complete the LAC mission. This objective is reflected in the validation of all cells in the area of interest. The completion of the mission is reached at a time noted t_{comp}.

The measure of progress in completing the project is noted as:

$$Progress(Z,\ t)\ =\ \left(\frac{\sum x,y\ |\ c_{xy} \in C \wedge status(c_{xy},\ t) = Validated\ \ c_{xy}}{\sum x,y\ |\ c_{xy} \in C\ \ c_{xy}}\right)$$

$$\text{with } Progress(Z,\ t_{comp})\ =\ 1.$$

The secondary objective is to **minimize the waste**, i.e. the surplus of acquisition of surfaces already acquired during the mission. With the discretization of the area of interest into cells, a mission without waste can be represented as the acquisition of all cells once only. We note the waste

$$Waste(Z,\ t)\ =\ \frac{Nb_{acq}}{Nb_{C_{acq}}}\ -\ 1$$

with:

- Nb_{acq} the total number of cell acquisitions in the Z area from the beginning of the work to the moment t,

▷ and $Nb_{C_{acq}}$ the number of cells of Z acquired at least once from the beginning of the mission to the moment t. For example, if all cells of Z were acquired only once, $Nb_{acq} = Nb_{C_{acq}}$ and $Waste(Z,\ t) = 0$.

5.3 Glimpse System

Multi-agent systems utilize small programming modules called **agents** that are able to perceive and modify their local environment. *AMAS* based systems such as Glimpse are cooperative and as a result define action rules or communication protocols between agents to help each other achieve local goals relative to their environment. The first step in the design of Glimpse is the **identification of agents** among system entities. This is explained further with agent interactions and cooperation design choices. Finally the Glimpse scheduler coordinating agents with real-time events is described step by step.

5.3.1 Agentification and Motivations

In section 5.2.1 we established a new meshing technique called dynamic meshing. When using dynamic meshing a precomputed grid of elementary subzones called cells covers the whole *AOI*. Meshes can then be placed and resized on the cell grid. A simple overview of the problem when formalized with dynamic meshing can be described as the following: an *LARM* receives a *LAC* request, a list of *MPFs* and passes over the *AOI* from their satellites. In a top-down approach the *LARM* must find meshes placed on the cell grid and contained within the corridors of the passes to reduce the rate of unsuccessful acquisition due to cloudy weather and thus optimize problem objectives. This process is presented in figure 5.5. However **designing an *AMAS* requires a bottom-up approach** where the behavior of the individual entities of the problem define the emergent behavior of the system overall. The agentification process thus starts from the micro level and gradually goes up to macro levels.

Agents. Based on this overview we identify system entities that have goals and could make decisions using their local environment to try and achieve these goals. First the **cells and meshes share the goal of being acquired and validated** during any pass. The **goal for each pass** can be described as **achieving the best ratio of surface validated by surface sent for acquisition** to the corresponding *MPF*.

Having identified these entities as agents of Glimpse we refer to them as ***Cell***, ***Mesh*** and ***Pass* agents** in the following. *Cell*, *Mesh* and *Pass* agents need to cooperate to find the best positioning and size for meshes. For example *Cell* agents provide information and send coverage requests to *Mesh* agents and *Pass* agents create and destroy *Mesh* agents based on their criticality. Figure 5.6 summarizes the interactions between the three types of agents.

Criticality. In order to guide these cooperative interactions we define the **agent criticality**. The criticality of an agent represents its importance relative to the other agents of the system.

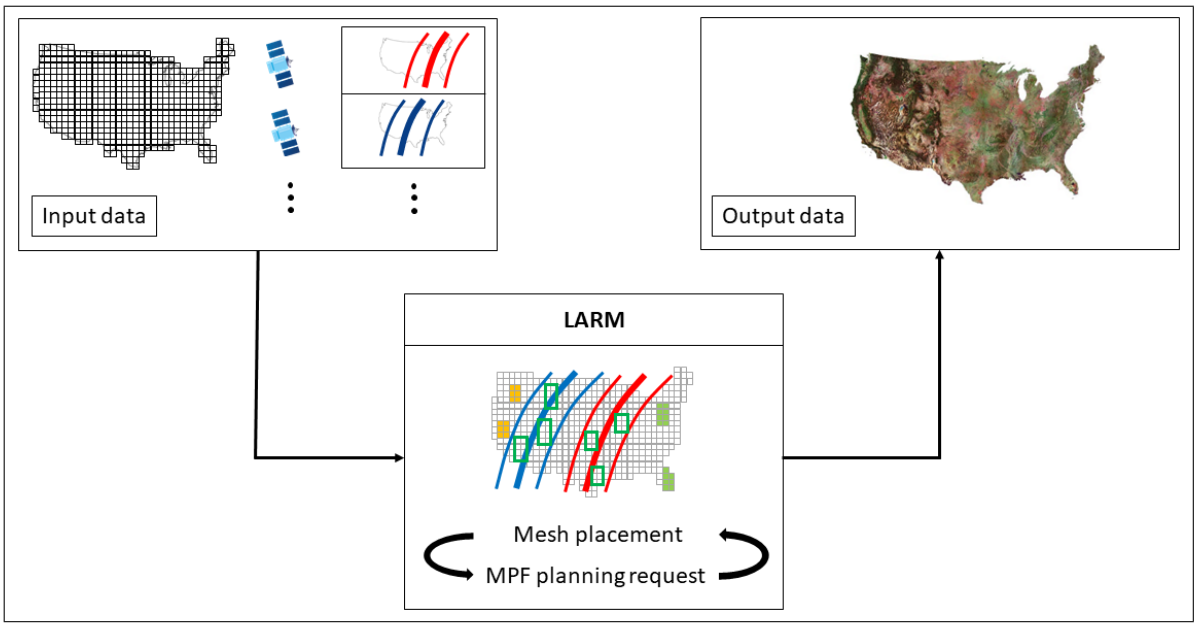

Figure 5.5: Treatment of a large area acquisition request by an *LARM* using dynamic meshing

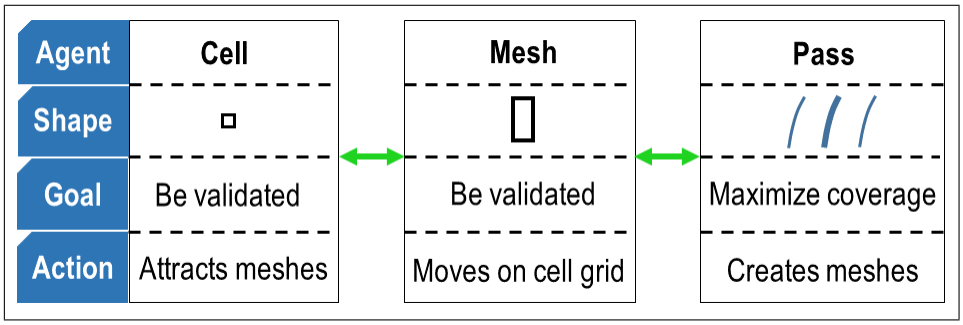

Figure 5.6: Organization of Glimpse agents: *Cell*, *Mesh* and *Pass* agents

The highest the criticality of an agent is the more prioritized it is by other agents and the more likely its local goals are to be fulfilled.

Each agent has its own local goal but in most cases in Glimpse only a subset of the agents relative to a pass can have their local goal fulfilled. For example in a pass that contains 100 cells with a limit of 5 meshes of maximum length and width 2x5 only 50 *Cell* agents can achieve their local goal of being validated in the best case scenario. **The agent criticality is used in order to select which agents should have their local goal fulfilled**. These criticalities are comprised of several criteria depending on the current properties and knowledge of the agent. They are subject to change during the life cycle of the agent as their properties are modified by their perception either direct or by message updates from other agents. **Agent criticalities are used during the decision phase of the agents to decide their action**. Agent interactions are thus based on criticalities and the messages exchanged are a result of their own decision process.

In the following, each agent is described with its representation of the system, knowledge of its environment, its local goals and eventual criticalities. Cooperation processes

between agents are then explained and illustrated through elementary interactions.

5.3.1.1 *Pass* Agents

Internal Representation. *Pass* agents are *Service* agents that represent both the temporal and geographical aspects of a pass. A *Pass* agent knows the characteristics of the satellite performing the acquisitions, the coordinates relative to its corridor (the sub-zone of the large area available for acquisition) and the anticipated time of start and end of the satellite flight over its corridor. It also knows the list of cells contained within the pass and the list of meshes that are available for acquisition on this pass. This list of meshes can vary in size as dynamic meshing allows for the creation and removal of meshes. **The goal of a *Pass* agent is to maximize the surface validated after the acquisition of the corresponding satellite**.

A visual representation of the characteristics of a *Pass* agent is represented as an agent diagram in figure 5.7. An agent diagram presents the internal elements of the agent with its variables and methods in the upper section of the diagram. Linked to the agent section are the perception and the action modules in which are listed the methods available to the agent during the perception and action phases of its life cycle. Finally the agent or agents listed below these modules are the agents with which the agent can communicate through the exchange of messages.

Behavior. The behavior of *Pass* agents is an iterative and correcting mesh creation process. ***Pass* agents first need to determine initial placements for *Mesh* agents and create them on these positions**. The method *manageMeshCreation()* is used by *Pass* agents to decide how many meshes need to be created and their initial placements. The meshes created are initially only as large as a cell, with length and width of one by one relative to the cell grid.

As cooperative agents, *Pass* agents create *Mesh* agents to help *Cell* agents with the highest criticality on the pass, to satisfy their goal. Thus, *Pass* agents select individual cells scoring the highest criticality as initialization coordinates for each newly created *Mesh* agent.

Mesh agents then interact cooperatively with the *Cell* agents following a process called **adhesion** to expand and improve their placement in the corridor. Once this process stabilizes (i.e. no more messages are emitted between *Cell* and *Mesh* agents), meaning no movement or expansion of the meshes can happen further, and the *Pass* agent is informed. The *Pass* agent then uses the *checkStabilization()* and *manageMeshCreation()* methods to ensure that no *Mesh* agent is trapped in local minima. In other words, there is no other part of the corridor that contains more critical cells than those already included in the currently placed meshes. Indeed, as high critical cells are often centralized on a small zone inside the corridor of the pass, this zone can be smaller than the areas covered by all the meshes initially created by the *Pass* agents. As a result meshes may travel from their initial placements to zones of lower criticality than the cell on which they were created initially. In which case, they can be trapped in local minima.

Each *Pass* agent is able to act cooperatively and allocate the resources (the meshes in its corridor) to the remaining most critical *Cell* agents that would not be acquired at the time of stabilization. **Starting from the most critical cells and in descending order, *Pass* agent**

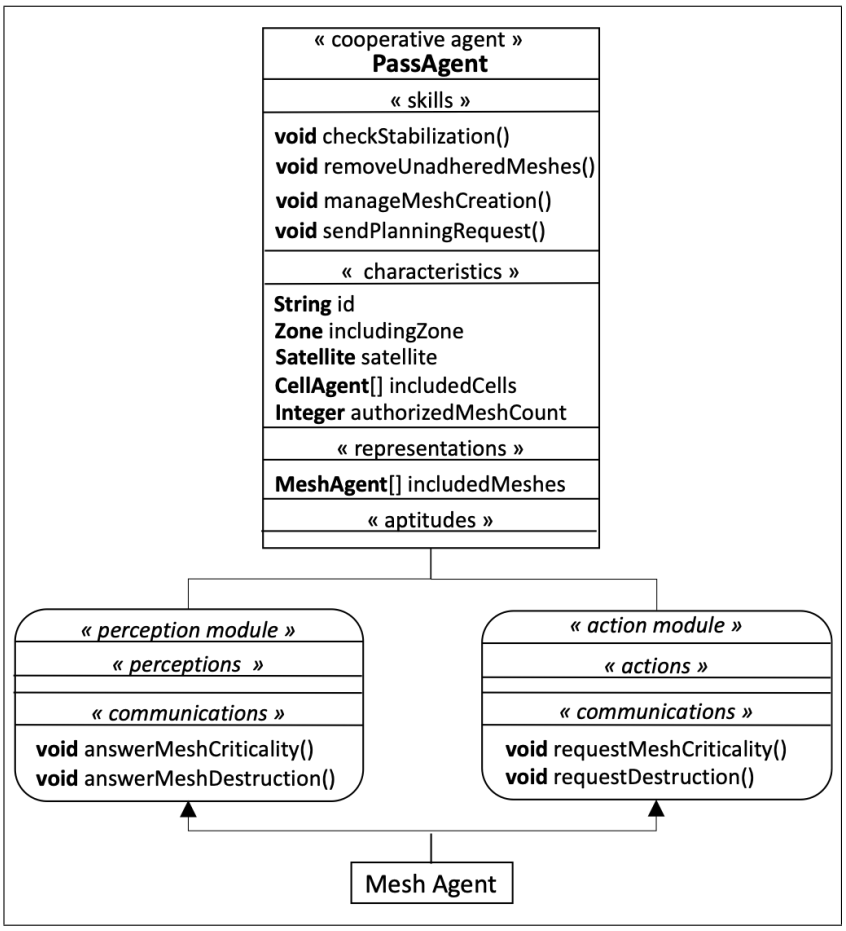

Figure 5.7: *Pass* agent diagram

removes and recreates meshes on the most critical cells not covered by a mesh. This is performed for all cells in the corridor of a pass. The following property is verified at the end of this process: no cell exists on the corridor more critical for this pass than the most critical cell adhering the least critical mesh. At which point removing and creating a mesh again would lead to its creation on the same cell it already covered with no progress made. The comparison to the most critical adhering cell of a mesh is tied to the criticality of mesh agents, further details are given in the *Mesh* agent description. Algorithm 1 summarizes in pseudo code the *manageMeshCreation()* method, with $C_{p_{jk}}$ the cells included in the corridor of the pass p_{jk} of the satellite s_j.

The behavior of *Pass* agents ends when no mesh creation is needed after a stabilization. The condition fulfilled then is that no single cell not adhering to a mesh in the corridor is more critical than the most critical cell of the least critical mesh. The sequence diagram of figure 5.8 illustrates the interactions between a *Pass* agent and the *Mesh* agents in its corridor after the pass stabilizes. In a sequence diagram such as this one the agents are represented as generic entities following a sequence in time from top to bottom. Internal methods are noted as arrows looping back on themselves while messages are noted as arrows directed towards other entities. The *opt* operator corresponds to a conditional statement where the section of the diagram is executed only if the boolean variable checked is verified as being

Algorithm 1 *Mesh* creation algorithm ((manageMeshCreation() method)

```
1: sorting of cells c ∈ C_{p_jk} by criticality
2: for all cell c_xy ∈ C_{p_jk} do
3:   if c_xy not covered and highly critical compared to the covered cells in the corridor then
4:     creation of a mesh m = (x, y, 1, 1)
5:     if maximum number of meshes threshold exceeded then
6:       removal of the least critical mesh
7:     end if
8:   end if
9: end for
```

true.

In figure 5.8, the *Pass* agent first checks the condition for stabilization on the pass. If it is verified it gathers all updated mesh criticalities using the *requestMeshCriticality()* service of the corresponding *Mesh* agents and receives the answer through the *answerMeshCriticality()* method. If the *Pass* agent detects any *Mesh* agent without any *Cell* agent adhering to it (*removeUnadheredMeshes()*), it sends destruction messages through *requestMeshDestruction()* to these *Mesh* agents that remove themselves from the system using *destroySelf()* method. This cleaning phase is only performed after pass stabilization as *Mesh* agents can have temporary states without any cell adhesion during the self-organization process followed by cell adhesion in later agent cycles. The adhesion process is explained further in section 5.3.2. The *Pass* agent then removes and recreates meshes using the method *manageMeshCreation()*.

After this is done, if no meshes have been created the mesh placement on this pass is considered complete and *sendPlanningRequest()* is used by the *Pass* agent to send the list of meshes to acquire to the *MPF* following a schedule presented in section 2.4. Indeed, if no meshes are created the absence of cells justifying the process of removing and creating new meshes is verified. If meshes are created the self-organization process between *Mesh* and *Cell* agent starts and the *Pass* agent continues at step 1 in the sequence diagram.

***Pass* Agents and Adaptability.** As an algorithm following the *AMAS* theory, Glimpse is a **single-solution** algorithm. In this type of optimization algorithm, **a single candidate solution is iterated upon** and improved to tend towards the global optimum. This is in opposition to **population-based** algorithms [Beheshti and Shamsuddin, 2013]. At any point in time (which can be represented as an agent cycle) during the solving of the *LAC* problem, Glimpse agents have a memory of their neighborhood, characteristics and state of planning. Single-state algorithms such as greedy algorithms often require a new launch and a total reset of the parameters when the data of the problem is modified. For the *LAC* problem we need to consider the possibility of data modification when weather forecasts are updated.

In this case a greedy algorithm applied to the *LAC* problem needs to be reset as the decisions early in the algorithm have a strong impact on the decisions made later in the solving. This is specifically true for greedy algorithms as no backtracking is allowed in the decision tree, but is also true for most single-solution algorithms such as *SA* algorithms

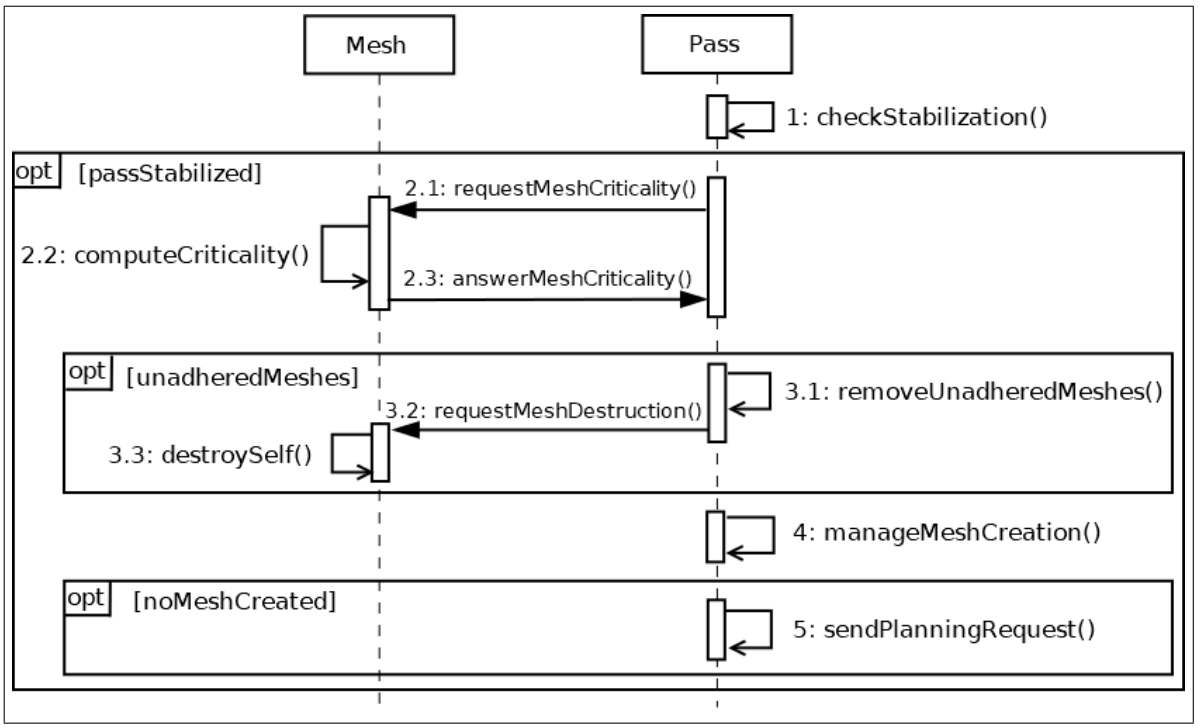

Figure 5.8: Sequence diagram of the interactions between a *Pass* and *Mesh* agents

as the shape of the objective function may have changed and present new local or global optima late in the solving process. ***AMAS* algorithms are more resilient to updates** during the solving as **they can keep the existing parts of the solution that are not affected by the updates** and only have the affected agents react to the change.

In the case of Glimpse, the behavior of *Pass* agents ensures that only the *Mesh* agents that are covering *Cell* agents which had their criticality worsened by an update are removed and other meshes created elsewhere on the corridor. Existing *Mesh* agents that are still covering critical *Cell* agents may move or extend to adapt to the new criticality of neighboring *Cell* agents but will not be recreated, which avoids restarting the computation of mesh placements in the corridor from zero. This process is illustrated in figure 5.9 using the same corridor through time and subareas of different criticality represented from light red (least critical) to bright red (most critical).

5.3.1.2 *Mesh* Agents

Internal Representation. *Mesh* agents are *Service* and *Constrained* agents that are the system representation of meshes. A *Mesh* agent has origin coordinates, length and width, a maximum size and a state of acquisition of its corresponding mesh. It knows the pass in which the mesh it is representing is contained and the list of cells that are covered by the mesh or adjacent to it within the pass. **The local goal of a *Mesh* agent is for the corresponding mesh to be validated**. To do so, the *Mesh* agent seeks to increase the likelihood of its corresponding mesh to be validated by covering the cells with the best chance of validation (highly critical cells). Thus, a *Mesh* agent is able to communicate with the cells covered by

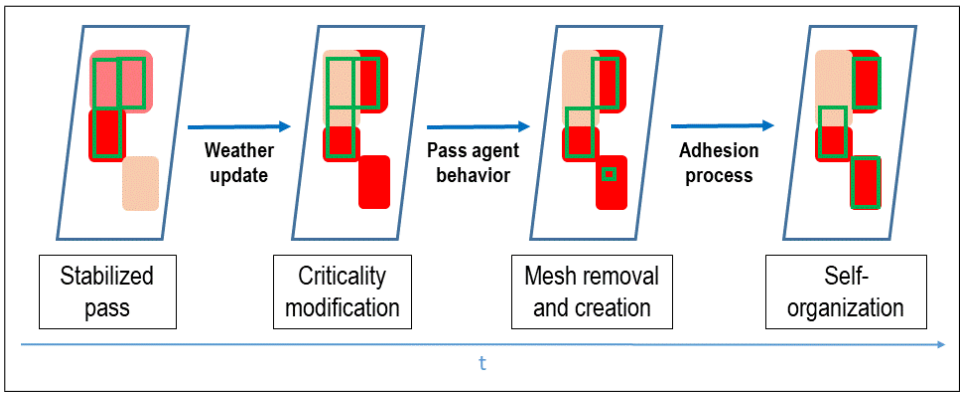

Figure 5.9: Adaptation of Glimpse agents to a weather update through creation of meshes and self-organization

the mesh, adjacent cells within the pass, and the pass in which it is contained. The local neighborhood of a *Mesh* agent consists of the agents representing the cells that are covered or neighboring the mesh and the pass on which the mesh is placed. *Mesh* agents are the only agents to interact with both *Cell* and *Pass* agents as presented in the agent diagram of figure 5.10.

Criticality. *Mesh* agents have a criticality based on a list of hierarchical criteria. This criticality represents the **importance of acquisition of the mesh on the pass** (their likelihood to be validated),and is thus relative to the cells covered by the mesh. To ensure that at least the most critical *Cell* agent covered by the mesh and willing to stay covered by the mesh has its local goal fulfilled, we based the criticality of a *Mesh* agent on the criticality of this *Cell* agent. The notion of a *Cell* agent willing to be covered by a *Mesh* agent is described as **adhesion** and explained in section 5.3.2. The criticality of a *Mesh* agent is based on the following normalized criteria:

1. **Adhering *Cell* agents.** The criticalities of the *Cell* agents currently adhering to a *Mesh* agent are the defining criterion for the criticality of the *Mesh* agent. The criticalities of the adhering *Cell* agents are listed in descending order. Comparison of *Mesh* agents with this criterion implies the comparison of the respective most critical adhering *Cell* agents two-by-two and in descending order, until a more critical adhering *Cell* agent is found. **The *Mesh* agent with the first more critical adhering *Cell* agent is considered more critical in the comparison**. In the case of equality but different numbers of adhering *Cell* agents, the *Mesh* agent with the most adhering *Cell* agents is more critical. In the case of equality and same number of adhering *Cell* agents the comparison is considered equal.

2. **Random.** Pre-generated value affected to each agent at creation to avoid oscillation if computed at each comparison. Used only if an equality is found when comparing the first criterion.

Behavior. The behavior of a *Mesh* agent consists of **improving its situation by expanding and moving the mesh on the cell grid**. Each *Mesh* agent communicates with its covered

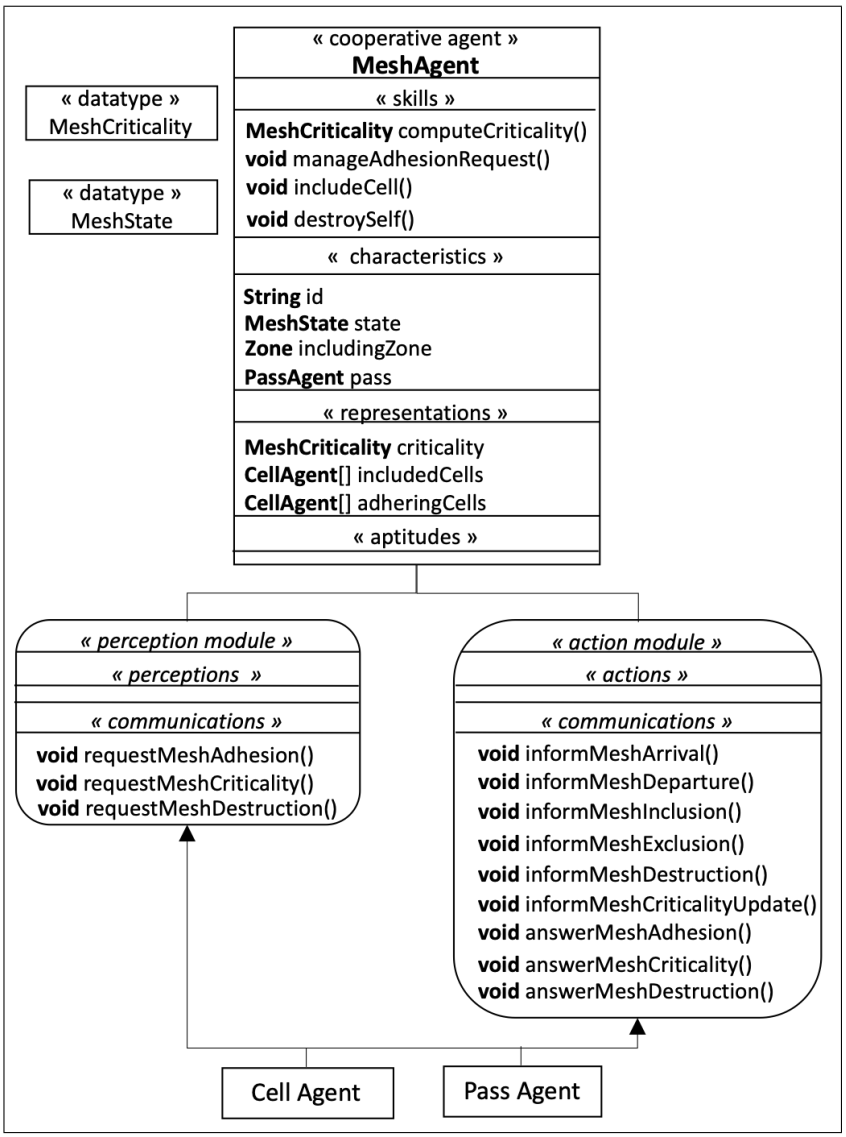

Figure 5.10: *Mesh* agent diagram

and neighboring *Cell* agents and follows the adhesion process (section 5.3.2) to decide if it should expand or move. At pass stabilization the *Mesh* agent answers the request of its relative *Pass* agent for its criticality. If the condition for planning is satisfied and the *Mesh* agent is included in the meshes sent for planning its internal state changes to **acquired**. After the delay necessary for the acquisition and answer from the corresponding *MPF*, the *Mesh* agent state becomes either *validated* or back to *active* for future passes.

Figure 5.11 presents the states and transitions of a *Mesh* agent. In this type of state diagrams a specific variable of the agent goes through different states depending on certain conditions. In this example the states available for the variable *state* of a *Mesh* agent (figure 5.10) are presented. The transitions between states are dependent on a conditional statement being verified as true for leaving the state. The modification of internal parameters and reception of messages during the behavior of the agent are possible causes for state transitions. As the behavior of *Mesh* agents is tied to the one of *Cell* agents, they are described jointly in

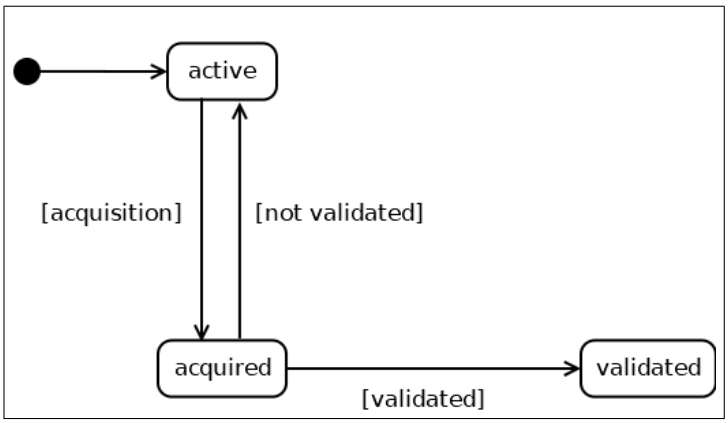

Figure 5.11: States of a *Mesh* agent and conditional transitions

section 5.3.2.

5.3.1.3 *Cell* Agents

Internal Representation. *Cell* agents represent the geographical zone covered by a cell in the *LAC* problem. A *Cell* agent has coordinates and knows the state of acquisition of its corresponding cell. It also knows the passes on which a satellite can potentially acquire the cell. For each of these passes, the cell agent knows the list of meshes that are either covering or adjacent to the cell. **The local goal of a *Cell* agent is for the corresponding cell to be validated**. For a cell to be validated an acquisition of a mesh covering the cell must be validated on any pass. To achieve validation a *Cell* agent is able to send messages to *Mesh* agents it knows. These agents represent the meshes previously mentioned and form the local neighborhood of the *Cell* agent. The agent diagram in figure 5.12 highlights the properties of a *Cell* agent and its modules for interacting with *Mesh* agents.

Criticality. The criticality of a *Cell* agent for a pass represents the **importance of acquiring the corresponding cell on this pass**. It is computed following a lexicographic order of criteria. To find the most critical of two *Cell* agents each criterion is compared one by one sequentially with the first inequality guiding the choice. For each criterion a place in this hierarchical list needs to be found representing their importance of acquisition. The criteria and their order in the criticality of *Cell* agents has been determined by experts of the research field and experimental results. *Cell* agents have a separate criticality for each pass on which they could be acquired. Indeed the criteria may change based on the local knowledge of the agent on this pass. A *Cell* agent could be very critical on a pass for which its weather is favorable and less critical on a pass with unfavorable weather for example. The criticality of *Cell* agents in a given pass is based on the following normalized criteria:

1. **Validation.** The criticality of a cell that has already been validated is null.
2. **Weather.** Probability of validation of the cell if acquired in this pass based on weather predictions. For cells also acquired in previous passes this criterion is weighted by their probability of validation in these passes.

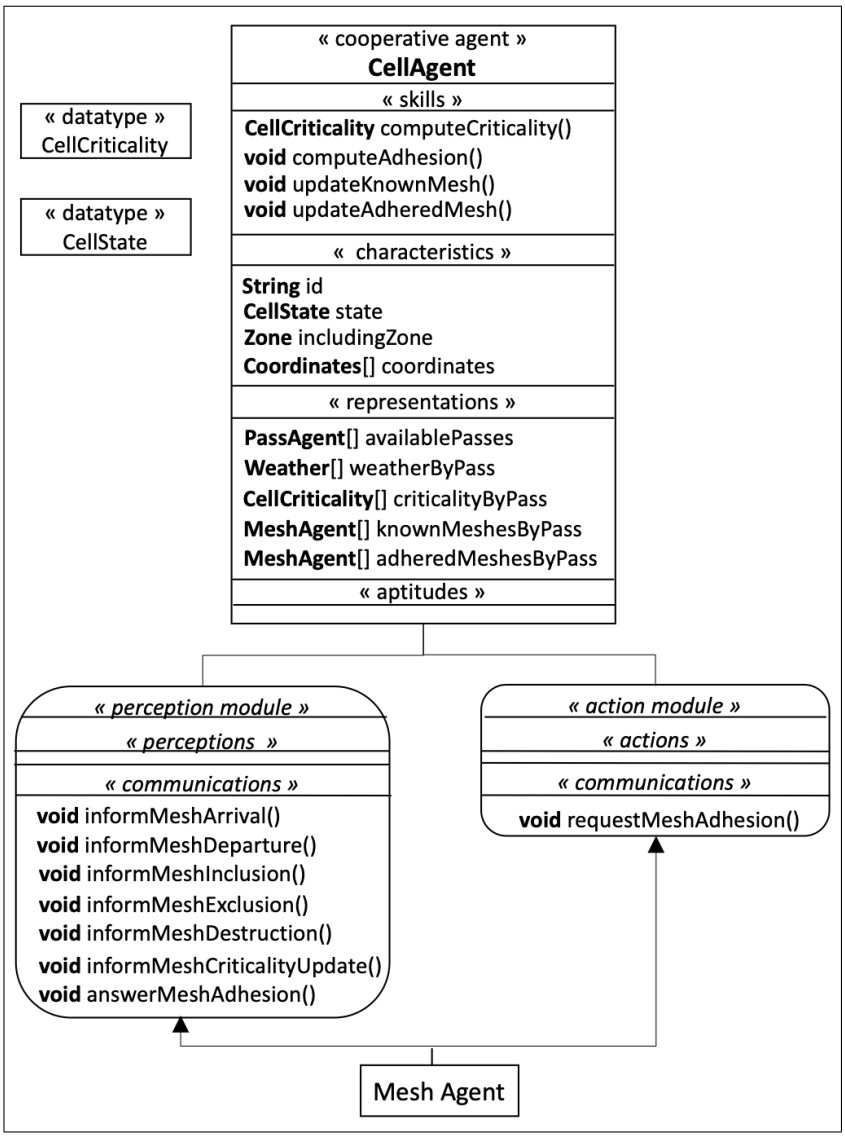

Figure 5.12: *Cell* agent diagram

3. **Anticipated weather.** Probability of validation of the cell in the remaining passes based on weather predictions on the planning horizon for which the corridor covers this cell. A higher chance of being acquired in future passes lowers the cell criticality.

4. **Available passes.** Ratio of remaining passes on the planning horizon for which the corridor covers this cell. Represents the future acquisition opportunities of the cell.

5. **Grid neighbor ratio.** Ratio of neighbouring validated cells. Cells with a high concentration of neighboring validated cells being more critical helps with the reduction of small gaps between validated areas on the grid. The acquisition of such gaps leads to reacquisition of validated cells and opportunity costs for the acquisition of meshes with more non validated cells. An example of gaps that the grid neighbor ratio criterion tries to solve is illustrated in figure 5.13.

6. **Random.** Pre-generated value affected to each agent at creation to avoid oscillation if

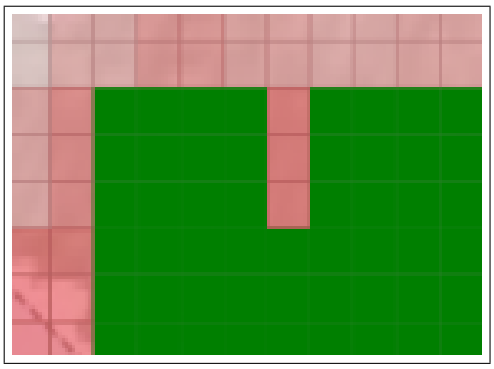

Figure 5.13: Gap between validated areas on the cell grid

computed at each comparison.

In this section and the experiments of section 6, the criticality of *Cell* agents follows the order presented above. Different criticality criteria and orders affect the behavior of Glimpse as different cells are prioritized during the acquisition. A more in-depth analysis of the importance of criticality criteria of *Cell* agents and their order is presented in section 6.5.

Behavior. The behavior of a *Cell* agent is one of a *Constrained* agent in *AMAS4Opt*. It consists of **observing its covering and neighboring meshes** and sending messages to either **adhere to a covering mesh** or **ask for the movement of a neighboring mesh** in order to adhere to it once it moves to cover the cell. **A *Cell* agent has a separate planning state for each pass that includes it in its corridor**. The states of a *Cell* agent are dependant of the coverage and adhesion status when active. A *Cell* agent can only be validated on a pass on which it is covered. If it is covered, it can adhere to a single mesh on this pass. If a mesh covering the cell on the pass is validated, the state of the corresponding *Cell* agent changes to become *validated* for all passes that include it in their corridor. Figure 5.14 presents the states of a *Cell* agents and the conditional transitions between states. An important part of the *Cell* agent behavior is its interaction with the *Mesh* agent through the adhesion process described in section 5.3.2.

5.3.2 Adhesion Process

The goal of the adhesion process is to satisfy the local goals of both *Cell* and *Mesh* agents. A *Cell* agent adhering to a *Mesh* agent on a pass represents its will to remain covered by the relative mesh in the corridor. **A *Cell* agent can adhere to a single *Mesh* per pass but a *Mesh* agent can have several *Cell* agent adhering to it**. By default a *Cell* agent always adheres to a *Mesh* agent if this *Mesh* agent is the only one the *Cell* agent knows for a pass and if the mesh covers the cell for this pass. In the case of several meshes covering the same cell, the *Cell* agent observes which *Mesh* agent is more critical and adheres to it. This satisfies its local goal as being covered by a highly critical mesh which improves its likelihood of being acquired on this pass.

The adhesion of cells is used to guide the behavior of *Mesh* agents on the cell grid. As seen in section 5.3.1.1, meshes are initially created by *Pass* agents with the smallest possible

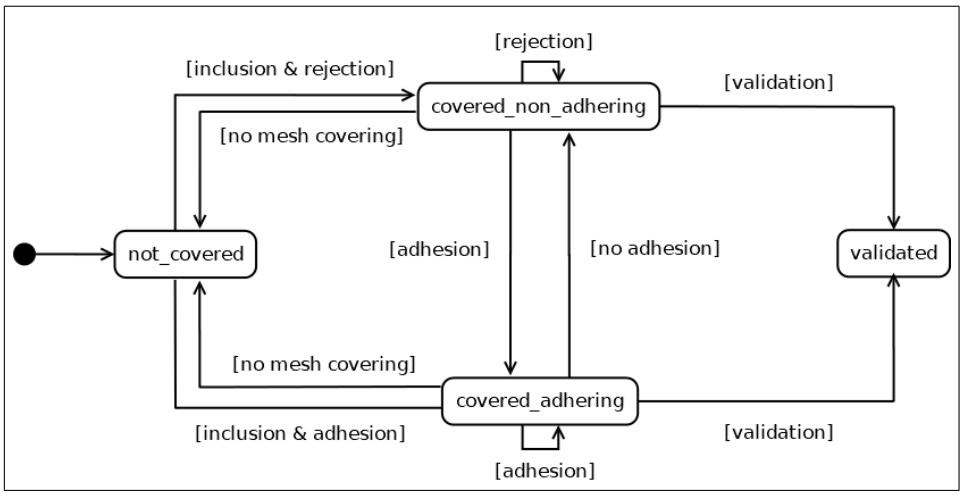

Figure 5.14: States of a *Cell* agent and conditional transitions

size of width and length one by one relative to the cell grid. After their creation *Mesh* agents receive adhesion requests from covered and neighboring *Cell* agents. The already covered cell is considered adhering if it emitted an adhesion request message to the mesh agent that acknowledges it back. **To include a *Cell* agent that requests adhesion, a *Mesh* agent may need to adjust its geometry and expand or move on the grid**. By default a *Mesh* agent always expands until its maximum width or height has been reached, at which point only a movement is possible to include neighboring cells. All mesh movements are made by moving the mesh in a cardinal direction by a distance of a cell which equals to shifting one of its origin coordinates by one on the cell grid.

Mesh Expansion and Movement. In the case of **expansion**, the *Mesh* agent updates its geometry and informs all the newly covered and newly neighboring *Cell* agents of its new status. Covered *Cell* agents can then observe their neighborhood and, if the *Mesh* agent is still the most critical one that is covering them, inform it of their adhesion.

In the case of **movement**, a Conflict NCS is reached where moving the mesh in a cardinal direction implies that the mesh will cover a new strip of cells, the **destination strip**, and that it will no longer cover a strip of cells on the opposite direction, the **origin strip**. To solve the NCS the *Mesh* agent compares the criticality of the most critical cell on the destination strip with the criticality of the most critical cell on the origin strip. If the destination strip has the most critical cell the movement is performed as it improves the criticality of the *Mesh* agent. If the origin strip has the most critical cell the adhesion request of the cell on the destination strip is rejected and the mesh does not move.

Figure 5.15 shows a *Cell* agent sending an adhesion request to a neighbouring *Mesh* agent and the result of an inclusion of the cell by expansion of the mesh.

Algorithms 2 and 3 respectively present the *computeAdhesion()* method of a *Cell* agent, and the *manageAdhesionRequest()* method of a *Mesh* agent, executed during the *decide* phase of the agents. The messages generated during the *decide* phase are then sent during the *act* phase.

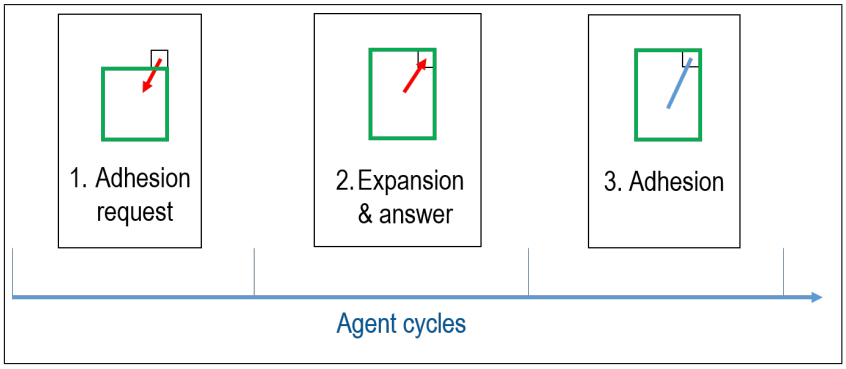

Figure 5.15: Adhesion request of a *Cell* agent to a neighbouring *Mesh* agent and expansion of the mesh

Algorithm 2 *Cell* agent adhesion process (computeAdhesion() method)

for all known p_{jk} **do**
  sorting of known meshes $M_{p_{jk}}$ by criticality
  for all $m_{p_{jk}} \in M_{p_{jk}}$ **do**
    if non adhered cell and no adhesion request **then**
      mesh adhesion request to $m_{p_{jk}}$
    end if
  end for
end for

Algorithm 3 *Mesh* agent adhesion process (manageAdhesionRequest() method)

for all adhesion request of c_{ij} **do**
  if c_{xy} is covered by the mesh **then**
    adhesion request acceptance
  else
    if c_{xy} is north or south of the mesh and mesh length limit is not reached **then**
      expansion towards c_{xy}
      adhesion request acceptance
    else
      if c_{xy} criticality is higher than the most critical adhering cell of the opposite mesh strip **then**
        movement towards c_{xy}
        adhesion request acceptance
        inform opposite cells of exclusion
      else
        adhesion request rejection
      end if
    end if
  end if
end for
reduction of edges without adhesion

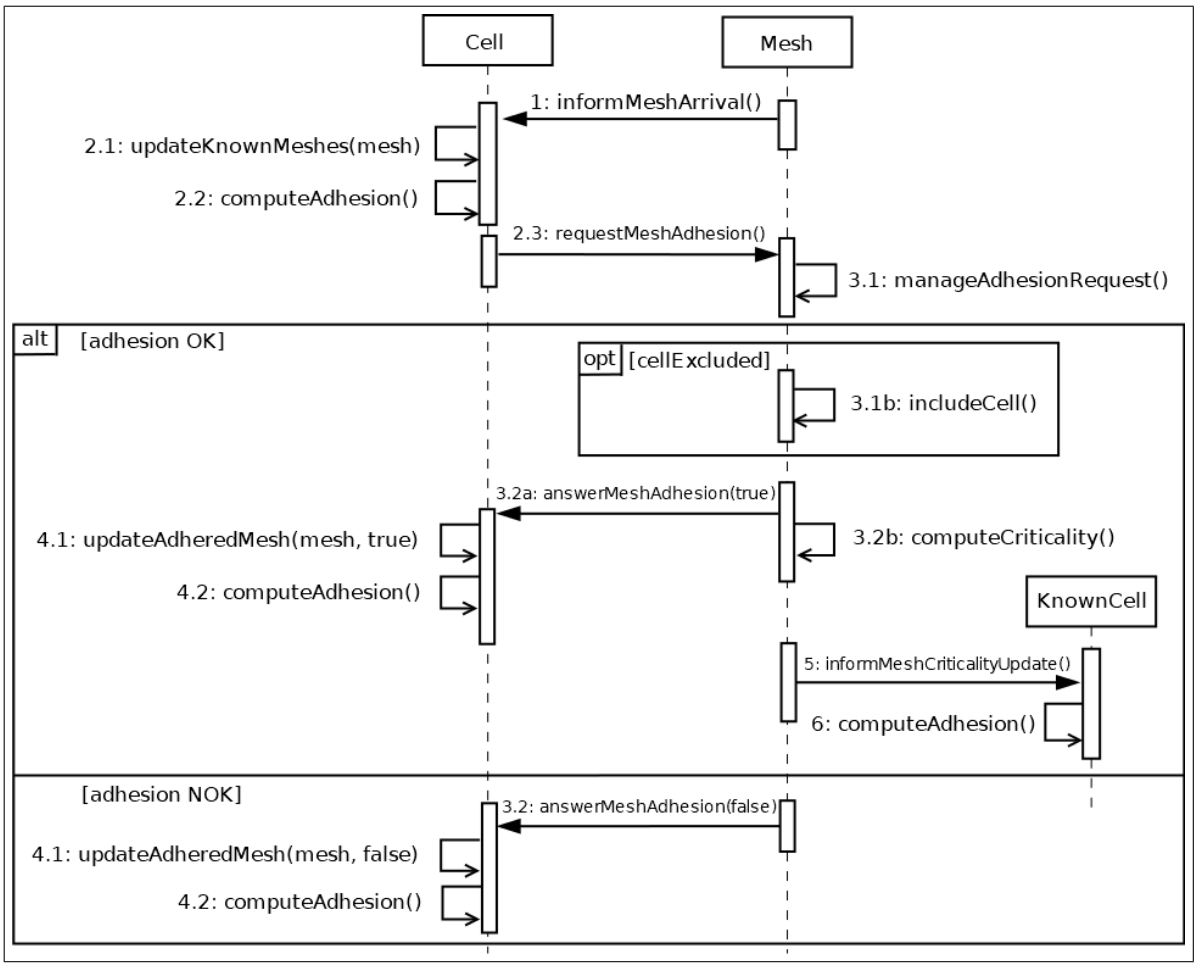

Figure 5.16: Sequence diagram of the interactions between *Cell* and *Mesh* agents during mesh arrival

The sequence diagram of figure 5.16 presents the arrival of a mesh near a cell, either by creation or movement, and the potential messages between *Cell* and *Mesh* agents. The *alt* operator corresponds to alternate states reached during the execution depending on the verification of a boolean variable. The internal methods and messages used in figure 5.16 are also present in the agent diagrams of figures 5.10 and 5.12.

In the sequence diagram of figure 5.16 the mesh is either created next to or on the cell, or moved or extended itself towards the cell during the previous agent cycle. Using the method *informMeshArrival()* a first message is sent by the *Mesh* agent to the *Cell* agent. The *Cell* agent updates its internal representation of its neighborhood. It then computes the benefit of adhering to the newly arrived mesh following algorithm 2 (*computeMeshAdhesion()* method). In this diagram ,the case in which the *Cell* agent requests to adhere to the *Mesh* agent is considered. The *Cell* agent then answers with a *requestMeshAdhesion()* message to the *Mesh* agent. Algorithm 3 (*manageAdhesionRequest()*) is then executed by the *Mesh* agent to determine the adhesion of the requesting cell.

If the adhesion of the cell is accepted by the *Mesh* agent but the cell is outside of the mesh, the mesh first adapts its geometry to include the cell. The *includeCell()* method of the *Mesh* agent determines the type of the required operation, either expansion or movement,

and performs this operation on the mesh. The *Mesh* agent then answers the adhesion request with a *answerMeshAdhesion(true)* message, updates its criticality and informs all the cell of its neighborhood of its new criticality. Like in the case of a mesh arrival, this criticality update may trigger new adhesion processes from these cells. The *Cell* agent that receives the acceptation of its adhesion request, updates its internal adhesion representation. It then checks if the *Mesh* agent to which it is now adhered to has the best criticality in its neighborhood. If a more critical *Mesh* agent is found a new adhesion process is started by the *Cell* agent.

If the original adhesion request of the *Cell* agent was instead rejected the *Mesh* agent answers with a *answerMeshAdhesion(false)* message. In this case the *Cell* agent marks the *Mesh* agent as having refused its request and enters a rejected state relative to this *Mesh* agent. **In this state the *Cell* agent do not send further adhesion request to the *Mesh* agent**. The *Cell* agent then uses the *computeMeshAdhesion()* method to check for additional adhesion opportunities. The state of rejection of *Cell* agents relative to each *Mesh* agent in their neighborhood is reset in the event of a mesh criticality update or when the mesh leaves the neighborhood of the cell. This allows for criticality updates on the pass or additional cell adhesion to the *Mesh* agent to restart the self-organization process with previously rejected cells.

Adhesion Process Dynamic Adaptation. The goal of the adhesion process is to **allow *Mesh* agents to self-improve on a micro level** after their creation on a macro level by the *Pass* agent. *Mesh* agents always reach their maximum allowed size if their position on the corridor allows it. **The adhesion of *Cell* agent to a *Mesh* agent is not definitive and *Cell* agents are always searching for a better *Mesh* agent to adhere to**, improving their likelihood of acquisition and potentially the likelihood of acquisition of any *Mesh* agent entering their neighborhood.

The adhesion process also helps in solving the problem of mesh overlaps present in both static and dynamic meshing in the case of heterogeneous satellites. The restriction of *Cell* agents to adhere to only one *Mesh* agent per pass helps in spreading meshes over the corridors. Indeed, *Cell* agents adhere to the most critical *Mesh* agent while the less critical *Mesh* agent accepts adhesion requests from *Cell* agents in the opposite direction of the most critical *Mesh* agent. This phenomenon is illustrated in figure 5.17. Overlaps are still allowed and can happen as a single *Cell* agent adhering to a *Mesh* agent can prevent the movement of the mesh in the opposite destination strip.

Adhesion example. Figure 5.18 presents an example of the adhesion process at different steps of an experimental scenario. In this case, meshes are initially created on the least critical cells instead of the most critical cells. The *Cell* and *Mesh* agents then self-organize through the adhesion process. The final result is a meshing pattern covering the most critical cells without overlap which emerged from the cooperation of *Cell* and *Mesh* agents.

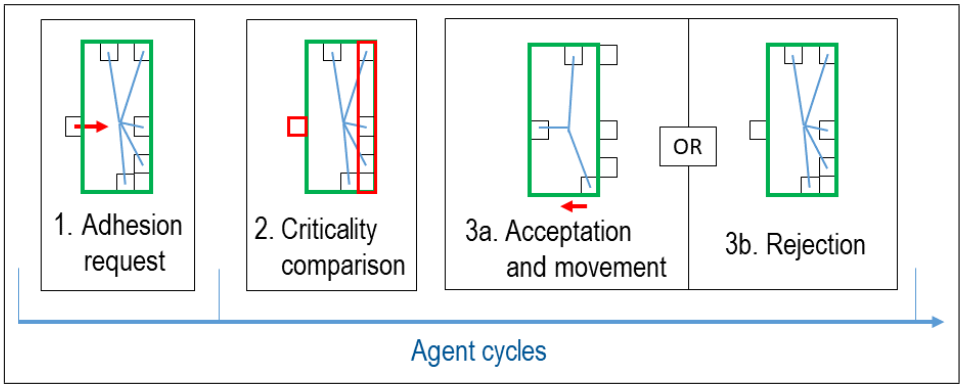

Figure 5.17: Adhesion request and possible consequences on the opposite cell strip of the mesh

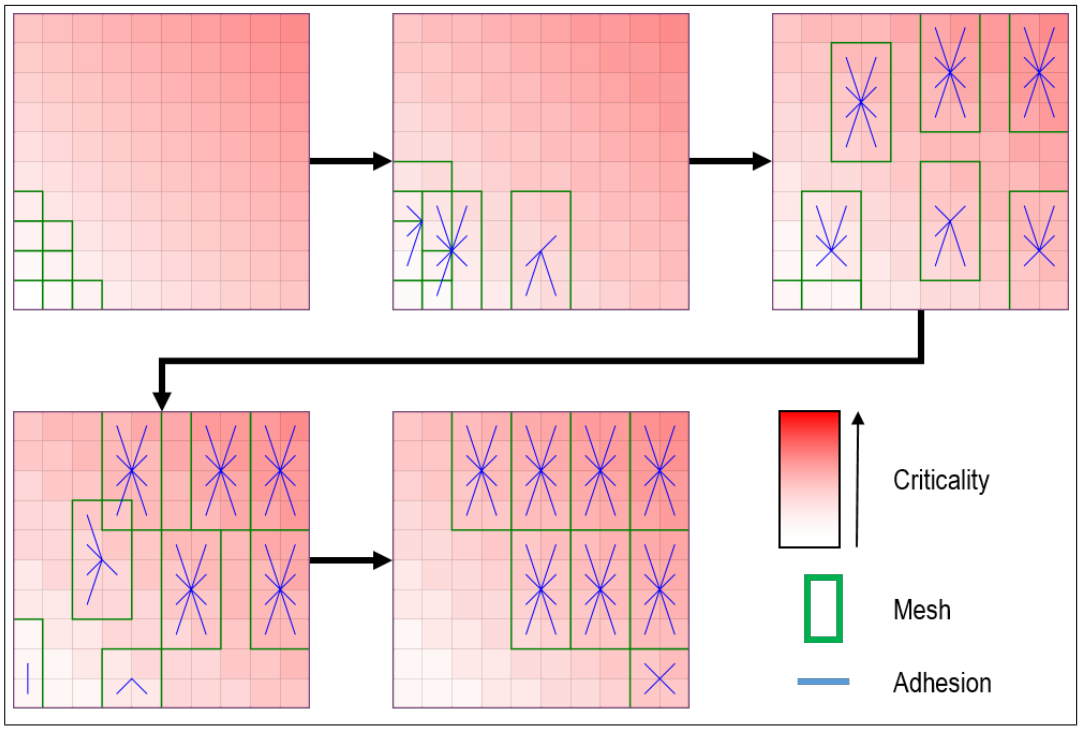

Figure 5.18: Example of the adhesion process between *Cell* and *Mesh* agents at different steps of an experimental scenario

5.3.3 Glimpse as an *LARM*

In order to replicate the process followed by an operational *LARM*, Glimpse simulates the steps of this process from large area request reception to the mission completion. **Passes are created progressively during the mission and are treated chronologically**. During the execution of Glimpse the present is represented by a **current pass**. Future passes up to the planning horizon are created to provide data for the *probability of future validation* criticality criterion of *Cell* agents. Passes in the past can either be labelled as ***acquired*** or ***validated***. An *acquired* status represents passes for which the *MPF* has sent acquisition orders to the corresponding satellite but has not yet received the resulting images. A *validated* status represents passes for which the images resulting from the acquisitions have either been *validated* or *rejected*, and in the case of mesh validation the relevant *Mesh* and *Cell* agents have been updated in Glimpse.

To illustrate the life cycle of a pass in Glimpse we consider the steps from creation to validation. At its creation a pass is part of future passes on the planning horizon, or directly the current pass in the case of the very first pass of the mission. As past passes are *acquired*, the following pass in this example becomes the current pass. The *Pass*, *Mesh* and *Cell* agents then find optimized positions and shapes for meshes up to a number of meshes allowed on the pass. When the pass has reached its final stabilization its status is changed to *acquired*, as is the case for the *Pass* agent and each *Mesh* and *Cell* agents included in the acquisitions, and the next pass in chronological order becomes the current pass.

For each pass acquired a lapse of time t_{acq} is then simulated, corresponding to the estimated time between acquisition orders to the satellite and image reception on a ground station. The pass, of occurrence date t_{occ}, changes its status to *validated* when the date of occurrence of the current pass is further than $t_{occ} + t_{acq}$, representing the reception of *acquired* images by the corresponding *MPF* and transmission to the *LARM*, in our case Glimpse.

The time t_{acq} is one of the mission parameters that form the Glimpse input. Other inputs include:

1. the ***AOI***,
2. the ***MPFs*** and their **satellites characteristics**,
3. a **list of passes** over the *AOI* for each satellite,
4. **weather data** for each pass,
5. a **cell size** in km,
6. a length of **planning horizon** in number of passes.

5.3.4 Conclusion

In this chapter we proposed our contributions. Dynamic meshing is a novel approach to meshing based on a grid of elementary geographical entities called cells. The formalization of the *LAC* problem is updated to incorporate the freedom of mesh placements over of the

cell grid. We then describe Glimpse, an AMAS for the solving of this formalized *LAC* problem. Glimpse is an *LARM* that uses *Cell*, *Mesh* and *Pass* agents to coordinate the meshing and mesh selections over successive satellite passes. Using the respective criticality of *Cell* and *Mesh* agents, the agents of Glimpse self-organize to answer the local goals of the most important agents. The adhesion process between *Cell* and *Mesh* agents is central to this organization as the mechanism driving the expansion and movements of the meshes in the corridors. As a result Glimpse optimizes the solving metrics through an emergent function produced by the cooperation of its agents.

In chapter 6 we present the results of Glimpse obtained on different use cases and using several sets of parameters. We then compare the results with references including those of a greedy algorithm and discuss the differences and implications on the performance of Glimpse.

6 Experimentation

6.1 Introduction

Operationally, *LAC* missions are often optimized either by human expertise or greedy algorithms selecting the most promising meshes. To best compare the dynamic meshing of Glimpse and other contributions to state of the art methods we used operational data from previous *LAC* missions. The results presented in this chapter are obtained by running different algorithms or sets of parameters on these use cases. First we evaluate the **performances** of Glimpse regarding the problem metrics. We define use cases for measuring the performances and describe the parameters used. We then compare the results obtained using a greedy algorithm and static meshing.

To expand on these results we evaluate the **robustness** of Glimpse. Additional use cases are defined with a focus on the variety in the *AOI* acquired and the Glimpse parameters used. The robustness results are then discussed and linked to the choices made during the design of Glimpse.

The goal of the experiments in both cases is to show the **adaptability** of Glimpse and the results in performance and ease of treatment of a variety of scenarios.

Finally, a **sensitivity analysis** regarding the criteria of the Cell agent **criticality** is presented. The impact on the quality of solutions found when using different sets of criteria and in different lexicographic orders is discussed.

6.2 Experimental Environment

6.2.1 Data

Input data for Glimpse includes *MPF*, satellite and pass characteristics that are coherent with the data used operationally to perform relevant experiments in this field of research. While these data can be recreated or simulated, the industrial partners of this thesis allowed us to use data from previous *LAC* missions as a base for experimentation. This gives us all mission parameters, which can be modified to allow tests of different use cases and monitor the behavior and results of Glimpse. For the weather during our test missions we used the observed weather at the time of the real mission as predicted weather. This simulated

weather prediction allowed us to use a realistic weather model for observation missions.

6.2.2 Weather

Weather is simulated using old weather observation data as previsions. An observation value is drawn over a normalized distribution around the weather prevision. A prevision consists of a value on this distribution obtained with inverse probability to its distance from the observation value. The observation value remains the same as the original observation data. In order to account for outliers in the global weather simulation during a mission, all scenario results presented in this chapter are averages over a sample of 100 runs of each tested algorithm. Standard deviation on these results is low as the large size of the *AOI* and number of passes smooth outliers over time. This also reduces the randomness inherent to a Glimpse execution due to the presence of the random criterion presented in section 5.3.1.3 in the *Cell* agents that acts as a decider between identical agent comparisons.

6.2.3 Metrics

As presented in section 3.3.1 two different metrics are considered for the *LAC* problem: the time required to complete the mission and the waste of satellite resources. For all experiments in this chapter both **completion time and waste** are considered as determinant factors to judge the quality of the obtained results.

Completion time. Completion time is measured as the number of passes necessary to reach 95% validated surface of the *LAC*. Indeed, at the end of most scenarios a common occurrence is the difficulty of validating the remaining 5% of the *AOI* as it is often the most clouded and most difficult area to acquire during the rest of the mission. A **bottleneck effect** can be created where every resolution algorithm needs to wait for a few favorable passes to occur in order to finish the mission to 100%.

These bottlenecks reduce or negate the differences in acquisition performances as less efficient algorithms have time to catch up to better algorithms during the waiting periods. To avoid this phenomenon near the end of the *LAC* mission **95% completion** is chosen as the threshold of estimated completion time when measuring performances. This is also justified by industry practices where **an *LARM* may send preliminary updates** to the clients with parts or most of the *AOI* already validated. The remaining 5% are then sent at a later date.

Waste. Waste is measured in percentage of surplus acquisition of surfaces during the mission. 100% waste corresponds to acquisitions having covered twice the *AOI* during the mission. The superfluous acquisition of a mesh is registered as waste whether the mesh was rejected by the *MPF* or in the case of clouded acquisition. **In this study we consider the waste to be a result of clouded acquisitions only**. Indeed, this provides more accurate results for the prediction abilities of the meshing algorithms as the waste is only dependant on the placement of meshes and **independent of an internal and unknown *MPF* planning method**.

For both metrics the results are observed at the end of the mission, defined as 95% acquisition of the *AOI*, but also during mission progression. As mentioned previously updates may be sent to the client during the mission, making intermediary results relevant to the industry process. The evolution of completion and waste on graphs during the mission also help to explain the differences between approaches or use cases employed.

6.2.4 MPF Simulation

As presented previously, an *LARM* such as Glimpse can be related to the *MPFs* controlling the satellites. In this study we consider the more generic approach in which ***MPFs* are independent and unrelated to the *LARM***. This implies that each *MPF* has its own planning algorithm and may select any number of meshes from the ones selected by the *LARM* to acquire. To simulate this, an *MPF* is implemented as a module consisting of a planning algorithm and with which Glimpse can interact. For the purpose of simplification, each *MPF* is considered as having a planning algorithm that **acquires every mesh selected for acquisition** by Glimpse. This behavior can be modified to reduce the number of meshes planned by the *MPF* as to simulate heavy load or more urgent requests being prioritized by the *MPF*.

6.2.5 Greedy Algorithm

In the *LAC* problem as defined in section 5.2.2 the goal of the resolution algorithms is to find for each pass a set of meshes to include in an acquisition request for the *MPF* responsible for this pass. A greedy approach to this goal is to select the best mesh relative to its criticality and include it in the acquisition request. This selection is then never reevaluated as per the greedy methodology. Selected cells on the pass have their criticality set to zero and the algorithm continues with the selection of best candidates meshes until the acquisition request is full relative to the acquisition capacity of the satellite. Algorithm 4 presents this simple greedy algorithm, with:

- **solution**: the list of selected meshes
- n_{mesh}: the number of meshes in the solution,
- n_{lines}: the number of lines in the solution,
- **criticality()**: the function calculating the criticality of a mesh, the same function is used to calculate the criticality of *Mesh* agents in Glimpse,
- **nb_lines()**: the function calculating the number of lines in the solution,
- **list_mesh**: the list of meshes in the pass,
- **mesh**: a mesh represented by a list [i,j,H,L] with i,j the coordinates of the cell on the lower left corner of the mesh and H, W its height and width
- L_{max}: the maximum number of lines of cells that can be acquired during the pass, representing the maximum acquisition capacity of the satellite.

Algorithm 4 Greedy algorithm

Require: $N_m \geq 0$
  $n_{mesh} = 0$
  $n_{lines} = 0$
  $solution = []$
1: **while** $n_{mesh} \leq N_{max} \& n_{lines} \leq L_{max}$ **do**
2: $best\_criticality = 0, best\_mesh = 0$
3: **for all** $mesh \in list\_mesh$ **do**
4: **if** $criticality(mesh) > best\_criticality \& n_{lines} = nb\_lines(mesh) \leq L_{max}$ **then**
5: $best\_mesh = mesh$
6: $best\_criticality = criticality(mesh)$
7: **end if**
8: **end for**
9: **if** $best\_criticality \geq 0$ **then**
10: $list\_mesh = list\_mesh \backslash best\_mesh$
11: $n_{mesh} \mathrel{+}= 1$
12: $n_{lines} \mathrel{+}= nb\_lines(best\_mesh)$
13: $solution.add(best\_mesh)$
14: **end if**
15: **end while**
16: **return** solution

This algorithm shows a greedy behavior of selecting meshes one by one. However the optimal solution is the best set of N meshes, which may be different from the set of the best N meshes. Indeed not taking into account the crossover surfaces of meshes means a mesh cannot be evaluated independently from others but needs to be evaluated jointly with the other meshes of the solution, making the problem NP-Hard.

6.3 Performances

As an *LARM* the goal of Glimpse is to optimize the metrics of the *LAC* problem. In order to evaluate how adequate the *AMAS* approach is to this problem a methodology of testing the performances of Glimpse has to be defined. In this section a preliminary study is proposed. The evolution of the completion time and waste are measured during the solving and at the end of *LAC* missions. A benchmark method and a *LAC* scenario on which the benchmark is run are first presented. Then the results of this preliminary are discussed in relation to the behavior of the benchmarked algorithm during the solving.

6.3.1 Comparison

For the purpose of comparison we choose to compare Glimpse using dynamic meshing to the greedy algorithm of section 6.2.5 using static meshing instead of the operational method of manually splitting the zone in subzones before mission start. The goal is to obtain a

benchmark of both mission result and evolution of the completion time and waste in order to observe the benefits of one method over the other. The grid used by the static meshing algorithm is obtained by dividing the zone in mesh placements covering the *AOI* entirely. For homogeneous satellites (with equal swathes for acquisitions) these grids overlap. For heterogeneous satellites these grids are distinct. In this experiment we test the performances of **static and dynamic meshing** approaches on missions including either **homogeneous or heterogeneous satellites**.

For fairness, we use the same evaluation method for meshes in static meshing as in dynamic meshing. This means the importance of acquiring a dynamic mesh, which for Glimpse is tied to its criticality, is the same for a static mesh placed at the exact same coordinates.

6.3.2 Scenario

With no knowledge of the benefits of a particular scenario to either meshing technique we selected a scenario relevant to operational use cases to compare static and dynamic meshing algorithm. The following characteristics were used:

1. Australia as *AOI*,
2. cell size of 30x30km,
3. 4 *MPFs* operating 1 satellite each,
4. a delay t_{acq} between satellite acquisitions and acceptation or rejection of images of 24h.

For the homogeneous satellite experiment we used mesh sizes of 60km in width and 120km in length. For the heterogeneous satellite experiment we used 4 different mesh sizes, one for each satellite, respectively:

- 60km/90km,
- 60km/120km,
- 60km/150km,
- 90km/180km.

All of the above characteristics were obtained by consulting domain experts and following a mission pattern that could have been followed operationally, with the exception of cell size which was chosen as a common denominator for the heterogeneous mesh sizes selected.

6.3.3 Results

Figures 6.1 and 6.2 respectively show a comparison of the completion rate and waste between dynamic meshing and static meshing for the same scenario using homogeneous satellites. The graphs present an evolution of each metric during the resolution, with time represented as the number of passes on which the meshing and planning process have been

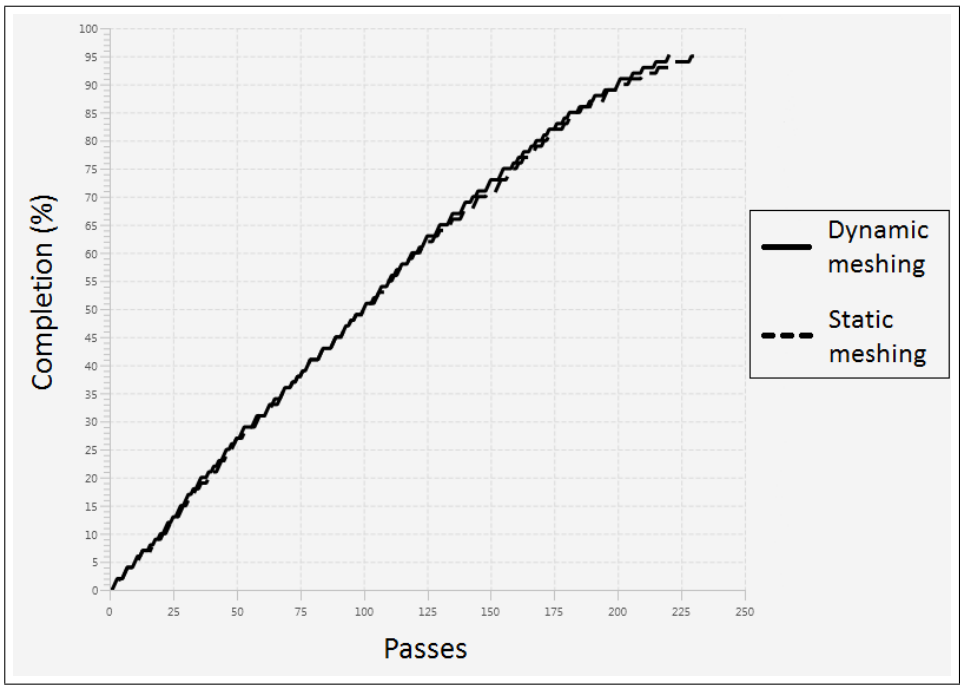

Figure 6.1: Evolution of the completion rate by meshing type for a scenario with homogeneous satellites

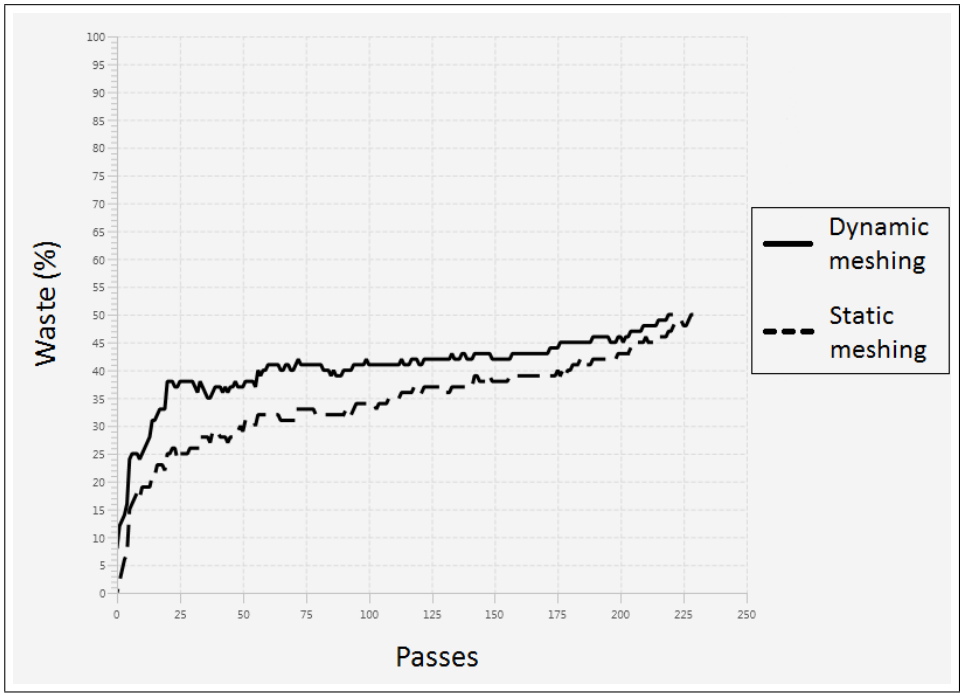

Figure 6.2: Evolution of the waste by meshing type for a scenario with homogeneous satellites

executed. Numerical results are presented in table 6.1, with the completion time represented by the number of passes computed at 95% completion and the waste by the percentage of total extraneous surface acquired at 95% completion.

Similarly, figures 6.3 and 6.4 respectively show a comparison of the completion rate and waste between static meshing and dynamic meshing for the same scenario using hetero-

	Passes	Waste(%)
Greedy (static meshing)	227	50
Glimpse (dynamic meshing)	**222**	50

Table 6.1: Performances of meshing methods on a scenario with homogeneous satellites

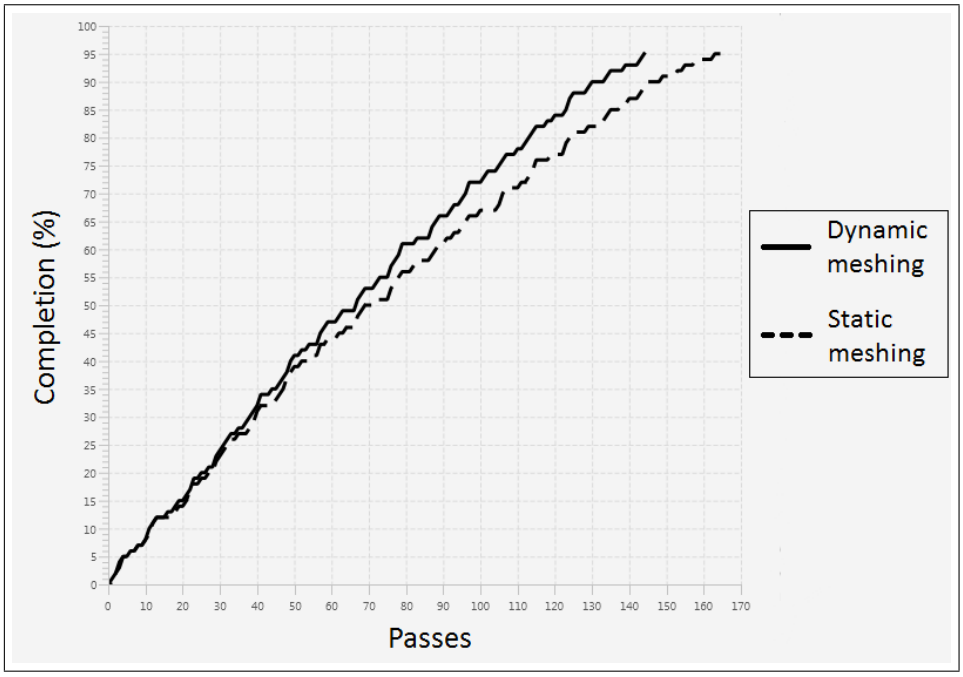

Figure 6.3: Evolution of the completion rate by meshing type for a scenario with heterogeneous satellites

geneous satellites. Numerical results are presented in table 6.2, with the completion time represented by the number of passes computed at 95% completion and the waste by the percentage of total extraneous surface acquired at 95% completion.

	Passes	Waste(%)
Greedy (static meshing)	163	61
Glimpse (dynamic meshing)	**144**	**52**

Table 6.2: Performances of meshing methods on a scenario with heterogeneous satellites

6.3.4 Analysis

From the results presented in the previous section we propose several observations:

- similar completion time and waste results are obtained with both meshing types on the scenario that features homogeneous satellites,
- **improved completion time and waste results are obtained with dynamic meshing above static meshing on the scenario that features heterogeneous satellites,**

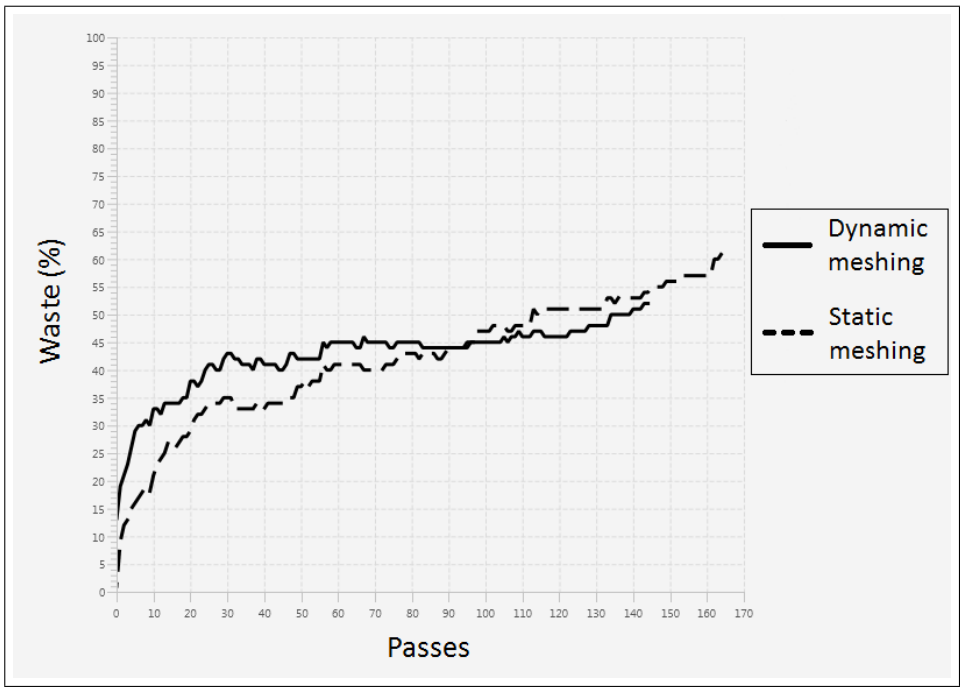

Figure 6.4: Evolution of the waste by meshing type for a scenario with heterogeneous satellites

- the curves of the waste evolution graphs for both types of meshing show differences in the rate of waste accumulation during the missions.

Homogeneous Satellites. As previously discussed in section 5.2.1 homogeneous satellites are the best case scenario when using static meshing. Meshes of the different satellites do not overlap, meaning a mesh acquired and validated on a pass of one satellite corresponds to the same mesh area being validated on all passes of any other satellite. The results obtained in table 6.1 show equivalent performances for both completion time and waste using both meshing methods, with only a small improvement in the number of passes necessary to reach 95% completion when using dynamic meshing. This improvement can be attributed to small optimizations present in dynamic meshing such as a better use of mesh placements close to the edges of the corridor or *AOI* where mesh positioning is more constrained.

The result presented in figure 6.1 intuitively points towards similar mesh placements in both meshing methods as the evolution of completion is similar during the mission. However the analysis of figure 6.2 reveals that using static meshing, the initial mesh placements create less waste than those placed using dynamic meshing. For example at 25 passes acquired the waste using static meshing is at 25% while the waste using dynamic meshing is at 38%. Inversely, more waste is created later during the solving when using static meshing as the total waste observed at 95% equalizes at 50% for both meshing methods. This corresponds to our assumption that **using dynamic meshing is less efficient than static meshing early on** in the *LAC* missions as the freedom of mesh placements creates gaps between meshes and overlaps that worsen the observed waste.

The waste using static meshing in these early stages is low due to the high number of meshes in each pass that present sufficient weather forecasts. As the mission progresses

more static meshes are validated and less static meshes are available for acquisition in each pass. Static meshes are then acquired repetitively even when only part of the acquisition is predicted cloudy, which makes the waste rise as a pass with good weather needs to be found to acquire each remaining mesh. When using dynamic meshing the meshes created during the later stages of the solving adapt to the already validated areas and clouded weather forecasts to target specific areas presenting high likelihood of successful acquisition. **The initial high waste is compensated by the adaptive placement of meshes lowering the rate of waste accumulation** to reach similar total levels to those observed using static meshing. In the case of scenarios using homogeneous satellites we find consistent similar results between static and dynamic meshing.

Heterogeneous Satellites. Strictly heterogeneous satellites (no satellite pair has the same swath) is on the opposite as the worst case for static meshing and intuitively the best case for dynamic meshing. In this case the static meshes have a high surface of overlap as the mesh grid of each satellite is distinct from each other. **Dynamic meshes are less affected as their placement and shape can change to fit in the gaps created by the acquisition of meshes of different sizes**. This is reflected in the obtained results. Figure 6.2 presents an improvement relative to the completion time of 19 or 11% less passes necessary to reach 95% completion using dynamic meshing. The waste is also improved at 9% less extraneous surface acquired using dynamic meshing.

To track the evolution of the results through the solving we use the same tools as for the scenario with homogeneous satellites. Figure 6.3 shows a near constant validation rate for dynamic meshing during the mission only decreasing near the end of the solving. **Static meshing is initially as effective but gradually reduces its validation rate during the solving**. From our observations in the previous experiment we can assume that the evolution of waste is relevant to the difference in completion time. Figure 6.4 presents similar behaviors of waste accumulation than those observed on the scenario with homogeneous satellites. The waste using dynamic meshing is initially high and stabilizes in the middle of the execution with a small augmentation near the end of the mission. The waste using static meshing is more constantly accumulated during the mission but also higher initially.

At pass 25, the observed waste levels with static meshing and dynamic meshing are respectively 34% and 41%. This corresponds to an augmentation of 9% for static meshing but only 3% for dynamic meshing when compared with the scenario with homogeneous satellites. During the rest of the solving we observe the same pattern with the waste using static meshing being higher than the waste using dynamic meshing after 95 passes or 58% of the total completion time. This behavior confirms the difficulty of validation of the static meshing at later stages of the solving. As the mission progresses more static meshes overlap already validated parts of the *AOI* and the total surface validated per pass is reduced. The near constant rate of validation of the dynamic meshing shows that **the surface validated per pass remains constant as the dynamic meshes adapt to fit around the validated areas**. This hypothesis is reinforced by the waste evolution stagnating as the validated areas are not re-acquired, with only a small increase near the end of the solving related to the difficulty of finding passes with sufficient weather to acquire remaining meshes.

Overview. These experiments show the improved performances of the dynamic meshing over the static meshing on the same *LAC* scenario with homogeneous and heterogeneous satellites. In the best-case scenario for the static meshing of homogeneous satellites the dynamic meshing produces small improvements in completion time. In the best-case scenario for the static meshing of heterogeneous satellites both the completion time and waste are significantly improved when using dynamic meshing. In the next section we expand on this experiment with other scenarios to test the robustness of Glimpse and dynamic meshing.

6.4 Robustness

Earth observation using satellites features a wide range of possible client requests. The goal of an *LARM* is to be flexible in order to answer any type of client request. An *LARM* needs optimize in order to find the best solution regarding the problem metrics with the resources available. The robustness of *LARMs* represents this ability to adapt. In this section we consider a larger variety of *LAC* scenarios and evaluate the robustness of Glimpse using these scenarios. First we describe the robustness criteria of the scenarios, the goal of studying the impact of each criterion and the scenarios themselves. We then present the results of Glimpse and dynamic meshing and compare them to those obtained with the greedy algorithm of section 6.2.5 and static meshing. Finally we discuss the robustness of Glimpse taking into account the results for each criterion and the evolution of the problem metrics during the solving.

6.4.1 Robustness Criteria

As presented in section 5.3.1 the input data of an *LARM* consists of several elements such as the *AOI*, satellites, passes and weather data. In order to verify our initial performance results, we define these elements as 4 criteria that are representative of the scenarios used. These criteria are:

1. **Area size.** Size of the *AOI* in km^2. The *AOI* in the acquisition request can be of different properties depending on the need of the client. The size of an *AOI* can range from small countries to continents. This criterion is related to the **scalability of Glimpse** as the search space grows exponentially with the the surface area to acquire.

2. **Area shape.** Shape of the *AOI*. This criterion has an impact on the mesh placements over the edges of the area. Irregular edges can cause meshing challenges depending on the method used. Additionally, a long and narrow *AOI* presents different corridors when compared to a short and large *AOI*. Because of the ephemeris of the *LEO* satellites which follow a slightly tilted north/south axis the average area covered during a pass is higher for a long area compared to a large one. The amount of meshes available for acquisition during each pass is tied to the length of the corridor. Longer areas thus allow for more meshes to be acquired in each pass. This criterion allows us to validate the ability of **the system to self-adapt** to a variable number of meshes acquired and different configurations of meshes to place to cover the whole *AOI*.

3. **Weather.** Type of weather of a fixed area of interest. The acquisition of a specific *AOI* may vary depending on the starting date of the mission. Seasons have an impact on how likely it is that a given mesh is successfully acquired on average. This in turn impacts the completion time and waste due to weather related rejected acquisitions. This criterion is used to study the ability of Glimpse to **adapt to different types of weather** observed in operational contexts.

4. **Number of satellites.** Number of satellites available during the mission. Satellite constellations vary in size and may be operated by one or several *MPFs*. Different numbers of satellites allotted to the mission change the **minimum completion time**. The **coordination between satellites** is also accentuated particularly in the case of heterogeneous swathes. This criterion allows us to experiment on the adaptability of Glimpse to adapt to large heterogeneous satellites constellations. It also guides operators in choosing the best size of constellations or individual satellites to submit for a mission.

Following the reasoning used in section 6.3.1 we benchmark Glimpse using dynamic meshing to a greedy algorithm using static meshing. We compare completion time and waste for both algorithms at 95% of the total *AOI* surface validated.

6.4.2 Scenarios

For each criterion presented in the previous section we define several example inputs and by extension scenarios to be used as benchmarks. Chosen scenarios represent both usual operational use cases and atypical or exotic use cases to highlight the potential differences in results obtained per criterion. Experiments on these scenarios are performed in batches of 10 and the results averaged in order to reduce the variance caused by the simulated weather. In this section we discuss the choice of scenarios.

For all robustness scenarios a number of global parameters are defined as follows:

- cell size of 20x20km,
- mesh size varying between 20x40km and 60x120km depending on the scenario,
- 4 *MPFs* operating 1 satellite each except when specified otherwise,
- strictly heterogeneous satellites (no identical swath pair),
- a delay t_{acq} between satellite acquisitions and acceptation or rejection of images of 24h.

The parameters for the robustness criteria defined in section 6.4.1 are as follow:

1. **Area Size.** For this criterion the following countries are selected by ascending order of area surface in km^2: Italy (301 338), France (643 801), Myanmar (676 575), Algeria (2 382 000) and Australia (7 692 000). An overview of each country and its precomputed cell grid in ascending order of size is shown in figure 6.5. Chosen countries show a range of *AOI* sizes from a medium sized country to a country of continental scale, with the size of the smallest sample (Italy) at 4% of the size of the biggest (Australia). Though the

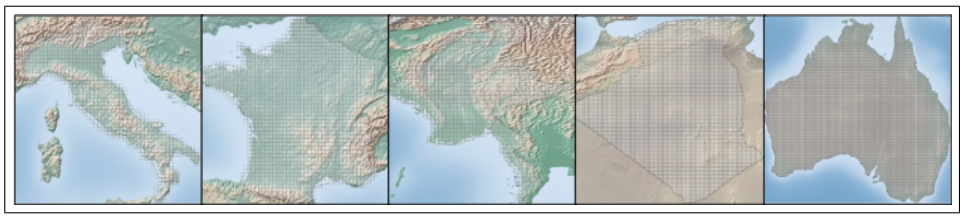

Figure 6.5: *AOI* of the scenarios used in the area size criterion evaluation by ascending order of surface size

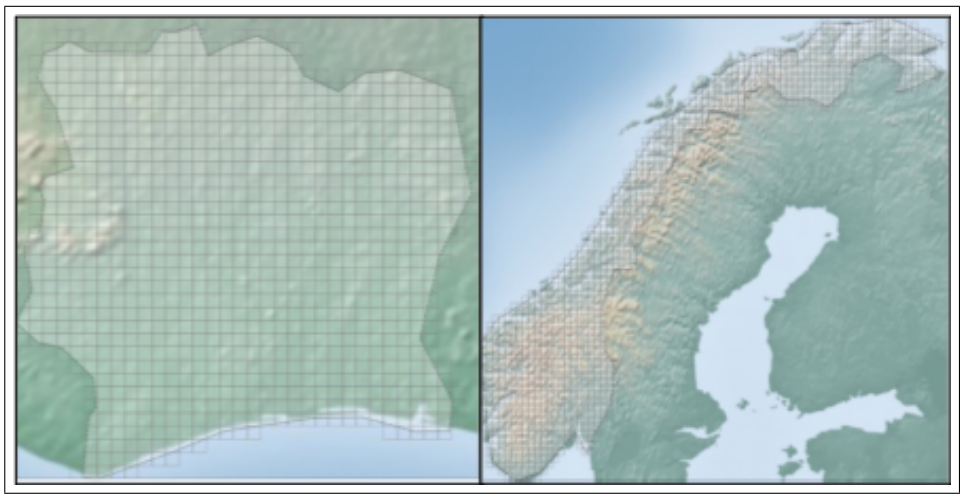

Figure 6.6: *AOI* of the scenarios used in the area shape criterion evaluation

range is not strictly continuous, the evolution of completion time and waste through the different scale may outline a function of performance relative to the area size when using static or dynamic meshing.

2. **Area Shape.** For this criterion 2 areas are selected by their shape: mainland Norway and Ivory coast both represented with their cell grid in figure 6.6. Both areas share a similar surface of about 320 000 km^2 but the outline of the corridors during the solving varies drastically. Ivory Coast presents a rectangular shape with mostly straight edges while mainland Norway is tilted diagonally from the north/south axis with irregular edges. These scenarios are chosen to highlight the importance of mesh placements in irregularly shaped corridors and the relation between area shape and performance for each meshing method.

3. **Weather.** For this criterion we consider a single *AOI* with a large variability between weather forecast depending on the date of the mission. Myanmar is chosen as the impact of the monsoon on both completion time and waste is expected to highlight the difficulty of acquisition during unfavorable weather when compared to other more favorable seasons. The starting dates for the two scenarios are fixed as January 2015 and June 2015, with June as the start of the monsoon season. Figure 6.7 shows an example of typical expected weather for both January and July on Myanmar with the precomputed cell grid.

4. **Number of Satellites.** For this criterion the coverage of Australia by 2, 3 and 4 satellites

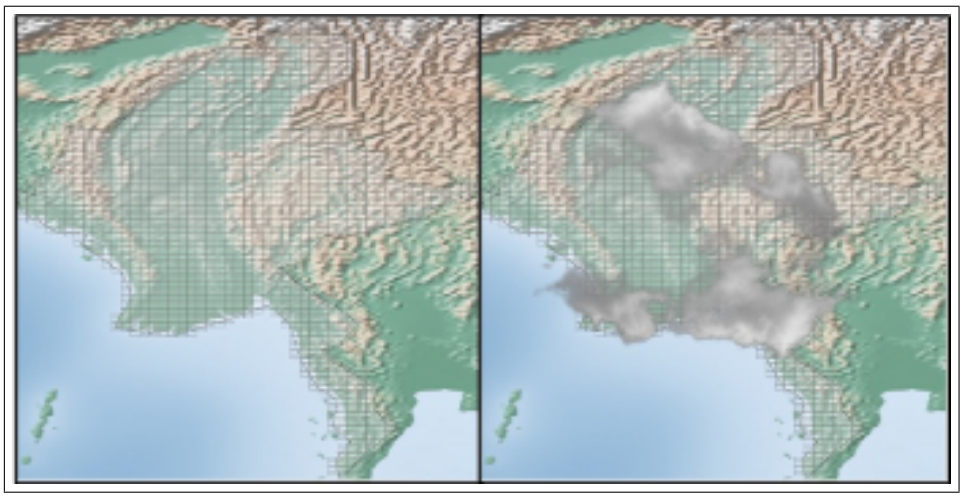

Figure 6.7: *AOI* of the scenarios used in the weather criterion evaluation of Myanmar in January (left) and June during monsoon (right)

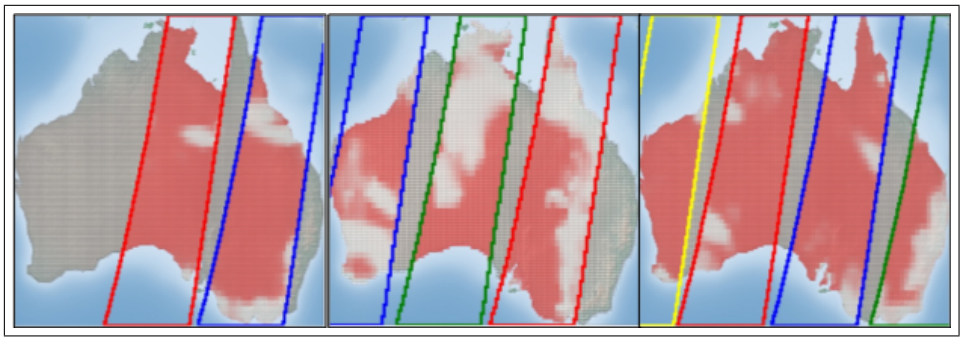

Figure 6.8: *AOI* and example corridors of the scenarios used in the number of satellites criterion evaluation from 2 satellites (left) to 4 (right)

is considered. As each satellite is strictly heterogeneous from each other the problem becomes more combinatorial for each added satellite. The difference in time necessary to complete the mission and waste relative to the number of satellites is also studied through this criterion. Figure 6.8 presents the *AOI* and different consecutive passes for each scenario by number of satellites operated.

6.4.3 Results and Analysis

1. Area Size

Completion time for this criterion are presented in table 6.3. Small *AOI* are represented by Italy, France and Myanmar. Time completion for Italy and France is equal between both algorithms, showing similar paths and bottlenecks in the search space leading to equal results in passes required. Myanmar is used as control sample for the possible improvement in completion time on small *AOI* by Glimpse. In the case of Myanmar Glimpse found solutions that are 13 passes better than the greedy solutions on average. We suppose that better use of earlier passes led to acquisitions of subareas that are otherwise only acquirable at pass 98. This corresponds to Glimpse avoiding a local minimum and finding a better overall

optimum than the one found on average by the greedy algorithm.

Similar results are found when considering larger *AOI* such as Algeria and Australia. Glimpse shows an improvement of around 10% in completion time over the greedy algorithm. **With these results we validate the scalability of Glimpse as it is best adapted to large countries and high number of cells**. On smaller countries the results may be equal or sometimes better for Glimpse. The improvement in time completion itself does not scale and remains stable at around 10% for each case of large *AOI*.

Glimpse also reduces the waste for every case by different margins as shown in table 6.4. Depending on the *AOI* the waste produced by Glimpse is decreased by 4% for France to 42% for Algeria. This constant decrease underlines the adequacy of the criticality measure to guide the agent decisions. We suppose better mesh placements during the solving lead to both a decrease in the waste produced and a better exploitation of passes to find better time optimums as mentioned previously. Differences in ratio of decreased waste and completion time observed may be due to other experimental factors that are explored in the following criteria.

	Italy	France	Myanmar	Algeria	Australia
Greedy (static meshing)	66(17d)	82(17d)	98(20d)	156(26d)	224(23d)
Glimpse (dynamic meshing)	66(17d)	82(17d)	**86(19d)**	**143(24d)**	**204(21d)**

Table 6.3: Completion time of static and dynamic meshing for areas of different sizes

	Italy	France	Myanmar	Algeria	Australia
Greedy (static meshing)	267	458	86	50	67
Glimpse (dynamic meshing)	**219**	**440**	**58**	**29**	**40**

Table 6.4: Waste(%) of static and dynamic meshing for areas of different sizes

2. Area Shape.

Due to the rectangular shape of Ivory Coast, a large number of meshes placed during the solving are not neighboring the outline of the *AOI*. The inverse is true for mainland Norway. This can be verified with an hypothetical *AOI* of width equal to the largest allowed swath between all satellites. In this example every mesh is adjacent to the edge of the *AOI*. As the width extends and the length reduces to keep the same area size and form a more rectangular shape more meshes become non-adjacent to the edges of the *AOI*. Similarly, Ivory Coast and mainland Norway share the same area size but present a very different distribution of area and outline.

Experimental results when using Glimpse and the greedy algorithm on both areas are shown in tables 6.5 and 6.6 for completion time and waste respectively. When considering Ivory Coast, the solving is fast in number of passes needed and a low amount of waste is produced. Results are similar for both algorithms with only a slight decrease in waste using Glimpse. In the case of mainland Norway both a diminution in completion time and waste are observed. 5 less days are required to acquire 95% of the *AOI* and waste is reduced by

36% when using Glimpse. We suppose this is due to the re-acquisition of cells on the edges of the *AOI*. Static meshes placed by the greedy algorithm cannot adapt their size to only acquire a few cells on the borders and utilize satellite resources to acquired many already acquired cells doing so. Meanwhile dynamic meshes placed by Glimpse vary in height to only acquire *Cell* agents that requested to be acquired and do not cover already acquired areas if not required.

This leads to a better overall use of satellite resources and the observed decreases in both completion time and waste. **Glimpse is thus more adapted to *AOI* presenting irregular edges while producing equal or better results on straight-shaped ones**. As for difference in completion time between both *AOI* it could be explained by different weather forecasts as explored in the next criterion experiment.

	Ivory Coast	Norway
Greedy (static meshing)	37(12d)	447(75d)
Glimpse (dynamic meshing)	37(12d)	**416(70d)**

Table 6.5: Completion time of static and dynamic meshing for areas of different shapes

	Ivory Coast	Norway
Greedy (static meshing)	62	3 801
Glimpse (dynamic meshing)	**52**	**2 422**

Table 6.6: Waste(%) of static and dynamic meshing for areas of different shapes

3. Weather.

Figure 6.7 presents the results of Myanmar in January 2015 and Myanmar during the monsoon season in June 2015 solved by Glimpse and the greedy algorithm. Similar results are observed both in the case of favorable and unfavorable weather forecasts for both algorithms, suggesting the monsoon conditions equalizes results instead of accentuating them. Through review of the experiments we noted that during bad weather events bottlenecks appeared in the solving. These bottlenecks correspond to specific areas that need to be acquired but are only under favorable weather during a few passes. The margins gained in completion time before these passes are negated as the observed bottleneck allows for more time to acquire other areas.

We observe this phenomenon with the advance gained by Glimpse using dynamic meshes being lost by waiting for a few specific passes to occur. Because of this phenomenon completion time results are smoothed out for both algorithms. Although local improvements obtained when using Glimpse before bottlenecks could be used to send early client answers or subareas of the *AOI*, the final results using completion time at 95% show equal completion time when using both algorithms.

Glimpse is however better at avoiding waste accumulation as shown in table 6.8. This is due to the adaptation of meshes to weather forecasts and reduction in meshes re-acquired due to being partially covered with clouds as is often the case with the greedy algorithm and

static meshes. The decreases completion time needed to reach each bottleneck when using Glimpse is due to this adaptation. **Clouded weather is thus still favorable for Glimpse as it utilizes satellites resources better and can provide anticipated client answers in the case of bottlenecks in the solving**.

	Myanmar January 2015	Myanmar June 2015
Greedy (static meshing)	63(15d)	670(166d)
Glimpse (dynamic meshing)	**59(15d)**	670(166d)

Table 6.7: Completion time of static and dynamic meshing for scenarios with different weather

	Myanmar January 2015	Myanmar June 2015
Greedy (static meshing)	130	2844
Glimpse (dynamic meshing)	**77**	**2427**

Table 6.8: Waste(%) of static and dynamic meshing for scenarios with different weather

4. Number of Satellites

All 3 scenarios for this criterion present similar completion time results between both compared algorithms as shown in table 6.9. On average the greedy algorithm is ahead by 3 passes on the scenario using 2 satellites and 2 passes behind on the scenario using 3 satellites. From these results we find that the difference in completion for both algorithms is not impacted by the number of satellites. The inverse is true when considering waste results presented in table 6.10. When compared to the waste produced by the greedy algorithm, the waste produced by Glimpse represents a decrease of 36%, 25% and 22% on the respective 2, 3 and 4 satellites scenarios.

While waste results are consistently improved, the best waste reductions using Glimpse are obtained for a small constellation of strictly heterogeneous satellites. This may be due to the fact that strictly heterogeneous swathes create incompressible waste for both algorithms. Another important point to consider here is the low difference in days to completion between 3 and 4 satellites. Adding a satellite reduced the number of days as one can expect but only by 2 while increasing the acquisition waste by 50% from 3 to 4 satellites. This can be explained by the overlaps between the corridors of the satellites and the gradual reduction in meshes to acquire over the course of the coverage.

Based on these results we consider Glimpse as a potential tool to optimize the number of satellites both at the start of a *LAC* problem and through its resolution. In this example the best configuration for the *LAC* mission could be the 3 satellites one as its completion time is 50% faster than the 2 satellites one while only producing 69% of the waste generated by the 4 satellites scenario. A new metric combining both completion time and waste is discussed is section 6.6.

	2 satellites	3 satellites	4 satellites
Greedy (static meshing)	**155(31d)**	123(16d)	140(14d)
Glimpse (dynamic meshing)	158(32d)	**121(16d)**	140(14d)

Table 6.9: Completion time of static and dynamic meshing for scenarios with different number of satellites

	2 satellites	3 satellites	4 satellites
Greedy (static meshing)	107	83	116
Glimpse (dynamic meshing)	**68**	**62**	**90**

Table 6.10: Waste of static and dynamic meshing for scenarios with different number of satellites

6.4.4 Robustness Discussion

Through experimentation on several scenarios of 4 criteria we evaluated the robustness of Glimpse and dynamic meshing when compared to a greedy algorithm using static meshing. Results obtained with Glimpse were consistently equal or better than those obtained with the greedy algorithm. Some scenarios presented specific strengths of Glimpse for each experimental criterion.

The area size criterion results validated the scalibility of Glimpse. Completion time and waste are significantly reduced in the case of very large areas which are more susceptible to be the focus of client requests.

The area shape criterion verified the ability of agents to self-adapt. By exploiting the advantages of dynamic meshing the agents were able to place meshes appropriately to fit the corridor borders and the area's outline geometry. The waste reduction is important in this case, showing a low number of validated subareas re-acquisitions on the edges of the *AOI*.

Weather anticipation was also confirmed with the corresponding criterion, **proving the ability of Glimpse to handle dynamics and possible weather modification during planning**. Waste reduction was important though the presence of bottlenecks was highlighted to show that in these cases the completion time equalizes through all algorithms used.

Finally the number of satellites criterion revealed that **Glimpse may act as a tool for operators in choosing the best constellations of satellites** in order to cover a specific zone in a given time. Waste reductions compared to the greedy were observed for any number of satellites with a larger difference for low amounts of satellites.

The stability of these observations validates the robustness of Glimpse to handle a large variety of cases that can be encountered in the *LAC* problem. **Glimpse is best suited for large, irregular shaped, cloudy areas making it a good algorithm for the solving of the most complex cases of the *LAC* problem**.

6.5 Criticality Sensitivity Analysis

Section 5.3.1.3 described the criticality of the *Cell* agents. This internal parameter guides the behavior of *Cell* and *Mesh* agents towards the prioritization of cells considered most critical (as in most important) throughout the agents' lifecycles. During the adhesion process presented in section 5.3.2 critical *Cell* agents requesting to be covered by neighboring *Mesh* agents have their requests accepted over less critical adhering *Cell* agents if the mesh already reached its maximum size and needs to move in order to satisfy the request.

In turn, more critical meshes are prioritized over less critical meshes in the acquisition request sent to the *MPF*. The criticality of *Mesh* agents is based on the criticality of the *Cell* agents that they are covering. This implies that **the criticality of agents is a determining factor in the quality of results produced by Glimpse** as different versions of *Cell* agent criticalities lead to different acquisition patterns. In this section a sensitivity analysis of the *Cell* agent criticality is described. First the criteria included in the criticality are presented, then the methodology used to compare these criteria and finally the results obtained through a benchmark of the different criteria combinations.

6.5.1 Methodology

The criticality criteria of *Cell* agents are described in section 5.3.1.3. Hereafter they are referred to as:

- "**Adhesion**" for the inverse ratio between passes on which the cell is adhering to a mesh and the total remaining passes on the planning horizon,
- "**Anticipation**" for the ratio between passes with better weather for the cell than the current one and total remaining passes on the planning horizon,
- "**Grid**" for the ratio of validated cells neighboring the cell on the grid,
- "**Passes**" for the inverse ratio between passes including the cell and the total remaining passes on the planning horizon,
- "**Validation**" for the state of validation of the cell,
- "**Weather**" for the probability of successful acquisition of the cell on the pass in regards to the weather weighted by the probability of successful acquisition on previous passes,

The criticality of *Cell* agents in Glimpse can be modified by changing both the criteria composing it and the order in which they appear. As the criteria are treated using a lexicographic order, criteria coming first in order have more impact as any decisive comparison cuts the comparison of criteria next in order. In the following a **criticality variant** represents an ordered list of *Cell* agent criticality criteria. A variant is named using the different criteria that compose it in lexicographic order, for example "validated_weather_pass_anticipation_grid". The list of variants is presented in table 6.11,

"weather_exp" is a different method of weather estimation that led to a decrease in performance and was not used as a result as presented in section 6.5.2 and "greedy" the greedy algorithm with static meshing included for the sake of comparison.

grid	0	validation_weather	21
passes	1	weather_grid	22
anticipation	2	weather_passes	23
validation	3	weather_anticipation	24
weather_exp	4	weather_validation	25
weather	5	validation_grid_passes	26
grid_passes	6	validation_anticipation_grid	27
grid_anticipation	7	validation_anticipation_weather	28
grid_validation	8	validation_weather_anticipation	29
grid_weather	9	validation_anticipation_weather_grid	30
passes_grid	10	validation_anticipation_weather_passes	31
passes_anticipation	11	validation_weather_grid_passes	32
passes_validation	12	validation_weather_grid_anticipation	33
passes_weather	13	validation_weather_passes_grid	34
anticipation_grid	14	validation_weather_passes_anticipation	35
anticipation_passes	15	validation_weather_anticipation_grid	36
anticipation_validation	16	validation_weather_anticipation_passes	37
anticipation_weather	17	adhesion_validation_weather_passes_anticipation	38
validation_grid	18	greedy	39
validation_passes	19	adhesion_validation_weather _passes_anticipation_grid	40
validation_anticipation	20		

Table 6.11: Criticality variants with relative criteria in lexicographic order

In order to compare different variants, experiments are performed on an aggregated test with 10 scenarios including the ones presented in section 6.3 and scenarios with perfect weather. All 10 scenarios of the aggregated test are numbered from 0 to 9 with their different parameters available in table 6.12. Scenario 8 features a simulated perfect weather as an experiment of the performances of variants on such a scenario, thus it does not have a start date. A set of 20 Glimpse executions is considered for each scenario to reduce the impact of randomness in weather and agent execution order. In section 6.5.2 criticality variants are referred to as the corresponding identification number in table 6.11.

6.5.2 Results and Analysis

Average Completion. The first benchmark for variants is the average completion of each variant for all 10 scenarios with aggregated results presented in figure 6.9. On this graph the theoretical maximum value is 0.95 representing 95% mission completion as per the mission constraints. Despite the unlikelihood of homogeneous satellites in an operational context

	AOI	Cell size(km)	Satellites	Mesh type	Passes	Date(dd/mm/yyyy)
1	Algeria	20	4	Heterogeneous	500	17/01/2015
2	France	20	4	Homogeneous	100	04/05/2015
3	France	20	4	Homogeneous	100	21/01/2015
4	France	20	4	Heterogeneous	100	04/05/2015
5	France	20	4	Heterogeneous	100	21/01/2015
6	France	30	4	Homogeneous	100	02/08/2015
7	France	30	4	Homogeneous	100	09/02/2015
8	France	30	4	Homogeneous	100	29/11/2015
9	France	20	3	Heterogeneous	300	X (perfect weather)
10	Italy	20	4	Heterogeneous	500	27/12/2015

Table 6.12: *LAC* scenarios for the aggregated sensitivity test of Glimpse variants

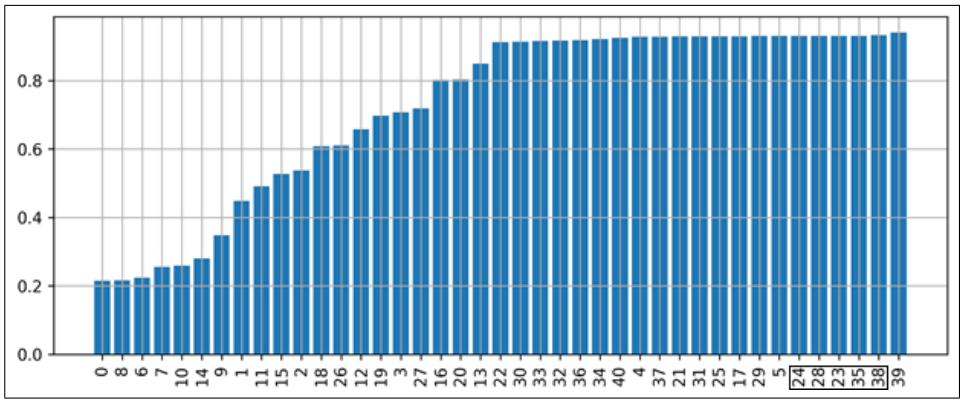

Figure 6.9: Average completion using different criticality variants on an aggregated scenarios experiment

we include 5 homogeneous satellites use cases due to them being the best case scenario for the greedy algorithm. The greedy algorithm then proves marginally better due to the high concentration of these homogeneous satellites scenarios in the set of scenarios experimented on. As for the variants evaluated, only 5 variants improve the results of the variant 5 which is only taking the weather of the cell into account:

- variant 23 (weather_passes),
- variant 24 (weather_anticipation),
- variant 28 (validation_anticipation_weather),
- variant 35 (validation_weather_passes_anticipation),
- variant 38 (adhesion_validation_weather_passes_anticipation).

A zoomed in view of the best results of figure 6.9 is available in figure 6.10. The results for the greedy algorithm average around 0.94 while the best variants average around 0.93 to 0.933.

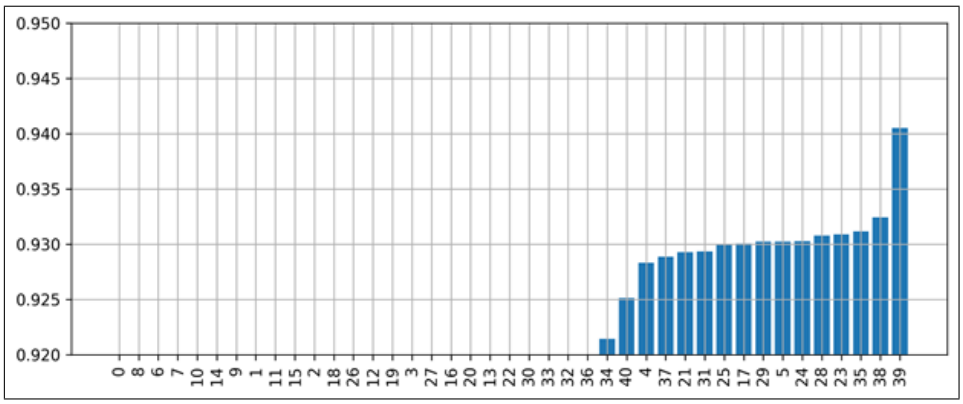

Figure 6.10: Zoom on the best criticality variants of the average completion graph

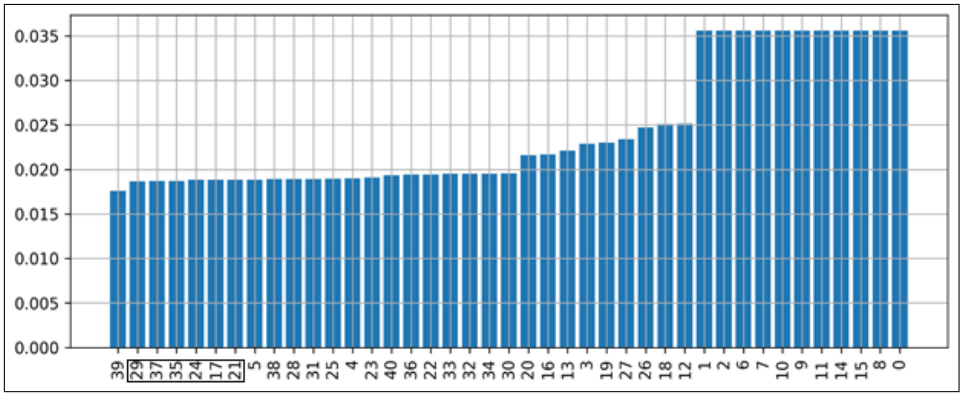

Figure 6.11: Average passes using different criticality variants on an aggregated scenarios experiment

Average Passes. A different approach of evaluating results is the observation of the average passes necessary to reach these levels of mission completion. To measure an average of passes across all scenarios a normalization of the data is required. Indeed depending on the scenario the average may vary a lot (for example 500 passes are necessary for a mission with Algeria as *AOI* while only 100 passes are necessary for France as *AOI*). Thus average the data without normalization would increase the weight of some scenarios. For each scenario the number of passes of a scenario is divided by the sum of all different passes on this scenario. Table 6.13 illustrates this process and figure 6.11 presents the results of the different variants in regard to the average passes found on the aggregated test.

Data	Raw	Raw			Normalized	Normalized
Scenario	Variant 1	Variant 2	Sum		Variant 1	Variant 2
Algeria	500	550	1050		0.47	0.52
France	95	100	195		0.48	0.51

Table 6.13: Normalization process for the number of passes on two example scenarios

In this case the best variant goes from left to right on the graph as a low number of passes indicates efficiency in the acquisition process to reach the completion level obtained in figure 6.9. 6 variants (29, 37 35, 24, 17, 21) show better results than variant 5 (weather

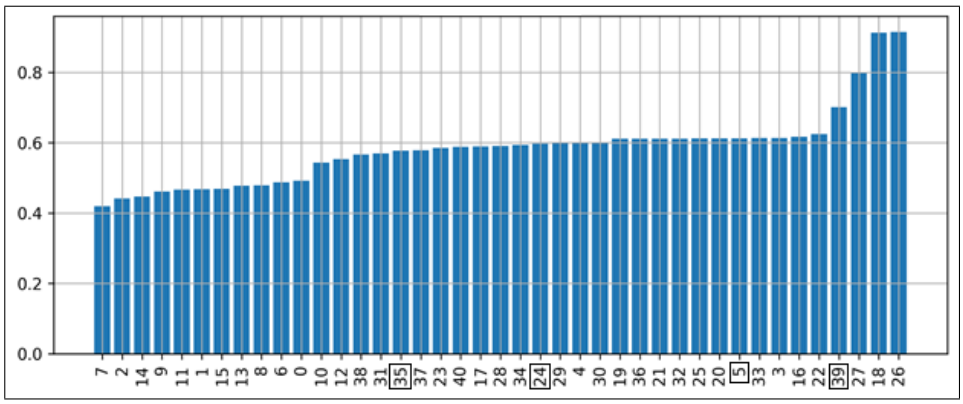

Figure 6.12: Average waste using different criticality variants on an aggregated scenarios experiment

criterion only) but only 2 are both better in both average completion and average number of passes:

- variant 24 (weather_anticipation),
- variant 35 (validation_weather_passes_anticipation).

Average Waste. Figure 6.12 presents the waste for each variant on the aggregated test. The first observation is that variant 39, representing the greedy algorithm, averages ∼70% waste and performs worse than most Glimpse variants. Variant 5 (weather criterion only) does not provide good waste results either. As for the most promising variants found previously, two variants also show good results for the waste metric:

- variant 24 (weather_anticipation) with ∼60% average waste,
- variant 35 (validation_weather_passes_anticipation) with ∼57% average waste.

Overview. According to these results the best overall variant of the ones tested appears to be **variant 35 (validation_weather_passes_anticipation)**. It proves to be competitive with all other variants and the greedy while significantly reducing the waste when compared to the greedy and other variants that performed well in terms of completion rate. For a more detailed analysis the comparison between this variant and the greedy algorithm is presented in figure 6.13.

Average passes of both the greedy algorithm and the best variant are equal or similar for most of the scenarios, particularly those featuring heterogeneous meshes. In the case of heterogeneous meshes, only scenario 8 of table 6.12 presents a significant difference in terms of number of passes to completion. In terms of waste, the best variant proves to be much more efficient than the greedy algorithm for almost all scenarios except scenario 6 of table 6.12. This may be due to the relatively small size of the *AOI*, France, and the favorable weather at the time of acquisition with a starting date of the scenario in August. These factors contribute to an easier acquisition overall which may favor the greedy algorithm as the selection of the best meshes in strict order and without going back on decisions may be

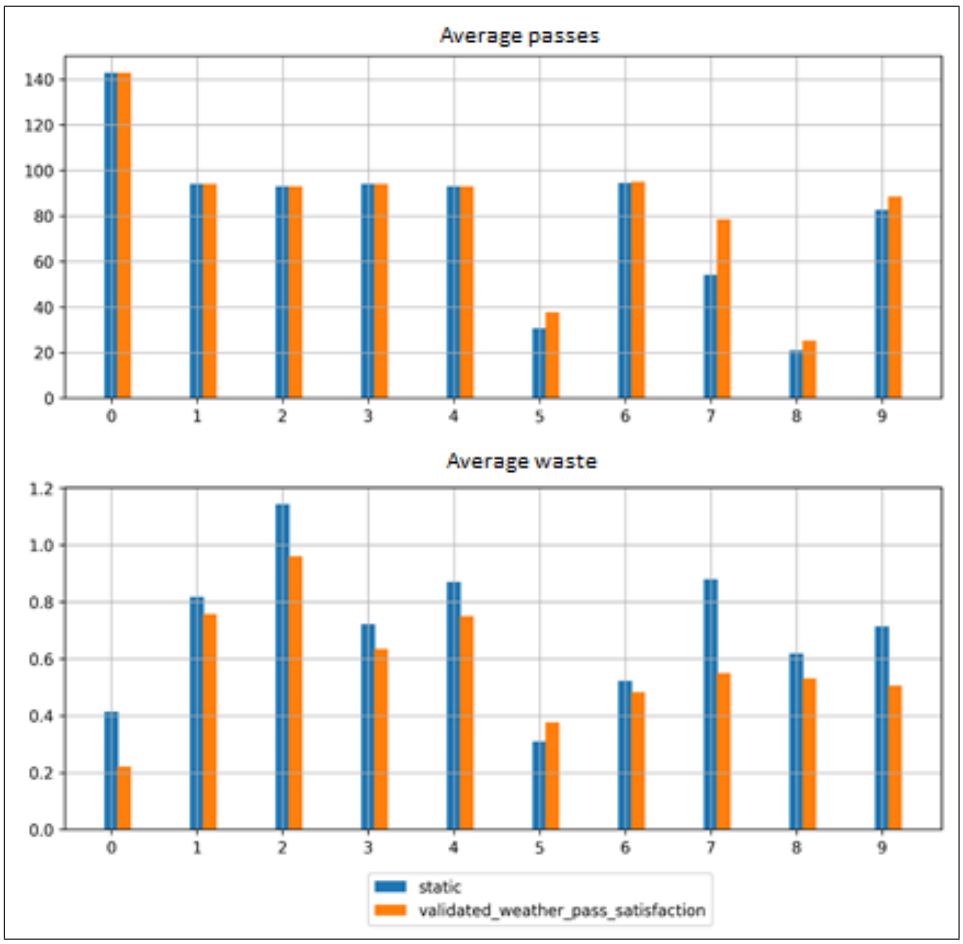

Figure 6.13: Comparison of average results per scenario of the best Glimpse variant (orange) and the greedy algorithm (blue)

sufficient for the solving. However on harder experimental conditions the greedy algorithm that uses static meshing suffers from a large amount of reacquisitions which worsen the waste results compared to the more accurate placement of the best variant of Glimpse that uses dynamic meshing.

Analysis. The results from this sensitivity analysis show that **different criticality variants do have an impact on the performance of Glimpse** as a system for the solving of the *LAC* problem. The intuitive approach that led to the design of the first criticality variant used for the experiments of section 6.3 proves to be reasonable. Indeed the variant used for these experiments is "validation_weather_anticipation_pass_grid" and the best variant found in this analysis is "validation_weather_pass_anticipation". **However the best variant found in the case of this aggregated test may not be the best for another set of scenarios**. Indeed, different variants may be more adequate depending on the number of satellites used, the size of the *AOI* or the average weather encountered during the time of the year from the starting date of the mission on the *AOI*.

In this regard **Glimpse allows for the tuning of parameters through the criticality criteria**. By using different *Cell* agent criticality criteria and orders of appearance one may find

a variant that is suited for the *LAC* missions of a specific *AOI*. The other approach consists of a systematic recommendation of variants suited to the characteristics of the mission. For example above a certain number of satellites a specific criterion may be recommended to be placed high in the lexicographic order. Indeed, a more in-depth sensitivity analysis demonstrated that the performances of Glimpse are improved through this method. Such a meta sensitivity analysis on more countries and using different sets of experimental conditions is a possible continuation of this work.

6.6 Synthesis

In this chapter the experimental conditions for the evaluation of performances of Glimpse and other *LAC* solving algorithms is first established. In this study weather forecasts are simulated using previous weather observations. This ensures a proximity with operational weather as randomly generated weather forecasts may presents patterns that are not observed in actual conditions. The metrics for evaluations are also chosen, with completion time and waste defined. A measure of the completion time at 95% of *AOI* validation is selected. This is justified by the lack of differences in performances when comparing *LAC* solving algorithm due to acquisition bottlenecks.

Indeed the areas in the remaining 5% are usually the harder to acquire, for example because of chronic cloudiness, meaning every solving algorithm needs to wait for a favorable pass which lets the less effective algorithms catch up on other acquisitions. For the sake of simplification an *MPF* that acquires every mesh included in acquisition requests is also chosen. The analysis of reasons for which an *MPF* would reject individual meshes is out of the scope of this study. To best compare the performances of *LARM* algorithms without this added parameter a compliant *MPF* is selected, though the algorithms are developed to communicate with an *MPF* that may reject acquisition requests.

Performances. A preliminary performance analysis is presented using Australia as *AOI*. The goal of this experiment is to observe the performances of *LARM* algorithms for different types of satellites constellations. The results of Glimpse using dynamic meshing and a greedy algorithm using static meshing are benchmarked on a use case of homogeneous satellites first, then on a use case of heterogeneous satellites. The homogeneous case is the best one for static meshing as no overlap exists between meshes, meaning the mesh grid of every satellites is the same. In this case results are near equal and the evolution of each metric similar, with Glimpse showing slight decrease in completion time required. **In the case of heterogeneous satellites a larger gap in results observed as both completion time and waste are significantly lowered using Glimpse**.

The evolution of waste through the execution of the greedy algorithm explains this difference. Indeed this waste gradually increases through the mission which can be explained by reacquisitions of already validated areas or worsening weather. However the evolution of the waste using Glimpse and dynamic meshing remains stable through most of the experiments, meaning the weather does not worsen on average. **This shows that the greedy using static meshing reacquires validated areas due to the overlaps between the mesh**

grids of the satellites. The diminution in efficient satellite resources use on each pass causes a decrease in average area acquired per pass. The completion time is impacted as a result, with less passes required for Glimpse to reach 95% completion than its greedy counterpart.

Robustness. These preliminary results are then expanded upon through the analysis of the robustness of Glimpse. The goal is to demonstrate the increase in performances when using Glimpse and dynamic meshing over static meshing on different use cases. This is justified by the large variety of *LAC* missions that may be encountered in an operational context. The parameters of the *LAC* missions that are modified to simulate this are the size and the shape of the *AOI*, the weather and the number of satellites. Relevant scenarios are built to represent possible differences in the respective parameters, such as Norway and Ivory Coast for the area shape parameter or Myanmar during monsoon or clear weather for the weather parameter.

The area size parameter shows that larger *AOI* tend to favor Glimpse as the differences in mesh positioning are exacerbated by the number of passes to reach 95% completion. **The area shape parameter reveals a tendency to favor Glimpse for elongated and irregular shapes** such as Norway due to the rigidness of the static mesh grid and its interaction with the edges of corridors on the irregular outline of the *AOI*. **The weather parameter highlights the ability of Glimpse to place and adjust meshes according to difficult weather patterns**, with equal waste results when compared to the greedy algorithm and static meshing during clear weather but reduced waste during monsoon season. Equal results are found in terms of completion time due to a significant number of acquisition bottlenecks through the execution, indicating that Glimpse acquired meshes more efficiently overall but needed to wait favorable passes to continue the solving. The number of satellites parameter shows similar results in terms of time to completion but consistent improvements in terms of waste when using Glimpse.

An observation is made that **Glimpse may be used as a tool to find the optimal numbers of satellites** for different *LAC* missions regarding the number of passes or days required to perform the acquisition. This study shows the robustness of Glimpse as every use case considered showed promising results either in completion time, waste or both when compared to a greedy algorithm and static meshing. It also highlights the potential of Glimpse as a decision tool to better tune mission parameters such as the number of satellites or their characteristics.

Criticality Analysis. The results obtained by Glimpse depend on the behavior of agents but also on the criteria chosen for the criticality of agents. In the previous experiments of this chapter the criticality of *Mesh* agents is based on a list of the most critical adhering cell agents in descending order, and the criticality of *Cell* agents on a list of criteria in a lexicographic order. These criteria and their order were found using advice from professional of operational systems and empirical tests. The last section of this chapter presents a sensitivity analysis that evaluates the performances of Glimpse depending on different sets and orders of criticality criteria called variants.

Experiments similar to the ones used in the robustness section are chosen in order to

benchmark the different variants. **Most variants obtain worse performances than the one using only the weather forecast as criterion which shows that the input of domain expert is sufficient to find a good variant**. The best variants found overall are both using a weather forecast criterion for the current pass followed by a criterion for future weather forecasts. This reasoning is applied in the original criteria ordering as the future passes help differentiate the potential for cells to be acquired successfully at a later date, diminishing the importance for them to be acquired on the current pass. The best variant adds two selection criteria, with the non validated cells and the cells having less remaining available passes on the planning horizons being prioritized.

This is also found in the original variant though the order of the future weather criterion and the remaining passes criterion is switched. Conversely, the cell grid criterion is not present in this variant, indicating that it may cause the worsening of performances contrary to the intuition of the criterion. **This analysis shows that the original variant used in the performance and robustness experiments is intuitively sound though it can be improved**. Different use cases may also benefit from different orders or criteria activated, the finding of which may be the goal of a future sensitivity analysis in the continuity of this study.

Glimpse. The performances of Glimpse were analyzed both in the context of comparison with a state of the art greedy algorithm and regarding its internal parameters represented by the criticality criteria. Using the observations and discussions of this chapter we add an *AMAS* row to the table 3.1 of section 3.4.4 as presented in table 6.14. *AMAS* are single state algorithms which lowers computing time. The agent life cycles are computed in parallel to allow for faster calculations, however the greedy algorithms are still faster due the absence of backtracking in the exploration of the search tree.

Section 6.4 showed the ability of Glimpse to handle different types of *LAC* scenarios with varied characteristics regarding the size and shape of the *AOI*, the number of satellites and the weather. The bottom-up approach of multi agent systems is suited for a large number of problem entities. **The scalability of Glimpse was verified using *LAC* scenarios with cells on large *AOI* such as Australia**. The ability of *AMAS* to avoid local minima sensitivity in optimization problems is tied to the representation of the constraint agent criticalities in the *AMAS4Opt* model. In section 6.5 this criticality was discussed in the context of the *LAC* problem and Glimpse. Though good criticality variants may be found using the logic and knowledge of experts of the *LAC* problem, systematic methods could be used to determine the best variants for specific problems. This perspective is discussed further in chapter 7.

A global observation based on the results of this chapter is **the ability of Glimpse using dynamic meshing to reduce the waste when compared with the greedy algorithm using static meshing**. This is partly explained by the better mesh positioning limiting waste due to meshes overlapping with areas subject to bad weather. Another explanation comes from the possibility of adapting the meshes to the geometry of the problem in the case of heterogeneous satellites. Overlapping meshes tend to create small gaps that lead to numerous reacquisitions when using static meshing, but only few reacquisitions when using dynamic meshing. Improvements in the completion time required to reach 95% of the *LAC* mission can be observed as a result but not always.

Algorithms	Computing time	Robustness	Scalability	Local minima sensitivity
Greedy	++	-	- -	-
Constrained	++	- -	++	- -
MCTS	- -	+	- -	+
GA	- -	+	+	++
EA	- -	+	+	+
SA	-	+	++	+
AMAS	+	++	++	+

Table 6.14: Evaluation of the *LAC* problem resolution metrics per optimization algorithm including AMAS graded from - - (low performance) to ++ (high performance) for each category

Indeed, a number of use cases present acquisition bottlenecks that lead to equalization of performances at one point in time during the mission and a homogenization of results. However a better acquisition rate during the mission is still useful operationally as client update can be performed when reaching a bottleneck. **Glimpse and dynamic meshing appear suited to reduce the mission completion time in some use cases including the case of heterogeneous satellites covering a very large *AOI* which is relevant operationally**.

Even for shorter missions, Glimpse can utilize satellites resources better than the greedy alternative to provide earlier update and can be used as a decision tool for the optimization of missions characteristics. As such, we suggest the use of a **cost metric** including both completion time, waste and number of satellites to evaluate *LAC* algorithms. This metric represents best the actual resources consumption of the observation satellites for the *LAC* mission and may be an alternative to completion time for long term acquisition missions such as Earth observation ones.

6.7 Conclusion

In this chapter we presented the experiments and methodology used to evaluate Glimpse. Different uses cases were established and Glimpse was compared to a greedy algorithm. Variants of the Glimpse agent criticalities were also presented with the associated results. The problem metrics obtained were analyzed and discussed for each experiment. In chapter 7 we summarize the contributions of the thesis and conclude with thoughts regarding the future of the *LAC* problem optimization.

7 Conclusion

EARTH observation using *Low Earth Orbit (LEO)* satellites is a field that includes a wide variety of optimization problems. Control centers called *Mission Planning Facilities (MPF)* handle a small constellation of satellites to answer client requests related to *Areas Of Interest (AOI)* that are completed after few acquisitions. This thesis considers the point of view of an overarching entity called a *Large Area Request Manager (LARM)*. A *LARM* answers client requests related to large *AOI* that require weeks or months of acquisitions to complete. The *LARM* itself acts as a client to several *MPF* in order to spread the acquisition workload. The scheduling of satellites is left to the *MPF* while the positioning and shape of meshes to acquire forms a new problem that the *LARM* needs to solve. We call this the *Large Area Coverage (LAC)* problem.

In chapter 2 we introduce the entities of the *LAC* problem. In this problem an *LARM* receives a client request for the acquisition of a large *AOI*. The *LARM* sends acquisition requests to *MPFs* during successive satellite passes. Acquisition requests may fail due to differences between weather forecasts and observed weather. Additionally, acquired meshes may partly cover areas that have already been validated. Both factors cause the underutilization of satellite resources, also called waste.

This is expanded upon in chapter 3 in which time to completion and waste are introduced as metrics for the *LAC* problem. A formalization of the problem is proposed and different approaches are considered. A simple operational technique that consists in dividing the area between available satellite is considered inefficient. Indeed limiting the acquisition possibilities of the satellite to limit interactions reduces the complexity of the problem but under uses the satellite resources. The formalized *LAC* problem is NP-Hard and as such we consider meta-heuristics instead of exact approaches. A state of the art of several candidate optimization algorithms for the solving of the *LAC* problem is then established.

In this thesis we focus on the solving of the *LAC* problem using *Adaptive Multi-Agent Systems (AMAS)*. These systems are composed of agents, elementary entities that self-organize to reach local goals while respecting cooperation rules. *AMAS* have been used to solve complex optimization problem and have proved efficient in their adaptation to perturbations during the execution. Chapter 4 presents the *AMAS* theory and several applications where the *AMAS* theory has been used to solve optimization problems.

In chapter 5, our proposition to address the problem using a dynamic meshing technique is introduced. This technique introduces cells, elementary subareas that allow for more precise mesh placement and adaptation to mission characteristics during the execution. A new formalization of the *LAC* problem is proposed. Based on this formalization, Glimpse, an *AMAS* for the solving of the *LAC* problem, is presented. The agents of Glimpse and their

interactions are then described.

Chapter 6 highlights several experimental results that validate the good performances and high level of robustness provided by Glimpse as an optimization algorithm for the *LAC* problem. A sensitivity analysis of the *Cell* agent criticality criteria shows the adequacy of the parameters used in previous experiments and the potential for tuning criteria variants to specific *LAC* use cases.

Scientific Contribution

The *LAC* problem explored in this thesis is an optimization problem that features resource allocations under time constraints. The number of acquirable meshes per pass is limited by the characteristics of the satellites such as maneuverability and speed. This restriction in the amount of resources and the importance of the meshes point towards similarities between the *LAC* problem and the knapsack problems[Martello and Toth, 1990]. In this field of optimization problems a container with limited capacity needs to be filled with items of differing importance and weight. The cutting stock problem[Cheng *et al.*, 1994] is another problem of the optimization literature with similarities to the *LAC* problem. In this type of problem different shapes need to be extracted from a plane while minimizing the wasted, unused surface. **Glimpse as an *AMAS* represents a contribution to these fields of optimization problems**, a topic on which we expand in the perspectives for future work.

Relative to the research in *AMAS*, this thesis features an analysis of the importance of agents through the criticality criteria. The *AMAS4Opt*[Kaddoum, 2011] agent model introduces "constrained" and "service" agent roles. Service agents use a notion of criticality to prioritize the fulfilment of requests sent by constrained agents. Agent criticalities consist of criteria listed in descending order of importance. **A sensitivity analysis is presented in this thesis relative to the performance of Glimpse when using different presets of criteria and order of criteria, called variants**. The results show that different Glimpse variants lead to performances gaps when considering the metrics of the *LAC* problem. **As a generalization, Glimpse variants may be used to optimize different problem metrics**. Specific variants may also be considered for specific use cases after a sensitivity analysis or other methods of finding the best variants. This topic is also discussed further in the perspectives of the thesis.

The criticality in Glimpse is used by *Pass* agents to relocate the least critical meshes and by *Mesh* agents to navigate on the cell grid. This agent architecture features different scales on the problems. *Mesh* agents represent a macro scale that communicates with *Cell* agents on a micro scale. The arrival of a *Mesh* agent triggers the start of a series of messages for neighboring and covered cells. **The proposed adhesion process between *Cell* and *Mesh* agents can be generalized to other optimization problems**. Indeed this process can be replicated using any macro scale agent that needs to answer to a large number of interdependent smaller scale agent that only accept one affection of a macro agent. This ensures potentially desirable properties such as a low number of affectation overlaps in macro agent over micro agents. In the *LAC* problem this is visible by the distribution of dynamic meshes over areas of high average criticality.

Operational Contribution

The process of developing a solving algorithm for a problem such as the *LAC* problem is often made considering the resolution in a top-down approach. Using the *LAC* example this refers to solving the problem with a central decision making entity that places meshes on the satellite passes and selects which meshes to send as acquisition requests to the *MPF*. **In this thesis we propose a bottom-up approach to the *LAC* problem.** The local goals of *Cell* and *Mesh* agents are first considered. Goals and communications to find the best mesh placements based on the local knowledge of the agents are described. *Pass* agents only relocate low criticality meshes after the self-organization of *Cell* and *Mesh* agents. This approach helps identifying the elementary agents and potential local goals for the *LAC* problem.

A decentralized decision making system answers some of the challenges of the problem. Multi-objective optimization with different problem metrics, such as the time to completion and waste, makes the definition and treatment of a global objective function difficult. Elementary agents with local goals allow for simple design and tuning phases during the development of the algorithm. The scalability of the algorithm is another important aspect for the solving of the *LAC* problem. A decentralized approach parallelizes the reasoning in different agents. The behavior of the agents can then be performed in parallel threads to lower computation time. **The agents and relations of the *LAC* problem developed in this thesis may be used for the design of other bottom-up optimization algorithms.**

The subdivision of the *AOI* in a cell grid itself is proposed in this thesis. The goal of the cell grid is to allow more accurate mesh placements in order to optimize satellite resource use. **We call the meshing process over a cell grid "dynamic meshing"** in opposition to the static meshing method where meshes are placed on a precomputed grid. In the case of static meshing meshes are never modified during the solving to keep the mesh grid intact while dynamic meshes can vary in both position and length. The hypothesis for the use of dynamic meshing is that meshes can be positioned to avoid overlaps between already validated areas, or to avoid unfavorable weather. Dynamic meshes should also help with irregular corridor shapes and small gaps between validated areas. **The experimental results show the adequacy of dynamic meshing with overall better results in completion time and especially in waste reduction**. As suggested dynamic meshing uses satellite resources more efficiently than static meshing in every considered use case.

The *LAC* problem is a known optimization problem in the Earth observation literature. Other approaches include the resolution of the problem using "strips", meshes obtained by acquiring in a straight line over the *AOI*. The goal of the problem when using strips is to find a series of satellite orientations, and thus strips, that maximizes area coverage. The introduction of meshes to the *LAC* problem adds a layer of complexity. When using static meshing a series of meshes on the mesh grids need to be selected for each pass. When using dynamic meshing a series of meshes need to be found on the cell grid for each pass. In this thesis we formalize the *LAC* problem for both static and dynamic meshing with problem characteristics, goal and objectives. **As operational contributions, these formalizations may be used to develop other optimization algorithms for the *LAC* problem using meshes.**

Future Work

This work opens new directions in both fields of Earth observation and AMAS research.

From a Scientific Point of View

As proposed earlier in this chapter the *LAC* problem shares similarities with different types of known optimization problems. Despite these similarities a number of characteristics of the problem contribute to its uniqueness and complexity. The number of meshes that can be acquired per pass varies in the case of dynamic meshing as the length of the meshes can be modified. The acquisition of meshes impacts future acquisition, cutting parts of the search tree for each validation. The uncertainty of the acquisitions means that each mesh needs to be considered as potentially validated or rejected after acquisition. All these characteristics separate the *LAC* problem from traditional knapsack or cutting stock problems.

However specific examples in these fields of problems similar to the *LAC* problem could be found. [Fajemisin *et al.*, 2023] presents an uncertain cutting stock problem in which the stock shapes are not known. [Neumann *et al.*, 2022] proposes an *EA* to limit the impact of uncertainty for the solving of a knapsack problem with stochastic profits. The literature regarding these problems and other similar problems would indicate potential optimization algorithms that have proved to be adequate to solve them. **A study comparing the results of Glimpse and other candidate algorithms would further highlight the strengths and weaknesses of Glimpse** and the *AMAS* theory when considering the types of problems similar to the *LAC* problem.

Regarding Glimpse itself, the sensitivity analysis proposed in chapter 6.5 reveals possible directions to improve the system. This analysis proved that the intuition of experts in the field of Earth observation led to good criticality variants that were used in the experiments of this thesis. The comparison of variants also showed significant differences between the worst and best performing variants when considering both problem metrics. A new variant that is not considered in this study could theoretically improve the results of Glimpse in most explored use cases. Finding a best variant for each use case is an optimization problem in itself, one that may be solved through the use of metaheuristics.

A possible approach is the use of machine learning to find both the criteria and their hierarchy. This would ensure a better performances when using Glimpse. Indeed, a more accurate determination of the criticality of cells and meshes would improve the likelihood of meshes being successfully acquired and increase the coordination between the passes of satellites. **This reasoning can be generalized to all constraint agents of the *AMAS4Opt* model**. Finding the criticality criteria that lead to the best self-organization of agents during the solving of any optimization problem is a topic that may be explored in future studies.

From an Operational Point of View

Earth observation problems can be viewed from different levels of relation to the satellites. The *LAC* problem considers a macro viewpoint where individual *MPFs* have their respec-

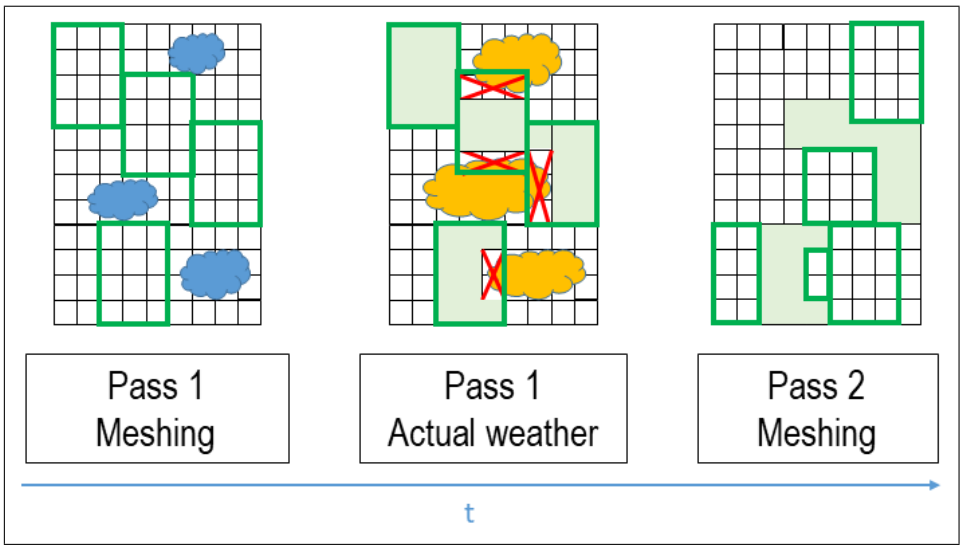

Figure 7.1: Partial validation of meshes and adaptation of dynamic meshes to the updated validated areas

tive planning and scheduling methods but share the characteristics of the satellites with the *LARM*. New technologies onboard the satellites or in image treatment on the ground can modify the problem. For example the improvements in the agility of the satellites in recent decades allows for the acquisition of meshes and a more flexible division of the *AOI*. **Another technology that is not explored in this thesis is the partial validation of meshes**.

Indeed each mesh acquired is considered fully validated or rejected after the acquisition process. This is reasonable when using static meshing as the precomputed mesh grid does not allow the adjustment of future meshes even if parts of the current mesh could be validated. **Dynamic meshing allows the use of partial validation where each cell that is usable in an otherwise clouded picture is validated**. This technique is made possible through image treatment on ground and reassembly of image parts when answering to the large area client request.

Dynamic meshes are adequate to use with partial validation as they can be rearranged to fit around already validated cells. Partially validated meshes drastically reduce waste and time to completion as they would be entirely rejected otherwise. Partial validation can be used with static meshing but its advantage is reduced as meshes often need to be re-acquired entirely if part of the resulting images is clouded. An illustration of partial validation used with dynamic meshing on a corridor is presented in figure 7.1. **Using Glimpse with partial validation is a possible continuation of this thesis in the evaluation of its performances with improved observation and imaging technologies**.

The *LAC* problem as formalized in section 5.2.2 abstracts a number of operational constraints for the sake of simplification. **Another future continuation of this work is the implementation of parameters that are considered operationally**. With ever-increasing image resolution capabilities the memory on-board of satellites is a constraint that needs to be considered in actual *LAC* missions. Memory calculations are performed to validate a mission plan for the satellite acquisitions between two downloading events. These are done by each *MPF* operating the satellites but could also be included in the mesh limit per pass that is

already implemented in Glimpse.

Similarly, the battery of satellites need to be recharged periodically through states during which the solar panels are directed towards the sun and the acquisitions are stopped. The satellites also need to perform maneuvers between each mesh acquisitions that are not strictly continuous. In the formalization of the *LAC* problem presented in this thesis all of these constraints that are abstracted by considering the *MPFs* as black boxes. The exchanges between *LARM* and *MPFs* are simplified as the *MPFs* always find an adequate satellite plan to acquire each of the requested meshes.

In operational conditions, this step may result in a number of rejected acquisition requests due to the impossibility to chain the maneuvers between acquisitions or due to other constraints such as the memory or power of the satellite. **A more transparent communication between *LARM* and *MPFs* would improve the coherence of simulated *LAC* scenarios** and allow the comparison of Glimpse to other algorithms that can be used in operational contexts.

Learning versus Adaptation

The field of wide coverage has highlighted a recurring issue when using cooperative agents for solving optimization problems: that of **defining priority criteria guiding the decisions of agents**. This work has shown the sensitivity of results to the hierarchical order of criteria, even if experts are able to specify a restricted subset of relevant criteria. There are two main methods for determining the ideal hierarchy:

1. by **learning from a corpus of examples**, based on feedback on the performance of each hierarchy. The first difficulty is the inherent combinatorial nature of all the criteria. The second difficulty is that learning presupposes a certain stability of the environment in which the learning takes place. This is not an obvious presupposition when we are interested in meteorology, especially in the current period of climate change which requires models to be constantly adjusted,

2. by **cooperative organization**, in which each agent can individually receive feedback from his or her neighbors, prompting for a reevaluation of its goals and a readjustment of the relative importance of these criteria. When this introduces a potential hierarchy update, the agent informs other agents of the same type to decide how to collectively self-organize the criteria. Cooperation thus becomes a means of adapting the internal behavior of agents, avoiding a fixed list of criteria.

Final Words

Earth observation problems form a vast field of research in optimization. Each step of the image acquisition process features numerous entities and resources that need to be managed. The scheduling of acquisitions and downloads by the *MPFs* operating the satellites are problems that are already discussed at length in the literature. In this thesis we focus on

the macro level, from the viewpoint of an *LARM* coordinating the *MPFs* to acquire a large area. The *LAC* problem we formalized in this context is complex. A continuous area needs to be imaged mesh by mesh over a long time period.

We increase the complexity further by proposing dynamic meshing, a novel approach to meshing that eases the constraints on mesh placements and shapes. To solve the *LAC* problem with dynamic meshing Glimpse, an *AMAS*, is developed. The results obtained using Glimpse on a variety of *LAC* scenarios show the adequacy of *AMAS* for the solving of such problems. Dynamic meshing itself proves to be a cost-efficient way to manage mesh placements, with significant improvements in the use of satellite resources over a more rigid meshing technique.

Through the study of the *LAC* problem using *AMAS*, this thesis contributes to several topics that may inspire future research. The coordination of satellites by an overarching manager is a complex field in which Glimpse provides an example of resolution by *AMAS*. The approach of using several types of macro and micro level cooperative agents for the solving of optimization problems is also described in this work. Additional improvements are also proposed for Glimpse such as the optimization of criticality criteria.

The emergence of a general behavior in the system is dependant on the local goal of agents and finding the best agent criticality for different types of problems and problem configurations appears essential. Machine learning and cooperative organization appear as possible adaptation mechanisms for the processing of updates with adjusted criticality criteria. An initial sensitivity analysis of the variants of Glimpse also suggests that finding optimal agent criticalities is itself a difficult optimization problem that may require meta-heuristics. As such this thesis highlights the importance of the methodology used to find agent criticalities during the design of *AMAS* for the solving of difficult optimization problems.

Part

Annexes

A Greedy Optimization

In section 3.4.3 we discussed the state of the art of optimization algorithms applied to the *LAC* problem and similar problems such as the daily satellite scheduling problem. Greedy algorithms were presented as optimization algorithms that are easy to implement but suffer from local minima trapping and scalability issues. Section 3.4.4 compares different algorithms with their estimated performances relative to the resolution metrics of the *LAC* problem that were introduced in section 3.3.2.

From these comparisons we can extract several better candidate algorithms to solve the *LAC* problem. This study presents the results of an internship by Enzo Pezzali on the improvement of the greedy algorithm used in section 6.3. The greedy algorithm developed as a result of this internship can be used to benchmark Glimpse against an algorithm with a more flexible mesh placement approach. This comparison is out of the scope of the thesis due to time constraints but may constitute a future perspective for the improvement of Glimpse. In this section an optimal approach for the solving of the *LAC* problem is presented, then the improved greedy algorithm that uses simulated annealing is described and finally the optimal approach and the two versions of greedy algorithms are benchmarked.

A.1 Optimal Solutions

A.1.1 Exact Approach

***Integer Linear Programming (ILP)* problems**[Graver, 1975] are optimization problems of which objectives and constraints are written as linear equations or inequalities with variables that only accept integer values. This type of problem is NP-Hard [Sherali and Driscoll, 2000]. An *ILP* problem can be formulated as follows:

$$\begin{gathered} \text{maximize } z = c.x \\ \text{subject to } A.x \leq b, \\ x \geq 0, \\ \text{and } x \in Z^n \end{gathered}$$

With:

- $x = (x_1, x_2, ..., x_n)^T$ the variable vector that contains the n variables of the problem,

- $A = (a_{i,j}) \forall i \in \{0, 1, ..., n\}, \forall j \in \{0, 1, ..., m\}$ the constraint matrix, with m the number of constraints,
- $c = (c_1, c_2, ..., c_n)$ the profits vector,
- $b = (b_1, b_2, ..., b_m)^T$ the vector of second members.

One of the appeal of converting a problem in an *ILP* problem is the possibility of using an **exact solver** to find optimal solutions. Optimal solutions are commonly found on small instances of the problem as exact methods explore the entire search space. On such small scale instances we can compare implemented algorithms and observe the distance between obtained results and the optimum. Due to the size of *LAC* instances (up to 500 000 variables) an exact solving is not feasible on large instances. **Comparison with the optimal solution on small *LAC* instances can however confirm empirically that the heuristics used are near the optimum**.

Solving a linear problem may be done using the simplex algorithm or the interior point method, which have the benefit of being exact methods as shown in [Mancel, 2004]. But the solving of an *ILP* is more complex and the simplex is not always satisfactory. Indeed the continuous relaxation of an *ILP* only has integer solutions in a few specific cases. As this is not the most common case, other methods are used such as branch-and-bound[Boyd and Mattingley, 2007] (separation and evaluation) on small instances. For larger problems, column generation methods such as described in [Mancel, 2004]. However for the size of problems similar to the size of big *LAC* instances, heuristics are better suited as they are faster [Sherali and Driscoll, 2000].

As such, **a formulation of the *LAC* problem as an *ILP* problem allows for the comparison of implemented algorithms to an exact solving on small scale instances**. In section A we propose a formulation of the *LAC* problem as an *ILP* problem and use it to implement an exact solver and compare its results to those of an improved greedy algorithm.

A.1.2 CPLEX

As an NP-Hard problem the *LAC* problem cannot be solved in reasonable time by an exact approach. For small instances of the problem however an *ILP* solver may find the optimal solution. This is useful for the comparison of performances of different algorithms and for measuring the distance to the optimum in each scenario. Section A.1.1 describes the formulation of an *ILP* problem. To solve small instances of the *LAC* problem as an *ILP* problem, we use **CPLEX**, an optimization tool developed by IBM to solve linear problems. CPLEX uses the simplex algorithm to find exact solutions of an *ILP* problem if execution is able to finish in reasonable time.

To illustrate this we consider the following example:

- France as *AOI*
- Weather data from July and August 2016

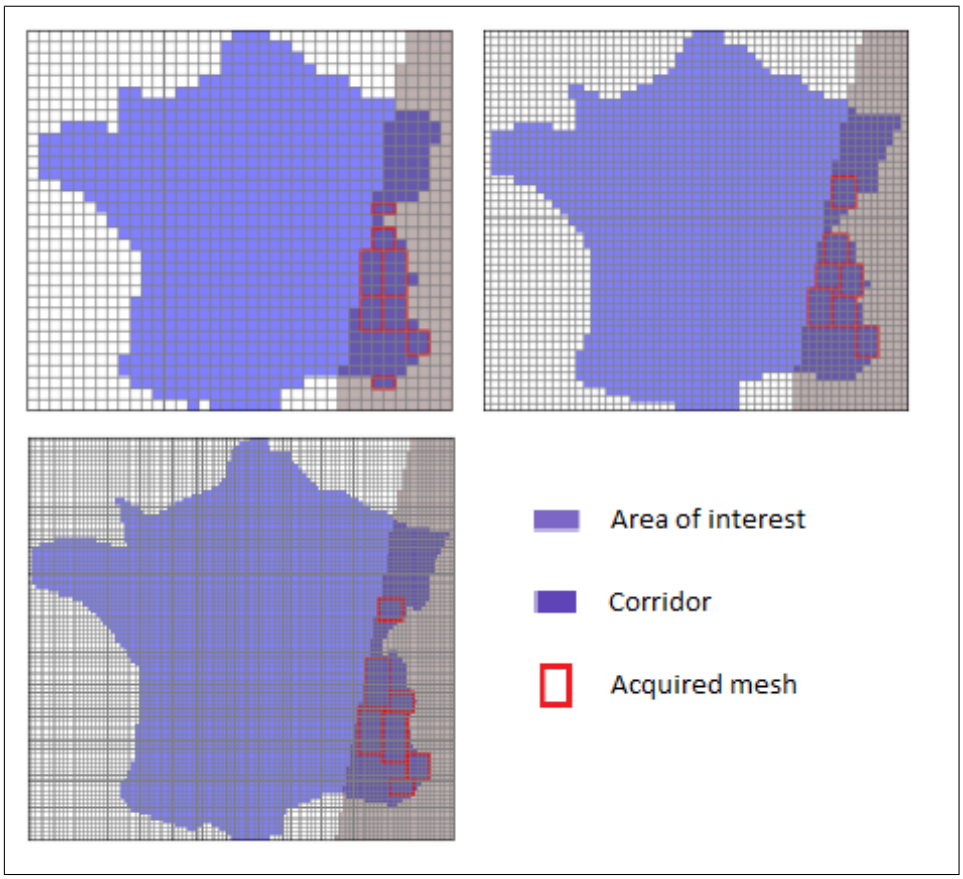

Figure A.1: Cell grids over France of respective size 30km (upper left), 20km (upper right) and 10km (lower right)

- A single *MPF* and two satellites of respective swathes 30km and 60km for 120km of maximum acquisition length

The cell grid is used to modify the scenario and change the number of variables that the CPLEX algorithm has to process. Three use cases are studied with cell sizes of 30x30km, 20x20km and 10x10km. Figure A.1 presents the three use cases with the respective cell grids and table A.1 presents the results of the execution of CPLEX on these instances. These experiments are performed on an Intel(R) Xeon(R) CPU E5-2695 v3 @ 2,30GHz two cores.

Cell sizes (km)	30	20	10
Number of variables per pass	4000	13 100	95 500
Computing time	1min10s	14min30s	>48h

Table A.1: CPLEX computing time relative to the size of the *LAC* problem instances

These results confirm the ability of CPLEX to solve small instances over larger ones. Indeed cell sizes of 30km and 20km require less than 15 minutes to find exact solutions which can be considered reasonable time. However the change from 13 100 to 95 500 variables when using 10km cells makes CPLEX require several days to complete the solving. The exponential change in variables is carried over to the computation time, making the use

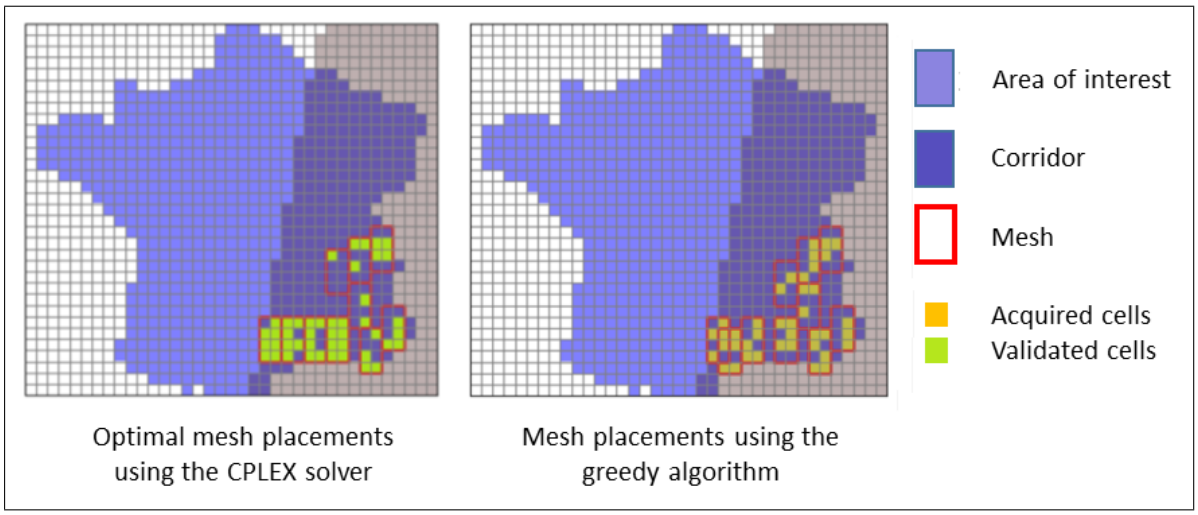

Figure A.2: Comparison of mesh placements between the CPLEX solver and the greedy algorithm for an early pass during a *LAC* mission

of CPLEX inadequate in this instance. This example is also a relatively small scale one as countries several times bigger than France may be *AOI* of client requests. Similarly, the solving is exponentially longer with the growing number of passes. For a use case of a small *AOI* but difficult weather, such as the monsoon example previously presented, the number of passes may render the exact solving impossible in reasonable time as well.

To solve larger instances metaheuristics are necessary but their performances can be evaluated using CPLEX on small instances. In section 6.2.5 a greedy algorithm for the solving of the *LAC* problem is described and a comparison to Glimpse in section 6.3. In section A.2 a preliminary evaluation of this greedy algorithm using CPLEX is presented.

A.2 Greedy Algorithm Evaluation Using CPLEX

To understand the difference between greedy solutions and CPLEX solutions we consider the scenario presented in section A.1. Figure A.2 shows the difference between the first acquisition performed by CPLEX and the greedy algorithm on the same instance. The obtained solutions differ by two meshes, which can impact the following solving on later passes significantly. The difference between acquiring the best set of meshes (left) and acquiring the best meshes one by one (right) is visible here. To note: the cells are not validated following the same patterns as the simulated weather differs in these cases but the relevant observation is that the greedy algorithm does perform suboptimal mesh selection when compared to the optimal solution of the CPLEX.

A simple demonstration is sufficient to show that the greedy algorithm can diverge by several meshes from the optimal solution as early as the first pass of a *LAC* instance. In section A.3 the greedy algorithm is improved using simulated annealing as a metaheuristic in order to get closer to the optimum while keeping reasonable computation time for large instances.

A.3 Hybrid Algorithm

A preliminary study of the greedy algorithm shows that its results are close to the optimal solution but vulnerable to local minima trapping. **To solve this local minima trapping issue, a local search is added to the greedy algorithm, making it a hybrid algorithm of a greedy algorithm using simulated annealing**. Simulated annealing is an algorithm presented in section 3.4.3 that gradually progresses from a exploratory state to an exploitative one using heating and cooling inspired methods. Simulated annealing is more resilient to local minima trapping than the greedy algorithm due to possible jumps in the search tree and an early exploration phase designed to avoid this phenomenon. However using simulated annealing alone proved to give similar results to the hybrid greedy and simulated annealing algorithm while requiring significantly more computing time.

Like the greedy algorithm, the hybrid algorithm needs to find a solution represented by a list of selected meshes. Its initialization state is obtained from the solution of the greedy on the instance. The algorithm progresses through the search tree by creating random neighbor states and evaluating their potential. **To create a random neighbor to the current solution a mesh is selected and moved by a length of 1 cell in any possible direction**, as is described in algorithm 6.

Algorithm 5 presents the general process of the hybrid algorithm. It first uses the greedy algorithm then looks for a good initial temperature thanks to a heat up loop and finally optimizes with the cooling. The internal process of the heating and cooling functions are abstracted for simplification. The parameters α, $NB\_TRANSITIONS$ and ϵ were chosen empirically. The function *check_constraints* has a solution as input and returns a boolean indicating if the constraints relative to the capacity in number of meshes and lines are valid.

Algorithm 5 Hybrid algorithm

Require: $N_m \geq 0$
1: $solution = greedy()$
2: $init\_temperature = heat\_up\_loop(solution)$
3: $solution = cooling\_loop(init\_temperature, solution)$
4: **return** solution

Simulated annealing does not only move meshes. Indeed the greedy algorithm has a tendency to select bigger meshes as they often have the best criticalities. The line capacity constraint is then quickly reached but not the number of meshes. To solve this issue the initial solution of the simulated annealing is modified by splitting the bigger meshes in two if the mesh capacity constraint is not saturated. Figure A.3 shows the purpose of this change as **the hybrid finds the same solutions as the CPLEX algorithm due to this ability of splitting the mesh**. The regular greedy algorithm only selects the mesh and is then unable to fit more meshes inside the corridor, wasting the acquisition of 3 acquirable cells with a better mesh split.

The neighborhood of a simulated annealing algorithm is important, as the computing time can grow considerably due to a bad neighborhood function or may even cause convergence issues. **Another possible approach is the random addition or suppression of a mesh**

Algorithm 6 Neighborhood

Require: $len(solution) > 0$
1: $neighbor = false$
2: **while** $neighbor$ *is* $false$ **do**
3: $S' = solution$
4: $ind\_mesh = random(0, len(solution))$
5: $ind\_coord = random(0, 1)$
6: $ind\_direction = random(-1, 1)$
7: $S'[ind\_mesh][ind\_coord] \mathrel{+}= ind\_direction$
8: $neighbor = check\_constraints(S')$
9: **end while**
10: **return** S'

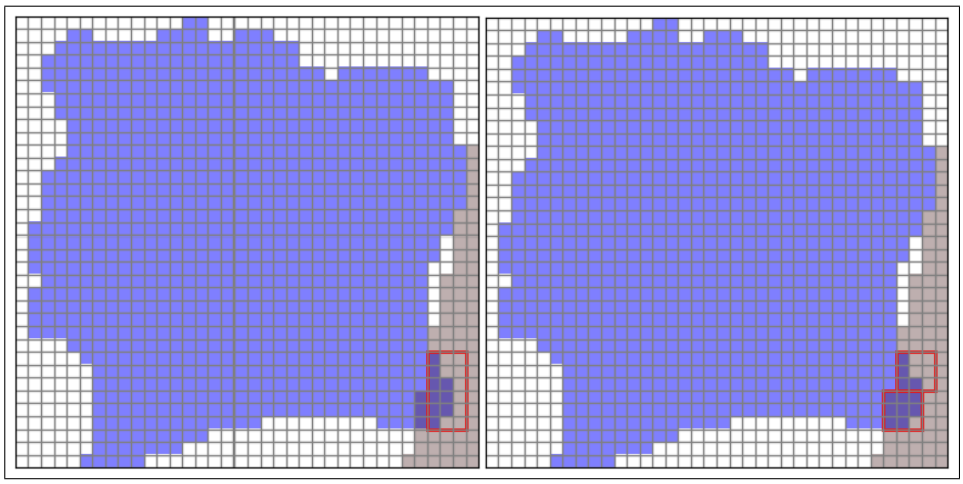

Figure A.3: Illustration of the impact of mesh splitting for the acquisition of more meshes within satellite use constraints with the same corridor acquired without mesh splitting (left) and with mesh splitting (right)

to the solution. This neighborhood function is commonly used in the knapsack problem and is easy to implement. This however presents several problems when considering the *LAC* problem.

First, the size of the instances is large enough that a search with so little guidance requires a large number of iterations to converge. Second, although a neighbor is easy to find computing its criterion requires a long time as the evaluation depends on the other meshes selected due to the possible overlaps. For these two reasons, **the convergence of this neighborhood is long when compared to the neighborhood presented in algorithm 6** which is small and searches around a local optimum. [Cheh *et al.*, 1991] compares several neighborhoods used on different problems and shows that a small neighborhood is often better as larger neighborhood spend time looking far to find good solutions.

In sections A.1 and 6.2.5 the exact approach and the greedy algorithm are presented. In this section a hybrid greedy solution using simulated annealing is proposed. Section A.4 describes the experimental use cases for the comparison of these algorithms.

A.4 Data Set

Section 6.2 describes the experimental environment for the comparison of Glimpse and the greedy algorithm. In this section the same experimental conditions are considered regarding the *MPFs* and satellites configurations. The instance pool evaluating different parameters includes:

- the number of cells using France (small instance) and Brazil (large instance),
- the weather using Myanmar with good weather and during monsoon,
- the shape of the country using Norway and Ivory Coast.

An additional emphasis on the weather randomization is considered in these experiments. Several methods of validation of the cells and meshes are possible as previously discussed in section 5.2.1. Three methods are presented in figure A.4.

- **Partial validation.** The weather for each cell is simulated using a uniform random selection between 0 and 1. If the obtained probability is inferior to the probability of validation of the cell then it is validated, otherwise it is rejected. However as is apparent in figure A.4 this approach creates many cell-sized holes that require additional meshes on future passes reacquiring already validated cells. It also doesn't consider the operational scenarios in which a cell that presents good weather is likely to be neighbored by other cells that present good weather, and vice-versa for bad weather.
- **Total validation.** Validation is done mesh by mesh instead of cell by cell, with either the whole mesh validated or rejected. The probability of validating a mesh is the sum of the probabilities of validating each included cell. A drawback of this method appears when considering crossovers. A cell covered by two meshes on the same pass can have its weather drawn for each mesh and be validated for one and rejected for the other. This leads to incoherent situations and better validation chances for cells covered by several meshes.
- The third method and the one used in the following sections is a **hybrid between partial and total validation**. In this method the cells are validated depending on the weather mesh they are included in. A random selection is performed for each weather mesh and cells are validated depending on their relative weather mesh. This solution is more spatially coherent as cells belonging to the same weather mesh are validated or rejected together if they are acquired on the same pass, and only subject to a single random selection for each pass.

Each instance was tested several times to average results obtained with different weather seeds and thus different mesh validations. As per the preliminary experiment of section 6.2.5 all experiments in section A.5 were performed on an Intel(R) Xeon(R) CPU E5-2695 v3 @ 2,30GHz two cores.

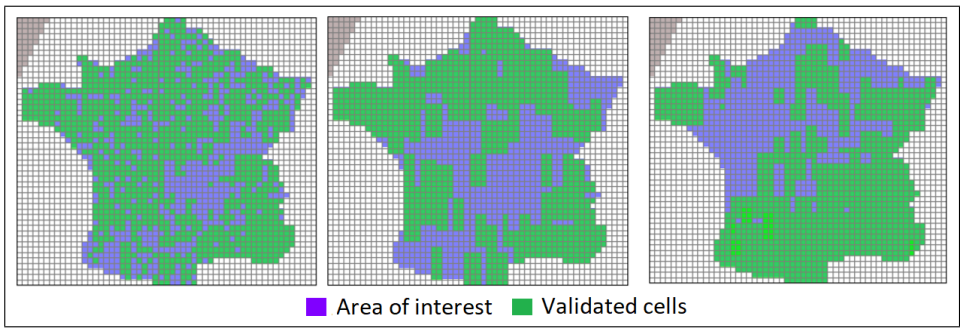

Figure A.4: Cell validation methods from left to right: partial validation, total validation and weather mesh validation

A.5 Results and Discussion

A.5.1 Area Size

The first use cases evaluated is France as a scenario with a small *AOI*. Experimental conditions are as follows:

- Cell size of 20x20km,
- Weather data from January to April 2015,
- 2 *MPFs* : one controlling two satellites of swathes 40km and respective maximum acquisition length of 80km and 120km and the other controlling to satellites of swathes 60km and respective maximum acquisition length of 120km and 160km.

Figure A.5 presents the evolution of the completion through passes during the execution of the greedy algorithm, the hybrid greedy and annealing algorithm and the CPLEX solver. The results indicate that, during the execution, the completion rate of the hybrid algorithm is on average below the optimal solution of the CPLEX solver but above the greedy algorithm.

Table A.2 compares the results of the algorithms at mission completion. The number of passes required for greedy algorithm is on average above the optimal CPLEX solution by 4.6% while the hybrid algorithm is on average above by 2.8%. While the computing time of the hybrid algorithm is above the one of the greedy algorithm, it remains reasonable when compared to the computing time of CPLEX. **The hybrid algorithm also presents completion time results closer to the optimal solution when compared to the greedy algorithm**. The standard deviation for these results is of 5 passes for each algorithm, indicating that the random weather validation element of the experiments is consistent for each algorithm.

To benchmark these results with a bigger *AOI* use case Brazil is considered. Experimental conditions are as follows:

- Cell size of 20x20km,
- Weather data from January to November 2015,

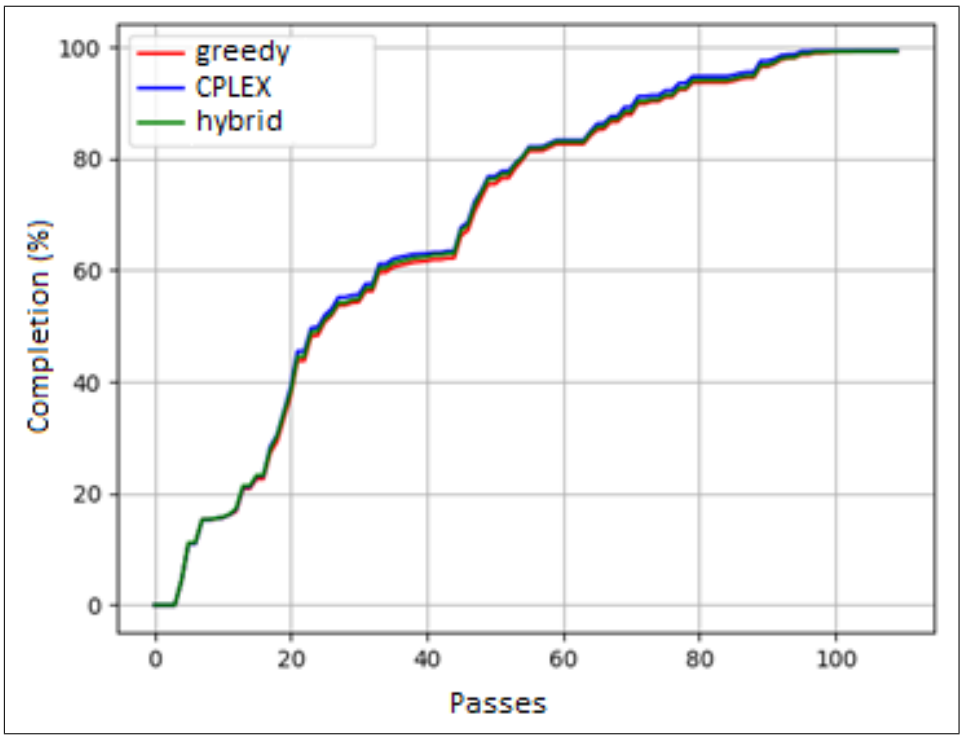

Figure A.5: Average evolution of the completion rate for the greedy, CPLEX and hybrid algorithms on the use case of France as *AOI* starting from the 01/01/15

Algorithms (km)	Passes at 95% completion	Waste (%)	Computing time
CPLEX	84,9	201	862
Greedy	89 (4,6%)	217	14
Hybrid	87,3 (2,8%)	207	153

Table A.2: Benchmark for the area size use case using France as *AOI*

- 2 *MPFs* : one controlling two satellites of swathes 40km and respective maximum acquisition length of 80km and 120km and the other controlling to satellites of swathes 60km and respective maximum acquisition length of 120km and 160km.

The scenario using France include 13 100 variables per pass while the scenario using Brazil include 68 000 variables per pass. Experimental parameters regarding the number of *MPFs*, number of satellites and characteristics of the satellites are unchanged between the scenarios using France and Brazil as *AOI*. Table A.3 presents the results of the three algorithms on the scenario using Brazil. Similar to the experiment of section 6.2.5 the CPLEX solver cannot find an optimal solution in reasonable time. The hybrid algorithm requires 4 times more computing time than the greedy algorithm but the time remains reasonable when compared to the CPLEX solver. Small improvements in both completion time and waste are obtained using the hybrid algorithm.

Algorithms (km)	Passes at 95% completion	Waste (%)	Computing time
CPLEX	-	-	>48h
Greedy	385	181	327s
Hybrid	382 (-0,7%)	178	20min8s

Table A.3: Benchmark for the area size use case using Brazil as *AOI*

A.5.2 Area Shape

To evaluate the performances of the greedy, hybrid and CPLEX algorithms regarding the area shape Norway and Ivory Coast are chosen as *AOI*. Experimental conditions are as follows:

- Cell size of 20x20km,
- Weather data from January to April 2015 for Norway, from June to November 2015 for Ivory Coast
- 2 *MPFs* : one controlling two satellites of swathes 40km and respective maximum acquisition length of 80km and 120km and the other controlling to satellites of swathes 60km and respective maximum acquisition length of 120km and 160km.

These countries are selected due to their similar size (385 203km^2 for Norway and 322 463km^2 for Ivory Coast) while having drastically different shapes with Norway being elongated and Ivory Coast more compact. These areas were also selected for the area shape scenarios of the Glimpse benchmark of section 6.4.2 with figure 6.6 illustrating the respective area shapes.

The results of table A.4 and table A.5, respectively for Norway and Ivory Coast, show how the shape of the *AOI* impact the performances of each algorithm. **For an elongated area such as Norway, the differences between each algorithm are smaller while they are more apparent for a compact area such as Ivory Coast**. This is partly due to the sameness of the solutions found for each pass when considering Norway as each corridor covers a small area with an equally small search space. The choices made by each algorithm have a higher likelihood of being the same and thus close to the optimal solution. A second explanation comes from the improved chance of having a pass that only covers already validated cells late during the execution. This extend the total time required to reach 95% acquisition as these passes count towards the overall number of necessary passes. Results are thus normalized and the differences in performance between algorithms lowered.

To consider the possibility that the results are being normalized when using Norway due to unfavorable weather additional experiments are performed on Norway at another time period. Results of Norway between June and August 2015 are presented in table A.6. The favorable weather appears to have an impact on the number of passes to completion but the gaps in results between each algorithm remains small.

Algorithms (km)	Passes at 95% completion	Waste (%)	Computing time
CPLEX	188,5	648	23min
Greedy	190 (1,1%)	636	31s
Hybrid	190,3 (1%)	625	5min34s

Table A.4: Benchmark for the area shape use case using Norway as *AOI*

Algorithms (km)	Passes at 95% completion	Waste (%)	Computing time
CPLEX	29,8	61,7%	109s
Greedy	31,2 (4,5%)	67,3%	3s
Hybrid	30,6 (2,6%)	64,9%	63s

Table A.5: Benchmark for the area shape use case using Ivory Coast as *AOI*

Algorithms (km)	Passes at 95% completion	Waste (%)	Computing time
CPLEX	127,6	734	12min18s
Greedy	127,9 (0,2%)	685	15s
Hybrid	127,8 (0,1%)	666	3min21s

Table A.6: Benchmark for the area shape use case using Norway in summer as *AOI*

A.5.3 Weather

Myanmar during monsoon is considered to compare the performances of each algorithm in unfavorable weather patterns. Experimental conditions are as follows:

- Cell size of 20x20km,
- Weather data from January to April 2015,
- 2 *MPFs* : one controlling two satellites of swathes 40km and respective maximum acquisition length of 80km and 120km and the other controlling to satellites of swathes 60km and respective maximum acquisition length of 120km and 160km.

Table A.7 presents the results and figure A.6 the evolution of the rate of completion through the passes during the execution. These results can be compared to those of a scenario with an *AOI* of similar surface area but more favorable overall weather such as those obtained in figure A.5 using France. The differences for the Myanmar instance are smaller than for the France instance, with respective 0.5% and 4.6% gaps and the shape of the evolution graph indicate a faster initial progression. These observations are explained by the relative difficulty of acquiring new surface for each pass during monsoon. Results equalize at weather bottlenecks during the mission where no mesh can be acquired for long periods, leaving time for slower algorithms to catch up and stabilize at the same overall surface acquired.

Algorithms (km)	Passes at 95% completion	Waste (%)	Computing time
CPLEX	398	1042	1h37min
Greedy	400 (0,5%)	996	68s
Hybrid	399 (0,25%)	967	9min17s

Table A.7: Benchmark for the area shape use case using Myanmar during monsoon as *AOI*

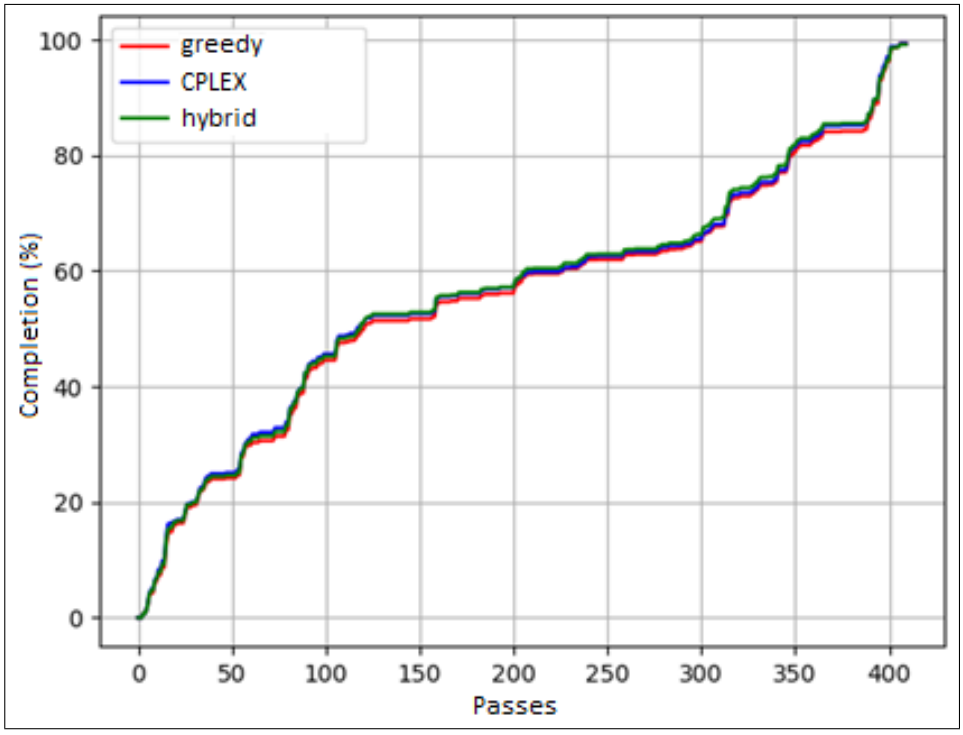

Figure A.6: Average evolution of the completion rate for the greedy, CPLEX and hybrid algorithms on the use case of Myanmar during monsoon as *AOI*

The observed waste is high due to the unsuccessful acquisition and multiple reacquisition of cells. **A small improvement in terms of waste can be observed when using the hybrid algorithm over the greedy algorithm**. Figure A.7 presents the evolution of the waste when using CPLEX for the acquisition of Myanmar during monsoon and illustrates the linear difficulty of successful acquisition through the mission.

A.6 Discussion

Results in this section show the importance of metaheuristics in solving the *LAC* problem. The proposed greedy and hybrid algorithms obtain results that are close to the exact solver CPLEX in regards to the completion time on small instances as shown in sections A.5.1 and A.5.2 while being much faster in computing time. **The number of passes to complete a small scale *LAC* mission never exceeds 5% difference to CPLEX for the greedy algorithm, and 3% for the hybrid algorithm**. The waste is dependant on the instance considered, with no clear algorithm performing better than all others.

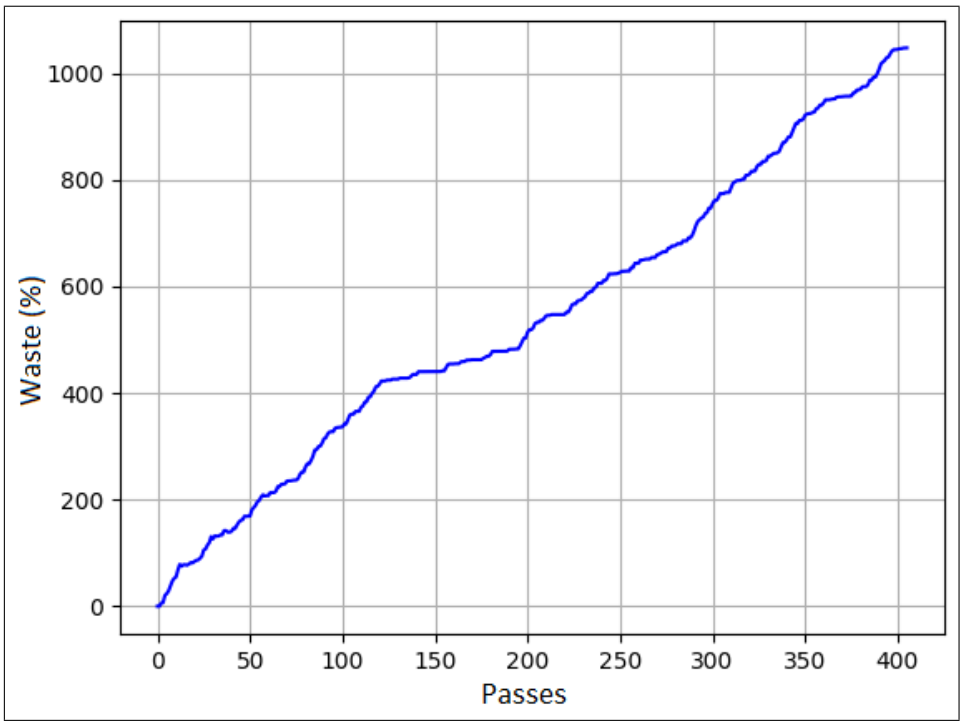

Figure A.7: Average evolution of the waste for the CPLEX algorithm on the use case of Myanmar during monsoon as *AOI*

Larger instances such as the scenario of Brazil presented in section A.5.1 show that the CPLEX solver is not adequate for these instances of the *LAC* problem. Indeed the computation time exceeds several days while the metaheuristics represented by the greedy and hybrid algorithms remain under 15 minutes necessary which is considered reasonable. **The exponential computing time of the CPLEX solver quickly becomes too high for operational use**. In an operational context launching the *LARM* algorithm several times may be required due to weather updates. This highlights the importance of efficient computing time. Furthermore a more precise grid cell also greatly impacts the size of the instance and variables for the CPLEX solver, which makes the solving of a single pass difficult to achieve in reasonable time using an exact approach.

The importance of metaheuristics is also shown through the study of different shaped *AOI*. For example when considering the elongated shape of Norway (section A.5.2) the corridors often intersect only a small subarea of the *AOI* which lowers the search space and normalizes the results between optimization algorithms. This is also the case for Myanmar during monsoon season (section A.5.3), this time due to unfavorable weather and the difficulty of acquisition on a significant number of passes. This is verified by the waste being as high as 1000%, meaning that a single cell needs to be acquired 11 times to be validated on average.

The obtained results also show that the maximum observable differences in performance between the CPLEX, greedy and hybrid algorithms during the solve are often found at around 50 to 80% mission completion. This can be explained by gradual better optimization choices normalizing after reaching local weather bottlenecks during the

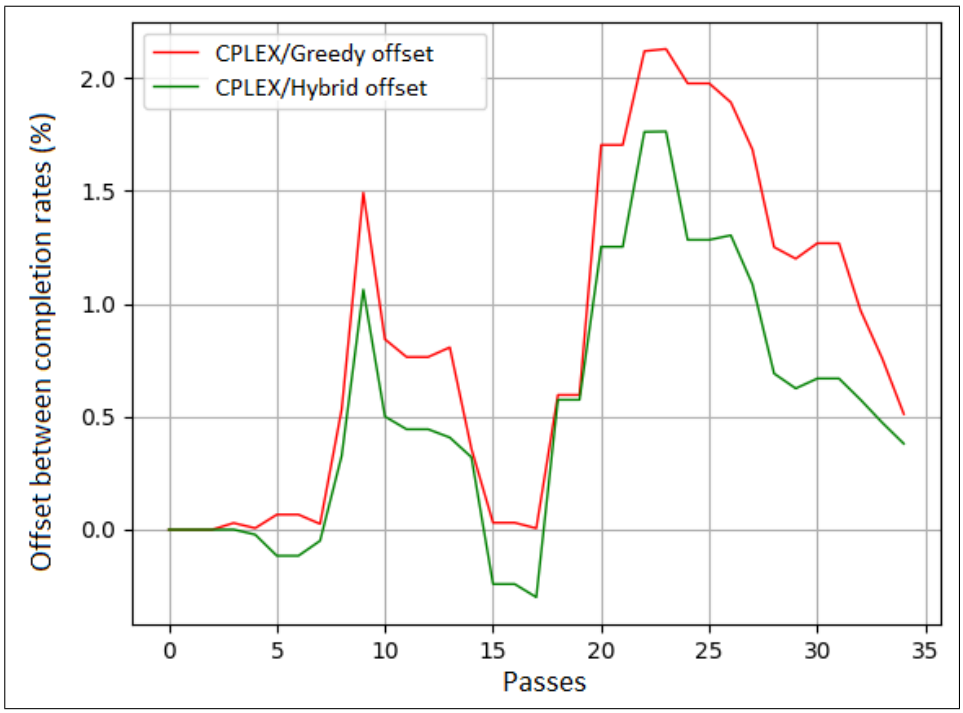

Figure A.8: Comparison of the offsets for the completion rate between CPLEX and the greedy algorithm and between CPLEX and the hybrid algorithm during the execution of an Ivory Coast *LAC* mission

mission, similar to those reached in the case of Myanmar during monsoon. As an example the hybrid algorithm may have validated all France except a specific subarea during the first 60 passes while the greedy algorithm required 65 passes, but both algorithm then need to wait until the 80th pass to acquire the specific subarea and complete the mission. This phenomenon explains the relative small differences in performance gaps between algorithms at 95% completion when compared to those observed at 50-80% completion.

Figure A.8 presents the differences in completion rates between the CPLEX and greedy algorithms and the CPLEX and hybrid algorithms using the scenario of Ivory Coast. The evolution observed confirms that the gaps in performances gradually lower towards the end where the two metaheuristics algorithms acquire surfaces that have already been acquired by CPLEX while CPLEX waits for area with unfavorable weather on average to be acquirable.

An observation applicable to each experiment using the greedy and hybrid algorithm is the consistently improved performances of the hybrid algorithm throughout the missions. Any client update performed before the end of the mission is then guaranteed to have better completion rate when using the hybrid algorithm over the greedy algorithm. Although the hybrid algorithm always improves on the results obtained by the greedy algorithm the margins are small in relation to the optimal solution. Such is the case for Norway in section A.5.2 where the hybrid algorithm only performs 0.1% better than the greedy algorithm while being 1% away from the optimal solution.

For other examples such as Ivory Coast, the improvement is of almost 50% (the hybrid algorithm being 2.6% away from optimal and the greedy algorithm being 4.5% away from

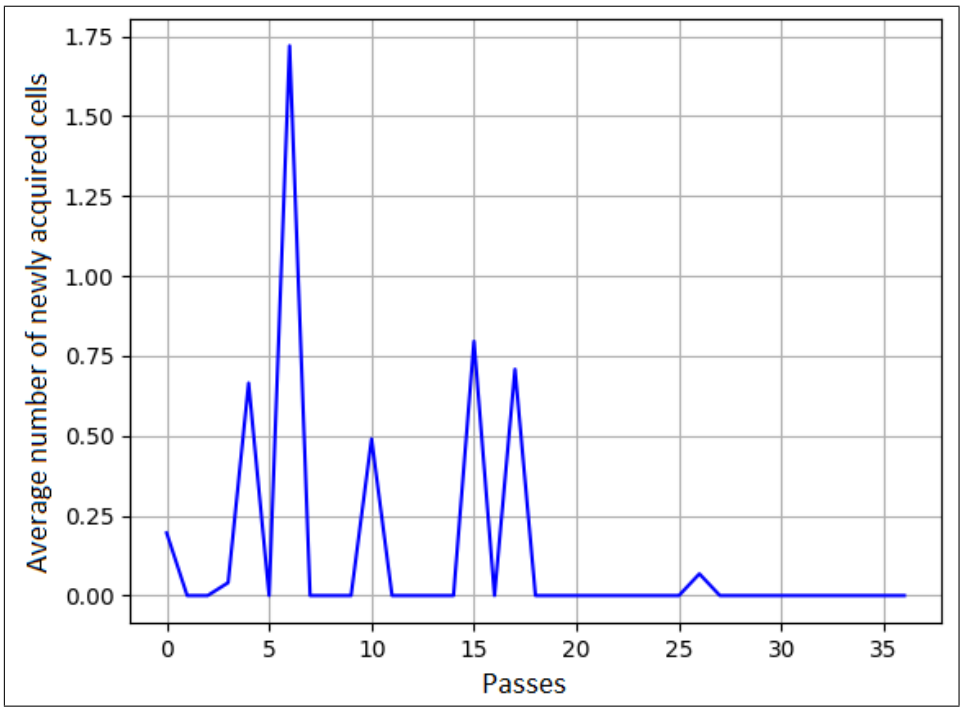

Figure A.9: Improvements of the simulated annealing over the greedy solution in average of newly acquired cells per pass on the use case of Ivory Coast

optimal) but the margin relative to the optimal solution remains significant. These cases can be explained by the relatively low number of iterations in the annealing part of the hybrid algorithm (NB_TRANSITIONS in the cooling loop is inferior to 1000) as the computing time of metaheuristics is limited by several factors:

- **On large instances the computing time is limited by operational constraints and the algorithms need to run in reasonable time depending on the number of satellites used**. If the mission only includes one satellite which only has visibility on the *AOI* every 24h an algorithm that requires several hours of computing time is sufficient. However if 20 satellites are used the algorithm needs to solve the problem in minutes instead.

- **On smaller instances the computing time is limited by the exact solution found by the CPLEX solver**. Using metaheuristics is justified if the solution is found in lower time than the time necessary for an exact solver to find an optimal solution. As such NB_TRANSITIONS cannot be set arbitrarily high or the hybrid algorithms risks being slower than the CPLEX solver.

In addition it may be counterproductive to augment the number of iterations in the cooling loop as it now always possible to improve on the solution found by the greedy algorithm. Indeed figure A.9 presents the improvement found by the simulated annealing part of the hybrid algorithm over the greedy algorithm for each pass and a number of passes can be observed for which the solution was not improved.

A.7 Conclusion

In this annex an improved benchmark for the sake of future comparisons with Glimpse is proposed. A first contribution is the formalization of the *LAC* problem as an *ILP* problem. Using the exact solver CPLEX optimal solutions can be found on small *LAC* scenarios. The exponential growth in complexity of the *LAC* problem is apparent through testing as larger *LAC* scenarios cannot be solved within reasonable time by CPLEX. Exact solutions found by CPLEX provide a reference for candidate optimization algorithms when considering small scale *LAC* instances.

An improved greedy algorithm is then presented. Using the output of a hierarchical greedy algorithm as input for a simulated annealing algorithm is proposed as a solution to the local minima trapping problem encountered by the greedy algorithms. The resulting hybrid algorithm uses elementary movements of a cell's length in the neighborhood of the meshes to explore the search tree. An alternative search that uses addition and suppression of meshes to a corridor is also evaluated. The mesh movement neighborhood search is preferred as the addition and suppression of meshes tend to create large neighborhoods and create overlaps between existing and added meshes.

Finally the greedy algorithm, the hybrid algorithm and CPLEX are compared. The robustness criteria of section 6.4.1 and similar experimental use cases are considered for this benchmark. The results indicate small but consistent improvements in both completion time and waste when using the hybrid algorithm over the greedy algorithm. The experiments also show the importance of metaheuristics for the solving of the larger *LAC* instances for which CPLEX is unable to find solutions in reasonable time. A future continuation of this work is the benchmark of Glimpse with the hybrid algorithm proposed in this annex and other candidate optimization algorithms.

Personal Bibliography

Timothée JAMMOT, Elsy KADDOUM, Serge RAINJONNEAU, Mathieu PICARD and Marie-Pierre GLEIZES: Dynamic meshing for satellite acquisition of large areas using adaptive multi-agent system. *In 11th International Workshop on Planning and Scheduling for Space (IWPSS 2019)*, 2019.

Timothée JAMMOT, Elsy KADDOUM, Serge RAINJONNEAU, Mathieu PICARD and Marie-Pierre GLEIZES: Maillage dynamique pour l'acquisition de grandes couvertures par système multi-agent coopératif. *In 14èmes Journées Planification, la Décision et l'Apprentissage pour la conduite de systèmes (JFPDA 2019@ PFIA)*, 2019.

Timothée JAMMOT, Elsy KADDOUM, Serge RAINJONNEAU and Mathieu PICARD: Large area coverage using adaptive and robust multi-agent system. *In International Astronautical Federation 2020 (IAC 2020)*, IAC-20,B1,4,12,x58180, 2020.

Timothée JAMMOT: Dynamic meshing for satellite acquisition of large areas using an adaptive multi-agent system. Technical report for the final progress meeting of the SYNAPSE project at IRT Saint-Exupéry, 2020.

Bibliography

Thomas BARTZ-BEIELSTEIN, Jürgen BRANKE, Jörn MEHNEN and Olaf MERSMANN: Evolutionary algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(3):178–195, 2014.

Martin BEER, Mark D'INVERNO, Michael LUCK, Nick JENNINGS, Chris PREIST and Michael SCHROEDER: Negotiation in multi-agent systems. *The Knowledge Engineering Review*, 14 (3):285–289, 1999.

Zahra BEHESHTI and Siti Mariyam Hj SHAMSUDDIN: A review of population-based meta-heuristic algorithms. *Int. J. Adv. Soft Comput. Appl*, 5(1):1–35, 2013.

E. BENSANA, M. LEMAITRE and G. VERFAILLIE: Earth observation satellite management. *Constraints*, 4(3):293–299, 1999.

E BENSANA, G VERFAILLIE, JC AGNÈSE, N BATAILLE and D BLUMSTEIN: Exact and approximate methods for the daily management of an earth observation satellite. *In Proc. of the 4 th Int. Symposium on Space Mission Operations and Ground Data Systems, Munich, Germany*, 1996.

C. BERNON, M.-P. GLEIZES, S. PEYRUQUEOU and G. PICARD: Adelfe: a methodology for adaptive multi-agent systems engineering. *In International Workshop on Engineering Societies in the Agents World*, pages 156–169. Springer, 2002.

Carole BERNON, Valérie CAMPS, Marie-Pierre GLEIZES and Gauthier PICARD: Engineering adaptive multi-agent systems: The adelfe methodology. *In Agent-oriented methodologies*, pages 172–202. IGI Global, 2005.

J BONNET: *Multi-Criteria and Multi-Objective Dynamic Planning by Self-Adaptive Multi-Agent System, Application to Earth Observation Satellite Constellations*. PhD Thesis, Université Paul Sabatier-Toulouse III, 2017.

Christine BOURJOT, Vincent CHEVRIER and Vincent THOMAS: A new swarm mechanism based on social spiders colonies: from web weaving to region detection. *Web Intelligence and Agent Systems: An International Journal*, 1(1):47–64, 2003.

MJ BOX: A new method of constrained optimization and a comparison with other methods. *The Computer Journal*, 8(1):42–52, 1965.

Stephen BOYD and Jacob MATTINGLEY: Branch and bound methods. *Notes for EE364b, Stanford University*, 2006:07, 2007.

Nicolas BRAX: *Self-adaptive multi-agent systems for aided decision-making: an application to maritime surveillance*. PhD Thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2013.

Zhongxiang CHANG, Zhongbao ZHOU, Ruiyang LI, Helu XIAO and Lining XING: Observation scheduling for a state-of-the-art sareos: Two adaptive multi-objective evolutionary algorithms. *Computers & Industrial Engineering*, 169:108252, 2022.

Kah Mun CHEH, Jeffrey B GOLDBERG and Ronald G ASKIN: A note on the effect of neighborhood structure in simulated annealing. *Computers & Operations Research*, 18(6):537–547, 1991.

CH CHENG, BR FEIRING and TCE CHENG: The cutting stock problemâa survey. *International Journal of Production Economics*, 36(3):291–305, 1994.

Xiaogeng CHU, Yuning CHEN and Yuejin TAN: An anytime branch and bound algorithm for agile earth observation satellite onboard scheduling. *Advances in Space Research*, 60(9):2077–2090, 2017. ISSN 0273-1177. URL `https://www.sciencedirect.com/science/article/pii/S0273117717305367`.

Kalyanmoy DEB, Amrit PRATAP, Sameer AGARWAL and TAMT MEYARIVAN: A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

Augustin DEGAS: *Auto-structuration de trafic temps-réel multi-objectif et multi-critère dans un monde virtuel*. PhD Thesis, Université Paul Sabatier-Toulouse III, 2020.

Erik DEMEULEMEESTER and Willy HERROELEN: A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management science*, 38(12): 1803–1818, 1992.

Arnaud DONIEC, Stéphane ESPIÉ, René MANDIAU and Sylvain PIECHOWIAK: Dealing with multi-agent coordination by anticipation: Application to the traffic simulation at junctions. *EUMAS*, 5(2005):478–479, 2005.

Adejuyigbe O FAJEMISIN, Steven D PRESTWICH and Laura CLIMENT: Cutting uncertain stock and vehicle routing in a sustainability forestry harvesting problem. *Top*, 31(1):139–164, 2023.

W. FEI and L. ZHI: Research on the optimization method of dynamic partitioning area target for earth observation satellites. *Procedia Engineering*, 15:3159–3163, 2011.

J. FRANK, A. JONSSON, R. MORRIS, D.-E. SMITH and P. NORVIG: Planning and scheduling for fleets of earth observing satellites. 2001.

Jean-Pierre GEORGÉ, Marie-Pierre GLEIZES and Pierre GLIZE: Conception de systèmes adaptatifs à fonctionnalité émergente : la théorie des AMAS. *RIA*, 17(4):591–626, 2003.

Marie-Pierre GLEIZES, Valérie CAMPS, Jean-Pierre GEORGÉ and Davy CAPERA: Engineering systems which generate emergent functionalities. *In Engineering Environment-Mediated Multi-Agent Systems: International Workshop, EEMMAS 2007, Dresden, Germany, October 5, 2007. Selected Revised and Invited Papers*, pages 58–75. Springer, 2008.

Marie-Pierre GLEIZES, Valérie CAMPS and Pierre GLIZE: A theory of emergent computation based on cooperative self-organization for adaptive artificial systems. *In Fourth European Congress of Systems Science*, pages 20–24, 1999.

Pierre GLIZE: L'adaptation des systèmes à fonctionnalité émergente par auto-organisation coopérative. *Hdr, Université Paul Sabatier, Toulouse III*, 2001.

A. GLOBUS, J. CRAWFORD, J. LOHN and A. PRYOR: A comparison of techniques for scheduling earth observing satellites. *In AAAI*, pages 836–843, 2004.

Jack E GRAVER: On the foundations of linear and integer linear programming i. *Mathematical Programming*, 9(1):207–226, 1975.

Davide GUASTELLA: *Dynamic learning of the environment for eco-citizen behavior*. PhD Thesis, Université Paul Sabatier-Toulouse III; Università degli studi (Catane, Italie), 2020.

Adrien HADJ-SALAH, Rémi VERDIER, Clément CARON, Mathieu PICARD and Mikaël CAPELLE: Schedule earth observation satellites with deep reinforcement learning. *arXiv preprint arXiv:1911.05696*, 2019.

Nicholas G HALL and Michael J MAGAZINE: Maximizing the value of a space mission. *European journal of operational research*, 78(2):224–241, 1994.

Chao HAN, Yi GU, Guohua WU and Xinwei WANG: Simulated annealing-based heuristic for multiple agile satellites scheduling under cloud coverage uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2022.

N. HOLVOET, W. VONGSANTIVANICH, S. CHAIMATANAN and D. DELAHAYE: Mission planning for a non-homogeneous earth observation satellite constellation for disaster response. *In SpaceOps 2018*, 2018.

Elsy KADDOUM: Optimization under constraints of distributed complex problems using cooperative self-organization. *Organization*, 2011.

R Baker KEARFOTT: An interval branch and bound algorithm for bound constrained optimization problems. *Journal of Global Optimization*, 2(3):259–280, 1992.

S. KIRKPATRICK, C.-D. GELATT and M.-P. VECCHI: Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

Renaud LACOUR: *Exact and approximate solving approaches in multi-objective combinatorial optimization, application to the minimum weight spanning tree problem*. PhD Thesis, Université Paris Dauphine-Paris IX, 2014.

Eugene L LAWLER and David E WOOD: Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966.

Michel LEMAÎTRE, Gérard VERFAILLIE, Frank JOUHAUD, Jean-Michel LACHIVER and Nicolas BATAILLE: How to manage the new generation of agile earth observation satellites. *In Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space*, pages 1–10. Citeseer, 2000.

Feng LI, Qiuhua WAN, Qien HE, Xing ZHONG, Kai XU and Ruifei ZHU: An improved many-objective evolutionary algorithm for multi-satellite joint large regional coverage. *IEEE Access*, 2023.

Zhiliang LI and Xiaojiang LI: A multi-objective binary-encoding differential evolution algorithm for proactive scheduling of agile earth observation satellites. *Advances in Space Research*, 63(10):3258–3269, 2019.

Yan LIU, Shengyu ZHANG and Haiying HU: A simulated annealing algorithm with tabu list for the multi-satellite downlink schedule problem considering waiting time. *Aerospace*, 9 (5):235, 2022.

W.G. MACREADY and D.H. WOLPERT: Bandit problems and the exploration/exploitation tradeoff. *IEEE Transactions on Evolutionary Computation*, 2(1):2–22, 1998.

Dalal MADAKAT, Jérôme MORIO and Daniel VANDERPOOTEN: A biobjective branch and bound procedure for planning spatial missions. *Aerospace Science and Technology*, 73:269–277, 2018.

Catherine MANCEL: *Modélisation et résolution de problèmes d'optimisation combinatoire issus d'application spatiales*. PhD Thesis, 2004.

René MANDIAU, Alexis CHAMPION, Jean-Michel AUBERLET, Stéphane ESPIÉ and Christophe KOLSKI: Behaviour based on decision matrices for a coordination between agents in a urban traffic simulation. *Applied Intelligence*, 28:121–138, 2008.

Mohamed AA MANSOUR and Maged M DESSOUKY: A genetic algorithm approach for solving the daily photograph selection problem of the spot5 satellite. *Computers & Industrial Engineering*, 58(3):509–520, 2010.

Silvano MARTELLO and Paolo TOTH: *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

J. MCCALL: Genetic algorithms for modelling and optimisation. *Journal of Computational and Applied Mathematics*, 184(1):205–222, 2005.

Georges MYKONIATIS, Laurent LAPASSET and Andrija VIDOSAVLJEVIC: An adaptative multi-agent model of delays propagation in the air traffic system. *In 7th International Conference on Experiments/Process/System Modeling/Simulation/Optimization (7th IC-EpsMsO)*, 2017.

Patrenahalli M. NARENDRA and Keinosuke FUKUNAGA: A branch and bound algorithm for feature subset selection. *IEEE Transactions on computers*, 26(09):917–922, 1977.

Bobby NEJAD: The mission planning facility. *In Introduction to Satellite Ground Segment Systems Engineering: Principles and Operational Aspects*, pages 139–150. Springer, 2022.

Aneta NEUMANN, Yue XIE and Frank NEUMANN: Evolutionary algorithms for limiting the effect of uncertainty for the knapsack problem with stochastic profits. *In International Conference on Parallel Problem Solving from Nature*, pages 294–307. Springer, 2022.

Xiaonan NIU, Hong TANG and Lixin WU: Satellite scheduling of large areal tasks for rapid response to natural disaster using a multi-objective genetic algorithm. *International journal of disaster risk reduction*, 28:813–825, 2018a.

XN NIU, H TANG and LX WU: Multi-satellite observation scheduling for large area disaster emergency response. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42(3), 2018b.

Simon PARSONS and Michael WOOLDRIDGE: Game theory and decision theory in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 5:243–254, 2002.

Alexandre PERLES: *An adaptive multi-agent system for the distribution of intelligence in electrical distribution networks: state estimation*. PhD Thesis, Université Paul Sabatier-Toulouse III, 2017.

Alexandre PERLES, Fabrice CRASNIER and Jean-Pierre GEORGÉ: Amak-a framework for developing robust and open adaptive multi-agent systems. *In International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 468–479. Springer, 2018.

Gauthier PICARD and Marie-Pierre GLEIZES: The adelfe methodology. *In Methodologies and Software Engineering for Agent Systems: The Agent-oriented Software Engineering Handbook*, pages 157–175. Springer, 2004.

Tim RENTSCH: Object oriented programming. *ACM Sigplan Notices*, 17(9):51–57, 1982.

Sylvain ROUGEMAILLE, Frédéric MIGEON, Christine MAUREL and Marie-Pierre GLEIZES: Model driven engineering for designing adaptive multi-agents systems. *In Engineering Societies in the Agents World VIII: 8th International Workshop, ESAW 2007, Athens, Greece, October 22-24, 2007, Revised Selected Papers 8*, pages 318–332. Springer, 2008.

Hanif D SHERALI and Patrick J DRISCOLL: Evolution and state-of-the-art in integer programming. *Journal of Computational and Applied Mathematics*, 124(1-2):319–340, 2000.

Maciej ŚWIECHOWSKI, Konrad GODLEWSKI, Bartosz SAWICKI and Jacek MAŃDZIUK: Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562, 2023.

Juan TERÁN, José L AGUILAR and Mariela CERRADA: Mathematical models of coordination mechanisms in multi-agent systems. *CLEI electronic journal*, 16(2):5–5, 2013.

Nicolas VERSTAEVEL, Christine RÉGIS, Marie-Pierre GLEIZES and Fabrice ROBERT: Principles and experimentations of self-organizing embedded agents allowing learning from demonstration in ambient robotics. *Future Generation Computer Systems*, 64:78–87, 2016.

Zhu WAIMING, Hu XIAOXUAN, Xia WEI and Jin PENG: A two-phase genetic annealing method for integrated earth observation satellite scheduling problems. *Soft Computing*, 23:181–196, 2019.

Guohua WU, Huilin WANG, Witold PEDRYCZ, Haifeng LI and Ling WANG: Satellite observation scheduling with a novel adaptive simulated annealing algorithm and a dynamic task clustering strategy. *Computers & Industrial Engineering*, 113:576–588, 2017.

Y. XU, X. LIU, R. HE, Y. CHEN and Y. CHEN: Multi-objective satellite scheduling approach for very large areal observation. *In IOP Conference Series: Materials Science and Engineering*, volume 435, page 012037. IOP Publishing, 2018.

Yingjie XU, Xiaolu LIU, Renjie HE and Yingguo CHEN: Multi-satellite scheduling framework and algorithm for very large area observation. *Acta Astronautica*, 167:93–107, 2020a. ISSN 0094-5765. URL https://www.sciencedirect.com/science/article/pii/S0094576519313694.

Yingjie XU, Xiaolu LIU, Renjie HE and Yingguo CHEN: Multi-satellite scheduling framework and algorithm for very large area observation. *Acta Astronautica*, 167:93–107, 2020b.

List of Figures

List of Tables

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

Université de Toulouse

Titre : Maillage dynamique pour l'acquisition de grandes couvertures par systèmes multi-agents adaptatifs

Mots clés : Optimisation, Systèmes multi-agents, Observation de la Terre

Résumé : Le domaine de l'observation de la Terre regroupe différents types d'appareils pour la prise de vue ou acquisition d'images terrestres. Les satellites d'observation en orbite basse sont conçus pour prendre des photographies haute résolution depuis l'espace. Ils sont utilisés pour des missions d'observation urgentes mais aussi pour des missions long terme dites grandes couvertures. Dans ce cadre une grande zone est photographiée progressivement à chaque passage de satellite qui acquiert des zones au sol dites mailles. Au terme de la mission les images résultant des mailles sont regroupées pour former la grande couverture.
En raison de la taille de la zone à acquérir les missions de grandes couvertures font appel à plusieurs satellites. La zone doit être acquise par ces satellites de façon à minimiser le temps de complétion de la mission et à maîtriser l'utilisation des ressources satellitaires. Le problème d'optimisation obtenu est complexe. De nombreux facteurs rendent la résolution en temps raisonnable difficile, comme la taille de la zone, l'hétérogénéité des caractéristiques des satellites et les prévisions météorologiques. L'utilisation d'un algorithme d'optimisation en méthode approchée est ainsi nécessaire.
Le grand nombre d'entités du problème et le besoin d'adaptabilité justifient l'utilisation d'un algorithme par raisonnement décentralisé. Dans cette thèse nous adressons la résolution du problème des grandes couvertures par systèmes multi-agents adaptatifs (AMAS).
Nous proposons d'abord une nouvelle méthode de maillage, le maillage dynamique, qui permet un positionnement précis des mailles sur la zone, et une formalisation du problème des grandes couvertures. Pour résoudre ce problème nous présentons le système Glimpse développé basé sur le modèle agent AMAS4Opt. Glimpse est ensuite validé sur divers scénarios simulés.
Une comparaison avec un algorithme glouton illustre les apports de notre méthode pour l'optimisation des métriques du problème. Glimpse réduit l'utilisation de ressources satellitaires en évitant les acquisitions superflues tout en minimisant le temps de complétion requis pour traiter de très grandes couvertures. Les résultats obtenus ouvrent de nouvelles directions de recherche dans les domaines des AMAS et des grandes couvertures.

Title: Dynamic meshing of large area coverage using adaptive multi-agent systems

Key words: Optimization, Multi-agent systems, Earth observation

Abstract: In the Earth observation field a variety of devices are used to take pictures or acquire ground images. Low Earth orbit satellites are designed to acquire high resolution images from space. They are used for urgent missions but also long term ones called large area coverage. In this context a large area is acquired gradually for each pass of satellite that acquire small surface areas called meshes. At the end of the mission the images acquired from the meshes are regrouped to form the large coverage.
Due to the size of the area to acquire large area coverage missions require the use of sev- eral satellites. The area needs to be acquired by these satellites so as to minimize the time to mission completion and optimize the use of satellite resources. The resulting optimiza- tion problem is complex. Many parameters render the solve in reasonable time difficult, such as the size of the area, the heterogeneity of the satellite characteristics and the weather forecasts. The use of an approximate optimization method is necessary.
The high number of problem entities and the need for adaptability justify the use of a decentralized reasoning algorithm. In this thesis we address the solve of the large area cov- erage problem by adaptive multi-agent systems (AMAS). We first propose a new meshing technique, dynamic meshing, that allows for a precise positioning of meshes on the area, and a formalization of the large area coverage problem. To solve this problem we present the Glimpse AMAS developed using the AMAS4Opt agent model.
Glimpse and dynamic meshing are then validated on different simulated scenarios. A comparison with a greedy algorithm illustrates the contributions of our method for opti- mizing the problem metrics. Glimpse reduces the use of satellite resources by avoiding superfluous acquisitions while minimizing the time required to acquire very large areas. The results open new research directions in the AMAS and large area coverage fields.