Rendu - Coordination de satellites

Projet COCOMA

Encadrants :

Nicolas Maudet

Equipe :

Rida Taleb
Jules Cassan

Aurélie Beynier

Sommaire

1 Introduction

L'article intitulé *"Auction-based and Distributed Optimization Approaches for Scheduling Observations in Satellite Constellations with Exclusive Orbit Portions"*, rédigé par Gauthier Picard, a été publié dans les actes de la 21e Conférence Internationale sur les Agents Autonomes et les Systèmes Multiagents (AAMAS 2022). Il aborde la question complexe de la coordination entre les utilisateurs ayant réservé des segments exclusifs d'orbite et un planificateur central qui doit traiter plusieurs demandes pouvant empiéter sur ces zones d'exclusivité. L'auteur nomme ce défi le Problème de Planification de Constellation de Satellites d'Observation de la Terre (Earth Observation Satellite Constellation Scheduling Problem, EOSCSP) et propose une formulation sous forme de programme linéaire mixte en nombres entiers pour y répondre.

L'article met en lumière les défis associés à l'exploitation des nombreux satellites d'observation de la Terre (EOS) et souligne l'importance cruciale d'améliorer la coopération entre les ressources disponibles ainsi que l'autonomie à bord pour optimiser l'efficacité du système. L'auteur met en avant que les progrès technologiques récents ont permis la conception et le déploiement de satellites EOS agiles, dotés de la capacité de modifier leur orientation et d'offrir une variété de prises de vue en utilisant divers capteurs. Bien que cette évolution ouvre la voie à des services plus riches pour un éventail plus large d'utilisateurs, elle introduit également un nombre accru de degrés de liberté et de variables décisionnelles dans la planification des activités des EOS. Cette complexité accrue pose d'importants défis, notamment sous l'angle des systèmes multi-agents, nécessitant des solutions innovantes pour la coordination et la gestion optimales des ressources satellites.

L'article propose des techniques basées sur le marché et une technique de résolution de problèmes distribuée basée sur l'optimisation des contraintes distribuées (DCOP), où les agents coopèrent pour allouer les demandes sans partager leurs calendriers. Les contributions sont évaluées expérimentalement sur des instances EOSCSP générées aléatoirement et basées sur des carnets d'ordres d'observation réels à grande échelle ou hautement conflictuels.

Le fait de considérer les constellations de satellites comme des ressources partagées nécessitant la coordination de plusieurs utilisateurs pour l'attribution de tâches d'observation dans des portions d'orbite exclusives est un problème totalement nouveau. L'auteur compare l'approche proposée avec les approches basées sur les enchères proposées, où chaque satellite est géré par un centre de mission différent, et où les centres de mission coordonnent leur allocation à l'aide d'enchères, en faisant des offres sur les observations ouvertes en fonction de l'impact sur le plan de bord et de sa récompense.

2 Modelisation des instances ESOP

Pour initier ce projet, la première étape a consisté à développer un **générateur d'instances aléatoires**, qui seraient par la suite traitées par nos différents algorithmes de résolution. Nous avons opté pour une représentation sous forme de dictionnaire, jugée comme étant la plus simple et la plus claire pour notre équipe. Cette structure offre également la facilité de convertir les données en format JSON, ce qui s'avère pratique pour archiver les instances générées. Chaque instance est définie par des informations clés : **les utilisateurs, les satellites, et les requêtes, ces dernières incluant toutes les observations possibles**.

Afin d'améliorer notre compréhension des instances les plus simples et de faciliter le débogage des solveurs, nous avons implémenté une représentation graphique inspirée de celle présentée dans l'article de référence. Cette visualisation graphique constitue un outil précieux pour analyser et affiner les performances de nos algorithmes de manière plus efficace. Voici un exemple des instances pouvant être généré :

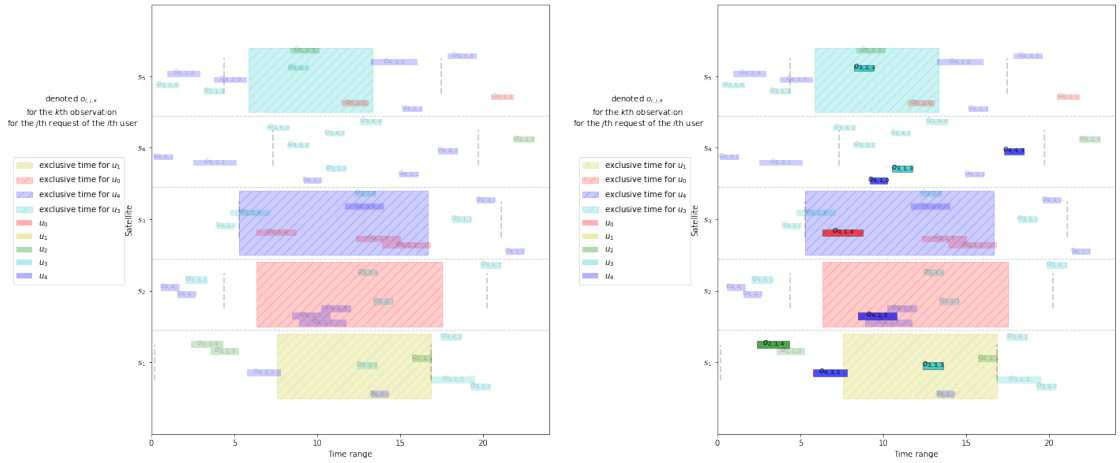

FIGURE 1 – Exemple d'instance ESOP généré - instance ESOP résolue (PSI)

L'instance générée, telle que visualisé ci-dessus, se déploie sur une période de 24 unités de temps, initialement envisagée en heures. Elle comprend 5 satellites et 5 utilisateurs, dont 4 bénéficient de plages horaires exclusives. Au total, 10 requêtes doivent être satisfaites, chaque requête comportant 5 observations possibles. Cette configuration permet de visualiser un environnement opérationnel complexe. Elle ne sert que de base pour essayer des instances plus simples et voir si tout fonctionne avant de monter en complexité.

3 Partie 1 : Optimisation de contraintes distribuées

3.1 Transformation en instances DCOP

Dans cette section du projet, notre objectif est d'aborder le problème de planification en le conceptualisant comme un Problème d'Optimisation sous Contraintes Distribuées (Distributed Constraint Optimization Problem, DCOP). Afin d'atteindre cet objectif, nous avons choisi d'exploiter une bibliothèque Python spécialement conçue pour résoudre les DCOP, qui inclut déjà une panoplie d'algorithmes de résolution prêts à l'emploi.

Pour assurer la compatibilité de notre instance avec cette bibliothèque, il a été nécessaire d'adapter notre modèle à un format différent, à savoir un fichier YAML. Ce fichier doit respecter une syntaxe spécifique définie par PyDCOP, la bibliothèque en question. Cette étape de conversion est cruciale car elle permet de transformer nos données initiales en une structure que la bibliothèque peut interpréter et manipuler efficacement pour trouver une solution optimale à notre problème de planification.

Les contraintes de différence sont établies entre les observations en conflit, spécifiant des fonctions associant des récompenses négatives en cas de conflit. Parallèlement, des préférences sont définies pour les observations prioritaires, attribuant des récompenses positives en cas de respect de ces préférences. Ces éléments sont exprimés dans le fichier YAML résultant, prêt à être utilisé par PyDCOP.

3.2 Comparaison des algorithmes

Au cours de l'implémentation de notre solution avec PyDcop, il semble que nous ayons commis **une erreur critique**. Bien que les résultats obtenus ne soient pas catastrophiques, **la complexité de la résolution est telle que nous n'avons pas été en mesure de tester l'algorithme sur des instances complexes**, comme pour d'autres algorithmes vus plus loin dans ce rapport (le temps de résolution était bien trop long même pour de petite instance). Cette limitation a entravé notre capacité à évaluer pleinement l'efficacité et l'efficience de l'algorithme dans des scénarios plus exigeants et diversifiés.

Nous avons donc mené la même expérience en utilisant les différents algorithmes de résolution de DCOP implémentés par la librairie PyDcop. Les instances créées pour cet essai se situent sur des plages horaires réduites de 24 unités de temps, avec un nombre de tâches variant de [5 à 50, avec des incréments de 5], impliquant 5 satellites, 5 agents, et 3 zones exclusives. Pour garantir la fiabilité de nos résultats, chaque configuration d'expérience a été répétée 10 fois. Cela nous a permis d'obtenir une moyenne des performances sur l'ensemble des essais, offrant ainsi une vision globale et précise de l'efficacité de chaque algorithme face à notre problème spécifique.

Cette approche méthodique est cruciale pour évaluer de manière juste et rigoureuse les capacités de résolution des différents algorithmes dans un contexte contrôlé et répété.

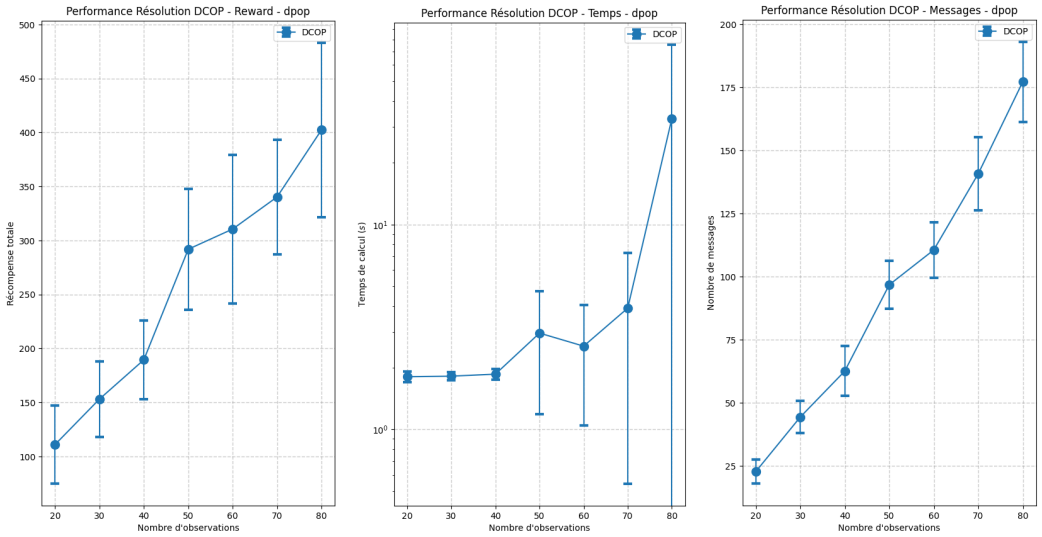

FIGURE 2 – Résultat Expérience - dpop

Ces résultats sont difficiles à comparer avec ceux obtenus par d'autres implémentations en raison du temps de calcul, qui peut rapidement devenir très long. Cette limitation représente un défi significatif dans l'évaluation comparative de notre approche, car elle affecte la faisabilité et l'efficacité de l'exécution des algorithmes, surtout pour des instances complexes ou de grande taille.

4 Partie 2 : Négociation entre agents

4.1 Protocoles de négociations

Dans cette section, nous abordons la mise en œuvre des protocoles de négociation visant à coordonner l'allocation des tâches entre les agents. Nous examinons en détail trois approches basées sur les enchères, notamment les enchères parallèles (**PSI**), les enchères séquentielles (**SSI**), et une variante des enchères séquentielles basées sur le regret. Chaque approche vise à optimiser l'attribution des tâches en utilisant des mécanismes d'enchères.

4.1.1 Enchères Parallèles (PSI)

Les enchères parallèles impliquent que chaque agent émet simultanément des offres pour l'ensemble des tâches disponibles. Chaque tâche est attribuée à l'agent qui propose l'offre la plus élevée. Ce protocole favorise la rapidité de la prise de décision, mais peut entraîner des problèmes d'efficacité si chaque agent ne dispose pas d'une information complète sur les coûts et les préférences des autres agents.

4.1.2 Enchères Séquentielles (SSI)

Contrairement aux enchères parallèles, les enchères séquentielles impliquent une émission itérative des offres, où chaque agent propose des enchères pour les tâches une par une. L'attribution se fait au fur et à mesure, ce qui permet aux agents d'ajuster leurs stratégies en fonction des enchères précédentes. Cela peut conduire à une allocation plus efficiente, mais le processus peut prendre plus de temps, mais moins de messages à envoyer.

4.1.3 Enchères Séquentielles Basées sur le Regret

Nous explorons une variante des enchères séquentielles basée sur le regret, une approche non étudiée dans les travaux cités. Cette méthode tient compte du regret de chaque agent par rapport aux enchères précédentes, visant à minimiser le regret total dans le choix final des tâches. Cela offre une flexibilité supplémentaire dans la gestion des enchères.

4.2 Comparaison des protocoles

Pour évaluer l'efficacité de notre approche et la situer par rapport aux travaux existants, nous avons conduit **deux expériences calquées sur celles menées dans l'article de référence**. **La première expérience** a pour but de générer **des instances dans un cadre temporel restreint où les conflits entre les observations sont fréquents**. Dans ce contexte, nous analyserons **trois métriques principales** : *le total des récompenses obtenues* pour chaque requête complétée, *le temps de calcul* nécessaire pour résoudre les instances, ainsi que *le nombre de messages échangés* durant le processus de résolution.

Les algorithmes que nous mettrons en compétition dans cette expérience incluent PSI, SSI, une variante de SSI axée sur le regret, ainsi que greedy_solver, un solveur basé sur une approche gloutonne. L'objectif est de comparer ces différentes méthodes afin de déterminer laquelle offre le meilleur compromis entre efficacité et coût computationnel, tout en maximisant les récompenses obtenues malgré les contraintes de temps et les conflits d'observations.

Pour cette expérimentation, les instances sont générées sur une période de 24 unités de temps, intégrant un nombre maximal de 25 observations par satellite pour un total de 5 satellites. Les requêtes formulées peuvent couvrir de 1 à 4 unités de temps, avec un volume variant de 30 à 100 requêtes, augmentant par paliers de 10, et chaque requête incluant 10 observations. Nous intégrerons également 5 utilisateurs et de 3 à 5 zones exclusives, soulignant une différence notable par rapport à l'article de référence : dans notre modèle, une zone exclusive ne peut être attribué qu'à un seul satellite.

Les graphiques ci-dessous sont la moyenne des résultats sur 20 essais pour chaque nombre d'observations.

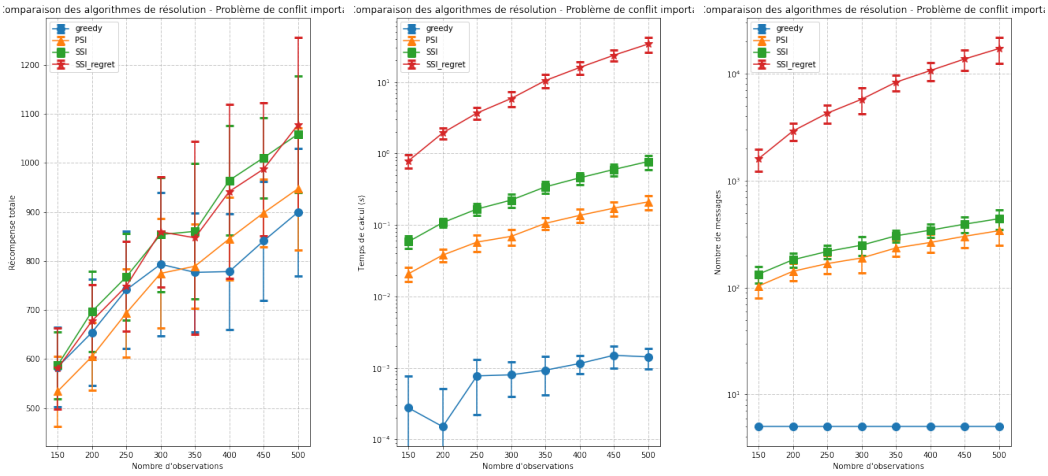

FIGURE 3 – Résultat Expérience - Problème de conflit important

Nos résultats indiquent que les récompenses obtenues à l'aide du solveur SSI sont légèrement meilleures, bien que ces résultats ne semblent pas pleinement représentatifs. Concernant le temps de résolution, nous avons détecté une anomalie avec le solveur SSI, qui prend plus de temps que celui observé dans l'article de référence. Cette situation est probablement due à une implémentation de notre solveur SSI qui pourrait bénéficier d'une optimisation supplémentaire. Par ailleurs, les résultats relatifs au nombre de messages échangés semblent être en accord avec ceux présentés dans l'article, ce qui suggère une certaine cohérence dans le comportement de communication de notre système par rapport aux observations.

Dans le cadre de notre **seconde expérience**, nous adoptons une approche visant à refléter **un scénario "réaliste"** en simulant un intervalle de temps étendu de 21 600 unités, représentant une période de deux ans. Pour cette simulation, le système comprend 8 satellites, chacun pouvant réaliser jusqu'à 800 observations. Les durées

des observations varient entre 40 et 60 unités de temps, offrant ainsi une gamme d'engagements temporels significatifs pour chaque satellite.

Le nombre d'utilisateurs impliqués dans cette expérience est maintenu à 5, similaire à la première expérience. Cependant, le volume de requêtes est augmenté, avec un total variant de 50 à 150 requêtes, et un incrément de 10 requêtes à chaque étape. Cette configuration permet d'explorer la capacité des algorithmes à gérer efficacement un flux substantiel de demandes sur une période importante, tout en naviguant à travers les contraintes temporelles et les limitations d'observation.

Les graphiques ci-dessous sont la moyenne des résultats sur 20 essais pour chaque nombre d'observations.

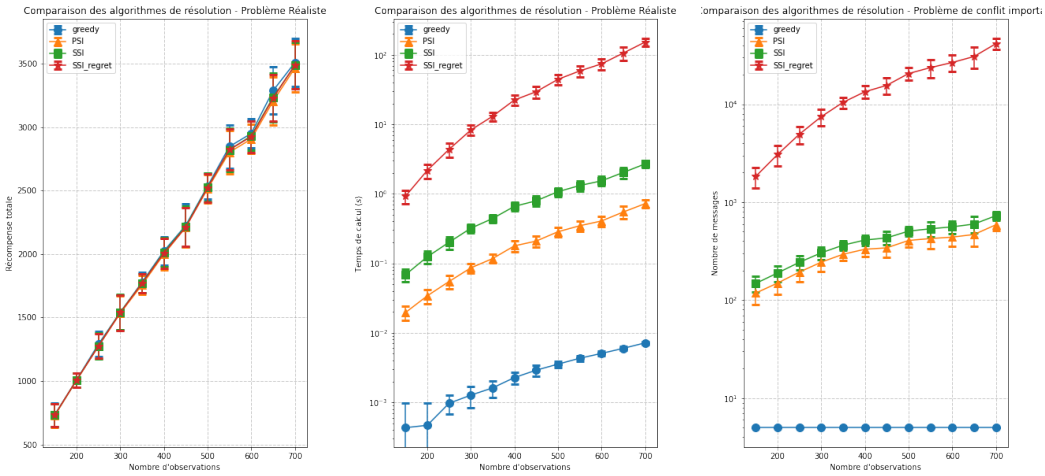

FIGURE 4 – Résultat Expérience - Situation "Réaliste"

Dans cette série d'expériences, nous observons que la performance en termes de récompense est similaire pour chaque algorithme, ce qui corrobore les attentes que nous avions établies d'après l'article de référence. En ce qui concerne le nombre de messages échangés et le temps de calcul, nos observations demeurent cohérentes avec celles de la première expérience. Nous notons un défaut d'optimisation dans l'algorithme SSI, qui ne parvient pas à reproduire les résultats observés dans l'article. Cela indique une marge d'amélioration dans notre mise en œuvre de SSI.

Quant à la charge de messages, bien qu'elle semble proche de celle rapportée dans l'article, la différence d'échelle de nos expériences pourrait fausser notre interprétation. Cela suggère que bien que nos résultats se rapprochent des tendances générales observées dans la littérature, des ajustements sont nécessaires pour une comparaison plus précise et une performance optimale de nos algorithmes.

Pour conclure cette partie expérimentale, plusieurs points importants peuvent être soulignés :

- **Efficacité des Solutions :** Les solutions testées semblent trouver de bon résultats pour des problèmes de planification soumis à des contraintes importantes. Cela démontre l'efficacité des algorithmes utilisés dans la gestion de situations complexes.
- **Comparaison des Solveurs :** Nos résultats, ainsi que ceux rapportés dans l'article, indiquent que certains solveurs se révèlent plus efficaces que d'autres. Notamment, la modélisation en DCOP ne présente pas d'avantages significatifs par rapport à d'autres méthodes, en particulier les protocoles liés aux enchères.
- **Versatilité de SSI :** Parmi les différentes solutions étudiées, le protocole SSI se distingue par sa versatilité. Ses performances sont relativement bonnes sur tous les critères et métriques analysés, y compris la charge de messages, une métrique spécifiquement étudiée dans l'article de référence.

En résumé, bien que les différentes approches de résolution présentent chacune leurs forces, le protocole SSI apparaît comme une option polyvalente et efficace, offrant un bon équilibre entre récompenses, temps de calcul et nombre de messages échangés. Cela suggère son potentiel comme solution de choix pour des problèmes de planification complexes dans des systèmes multi-agents.

5 Conclusion

Pour conclure, ce projet a rencontré de nombreux défis d'implémentation et souffre vraisemblablement d'un manque d'optimisation sur plusieurs aspects, ce qui aurait pu contribuer à l'obtention de résultats plus satisfaisants de manière globale. Cependant, nos résultats offrent un aperçu encourageant des possibilités offertes par la modélisation de problèmes complexes à travers des systèmes multi-agents. Ils soulignent l'importance et le potentiel de cette approche pour aborder et résoudre des scénarios d'opération et de coordination exigeants, tout en mettant en évidence les domaines nécessitant une attention supplémentaire pour améliorer l'efficacité et la performance des solutions proposées.