# Sequential Assignment Problems and Their Application to Task Allocation in Satellite Constellations

Joshua Holder

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Aeronautics and Astronautics

University of Washington

2024

Reading Committee:
Mehran Mesbahi, Chair
Behçet Açikmeşe

Program Authorized to Offer Degree:
Department of Aeronautics and Astronautics

University of Washington

**Abstract**

Sequential Assignment Problems
and Their Application to Task Allocation
in Satellite Constellations

Joshua Holder

Chair of the Supervisory Committee:
Mehran Mesbahi
Department of Aeronautics and Astronautics

As satellite constellations grow in size, there is an increasing need for methods of autonomous, scalable, and real-time dynamic task assignment. The satellite constellation setting is unique among task allocation problems in that orbital mechanics dictates that assignments between agents and tasks must change frequently. This gives rise to the Sequential Assignment Problem (SAP), in which agents must be assigned to tasks at several consecutive time steps, while accounting for assignment values which are constantly evolving and the potential penalties associated with changing assignments. This thesis formalizes the Sequential Assignment Problem, and proposes two classes of algorithm to generate solutions to it. The first, Handover Aware Assignment with Lookahead, is an optimization-based algorithm which can generate assignments which are computable in polynomial time, fully distributed, and provably optimal within a bound. The second algorithm is Reinforcement Learning Enabled Distributed Assignment, which utilizes reinforcement learning to find performant solutions to a larger class of Sequential Assignment Problems. Experimental results show that both approaches vastly outperform the state-of-the-art across a variety of challenging, realistic domains with hundreds or even thousands of agents and tasks, with both approaches having clear strengths and weaknesses.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# GLOSSARY

AGENT: an entity which has the ability to contribute to the completion of a task (typically satellites in this work).

CONSTELLATION: a group of satellites working together to achieve a common goal or goals.

HANDOVER: an instance of an agent switching its assigned task in consecutive time steps.

LEO: low earth orbit, typically defined as orbits with an altitude of 2,000 km or less.

TASK: a sub-goal necessary for achieving the overall objectives of the system that can be achieved independently of other such sub-goals [25] (typically providing internet to or observing regions of Earth in this work).

# ACKNOWLEDGMENTS

I'd like to thank all the many collaborators I leaned on in my work on this thesis—Dr. Mehran Mesbahi, for his many insightful suggestions that were invaluable in nudging this research in a fruitful direction.

Spencer Kraisler, for pushing me to strive for absolute clarity in my work.

The members of RAIN lab (Aditya Deole, Jackson Zhou, Shiva Shakeri, Shahriar Talebi, Berit Syltebo, Seth Reed) who sat through many muddled presentations when this work was still in its infancy and provided helpful feedback.

Natasha Jaques and Abhishek Gupta, for taking a chance on an unknown aerospace engineer and teaching me so much about reinforcement learning and conducting effective research in computer science.

To all the fantastic professors from whom I learned so much during my time at UW.

And finally to my family, who I love and respect so much and who I know is always in my corner even as my dreams take me far afield, geographically but also in the space of ideas.

# DEDICATION

To my parents, for showing me the value of doing hard work you love, and providing me the resources to do it.

# Chapter 1
# INTRODUCTION

In this chapter, we provide the motivation and historical background which underlies satellite task assignment problems, and describe how the work of this thesis fits into the current state of the art.

## *1.1 Motivation*

Much has been made of the dramatic rise in the number of objects launched to orbit in recent years, precipitated by the reduced cost and increased cadence offered by companies like SpaceX. Various pundits [11, 65] have cited this exponential trend to motivate a prediction for a corresponding explosion in unique space companies and use cases.

While this prediction may eventually be realized, looking only at the exponential rise in the number of objects (the blue line in Figure 1.1) obscures a deeper trend. As shown by the orange line in Figure 1.1, a large majority of this recent increase in the number of objects being placed in orbit can be attributed to just a few large projects—namely, SpaceX's Starlink and Planet's Dove, SkySat, and RapidEye constellations.

This suggests that the more meaningful trend illuminated by Figure 1.1 is not an exponential increase in the raw number of objects in orbit, but, driven by decreased launch costs and a desire for increased performance, rather an exponential increase in the *number of satellites used to complete any given task*. In other words, aerospace missions are moving away from large, monolithic satellite platforms and towards large-scale distributed systems.

This trend is likely to only accelerate—SpaceX already has approval for a constellation of up to 12,000 satellites and plans for 30,000 more, Amazon's Kuiper project will launch at least 3,236, and if all planned constellations come to fruition, the total could near 100,000

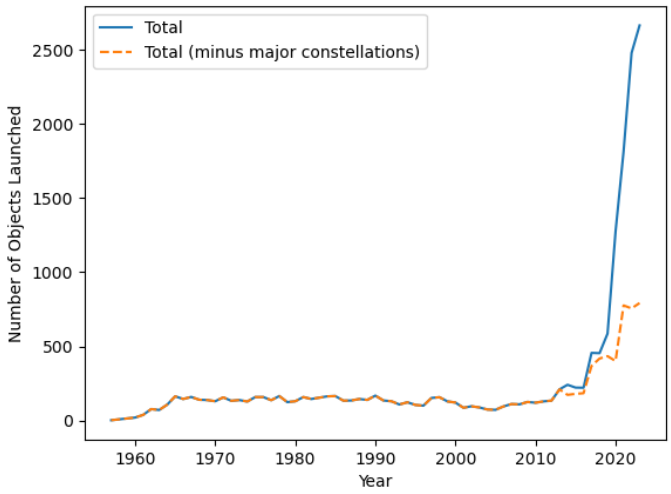



Figure 1.1: Annual number of objects launched to space (United Nations Office for Outer Space Affairs (2024) – with major processing by Our World in Data).

[14, 66]. Given a single satellite, a mature body of work describes how to place, maintain, and operate it in order to complete tasks. However, algorithms which allow for the efficient use of large groups of satellites are still in their infancy—how should such groups most effectively collaborate, share information, and respond to dynamically evolving situations?

Specifically, there is a need for improved methods of *satellite task allocation* for large constellations. In this setting, a *task* is defined as a sub-goal necessary for achieving the overall objectives of the system that can be achieved independently of other such sub-goals [25] (i.e. providing internet to a group of users or taking a scientific measurement of a geographic region). An *allocation* algorithm is then a method of assigning these sub-goals to satellites in an efficient way so that the overall goal is accomplished. Such algorithms will be critical to the performance and economic viability of these next-generation space systems—gone are the days when satellite operators could manually assign tasks to individual satellites [21].

The satellite constellation environment poses several unique challenges that distinguish

it from the more general body of work on task allocation in large scale systems. First, communication with satellites is highly constrained—centralized task allocation in this setting requires a costly ground network [13] and can suffer from significant latency.

Second, and more profoundly, satellite constellation task allocation algorithms have a uniquely sequential nature. In most classical task allocation problems (i.e. processor scheduling, mobile robot routing, etc.) once a task is assigned to an agent, it can be completed by the same agent until the task is completed. In the case of low-earth orbit (LEO) satellites, however, orbital mechanics dictates that satellites cannot accomplish the same task indefinitely, and thus that the transition between assignments must be considered. For example, switching tasks often means changing the attitude of the spacecraft or reestablishing a network connection—this cost should be integrated into the assignment decisions made by a satellite task allocation algorithm. An instance of a satellite changing its assigned task is defined as *handover* in this work.

Although many of the algorithms and problem formulations defined in this work are highly general, we take special interest in contributing to the growing field of satellite constellation task allocation. As such, our proposed solutions were developed with the unique design criteria necessitated by the satellite constellation setting in mind—namely, the desire for fully-distributed solutions that account for the constantly changing positions of satellites and the desire to minimize handover. First, we outline existing work on task allocation in satellite systems.

## *1.2 Prior Work on Satellite Task Allocation*

Driven by evolving technology and new application areas, the field of satellite task allocation has experienced constant development since the launch of Sputnik in 1957.

### *1.2.1 Single-Satellite Task Allocation Problems*

Initially, satellites were rare, expensive, and functioned largely as independent agents. This motivated work on the single-satellite Agile Earth Observing Satellite (AEOS) scheduling

problem starting in 1997 with [24], in which a satellite attempts to optimally execute a series of tasks and data downlinks from a list of possibilities. Although most formulations of this problem are NP-hard, today algorithms which can generate solutions to the single-satellite AEOS scheduling problem are relatively mature and span the spectrum from exact mixed integer linear programming methods (MILPs) [56] to machine learning approaches [28].

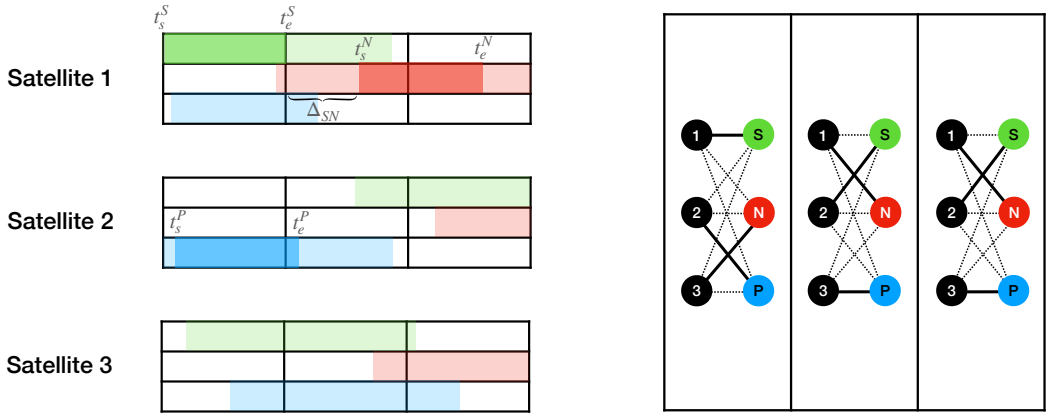
### *1.2.2 Multi-Satellite Task Allocation Problems*

Starting in the 1980s with the "Brilliant Pebbles" missile defense program, interest grew in explicitly using collaborative groups of satellites to accomplish tasks. The addition of multiple satellites into the task allocation problem greatly increases its computational complexity and introduces issues of communication.

Additionally, it introduces a subtle distinction in the definition of tasks. In the majority of literature on task allocation, tasks take the form of an action to be completed once (i.e. take a single picture of Paris). Accordingly, the problem takes on the flavor of a "job shop scheduling problem", in which tasks with defined start and end times are scheduled across multiple machines to be completed once each [17]. This is necessary in small constellations, where satellites are too sparse to provide coverage of multiple regions at all times. Hereafter, we refer to this task allocation formulation as the *task scheduling problem* or the Agile Earth Observing Satellite Scheduling Problem (AEOSSP), visualized in Figure 1.2a.

With larger constellations, as are our focus in this work, there are enough satellites to achieve constant coverage of entire areas of the Earth. Internet constellations such as Starlink and Kuiper are examples of constellations using this tasking paradigm—rather than needing to provide internet to a location at a single time, the constellations must consistently assign a satellite to each region. This is visualized in Figure 1.2b and hereafter referred to as the *task assignment problem* (or simply the task allocation problem in succeeding sections).

Although the task assignment problem could be expressed as an AEOSSP with a task scheduled for every region at every time step, an assignment approach allows us to exploit efficient and optimal solutions to assignment problems, as we will see later.




(a) The *task scheduling problem*, where observation tasks are completed once each and assigned continuous start and end times $t_s, t_e \in \mathbb{R}$ while respecting transition times $\Delta$.

(b) The *task assignment paradigm*, where at each discrete time step an assignment problem is solved such that all observation tasks are allocated to a satellite in each time step. This is the focus of our work in this thesis.

Figure 1.2: Diagram of the *scheduling* vs. *assignment* task allocation paradigms.

*Multi-satellite task scheduling problems (AEOSSP)*

Just as in the single-satellite setting, the AEOSSP is NP-hard in all but the most simple of scenarios [17], and with the addition of multiple satellites, the problem becomes even more intractable. Research in this area is still ongoing [63], but approaches include:

- **MILP methods:** express the full AEOSSP as a single optimization problem. These methods are highly expressive and performant, but scale poorly as the number of agents and/or tasks grows [9, 12, 40].

- **Learning-based methods:** use pretrained machine learning models in the task scheduling pipeline. While computationally expensive to develop, these methods are typically computationally cheap during operation. Typically, they achieve good em-

pirical performance but have naive communication frameworks and unknown ability to generalize to new scenarios [15, 16, 29].

- **Auction-based methods:** run "auctions" among satellites to allocate tasks. Solutions in this framework can be generated in a fully distributed way, but are often characterized by restrictive problem settings, highly suboptimal solutions, and/or prohibitive communication requirements [33, 46, 48].

*Multi-satellite task assignment problem*

Literature on the satellite task assignment problem (which we aim to address in this work) is relatively sparse, as only recently have such large constellations been imagined. Work that does exist falls into similar categories as work in the scheduling paradigm.

Several works employ classical methods to this problem setting. In [44], the authors develop greedy heuristic approaches to the problems of assigning users to beams, and assigning beams to satellites and frequency ranges. However, the purely heuristic approach has no guarantees of optimality. In [26], the authors formulate a similar problem, but apply MILP as a solution method. This limits the scaling of the strategy significantly, with the experiments in the paper being limited to 15 satellites. Finally, [43] uses particle swarm optimization to assign beams to satellites.

All three classical approaches mentioned are fully centralized, and none take into account the sequential nature of the problem, instead computing their solutions from scratch at each time step. This will result in an increased number of handovers, a fact which the authors of [26] explicitly mention.

Several learning-based approaches have also been applied to the task assignment problem. In [27, 37], the authors apply multi-agent reinforcement learning techniques to the task assignment problem, showing improved performance when compared to various heuristic and metaheuristic approaches in the literature, even in complex problem domains.

However, both works focus on a single multi-beam satellite rather than a constellation of

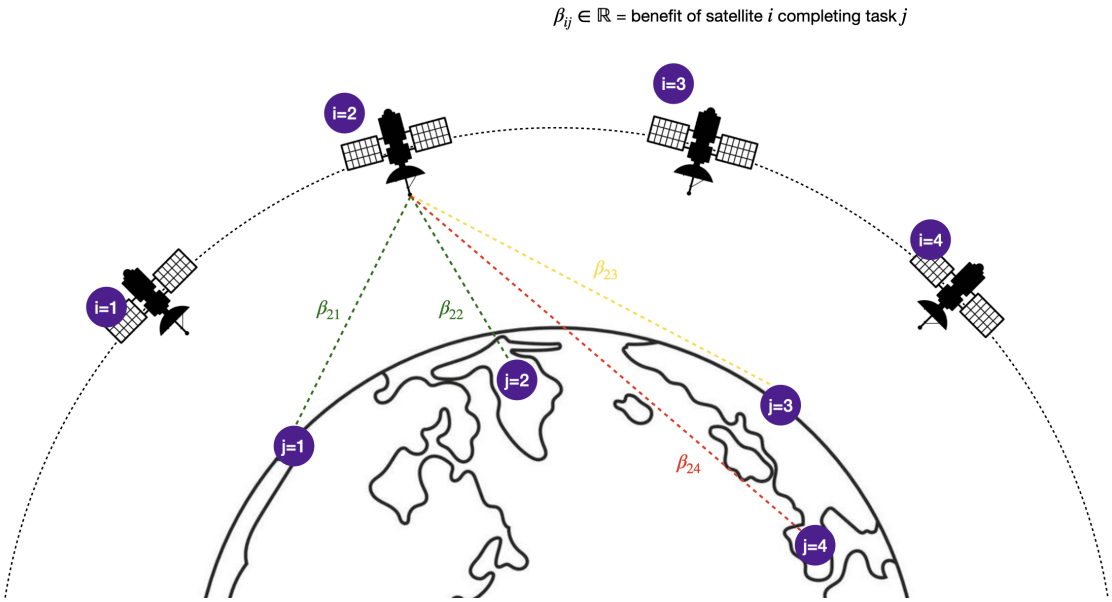



Figure 1.3: Diagram of a typical satellite assignment problem. The goal of a typical assignment problem is to assign each of the $n = 4$ agents (satellites) to one of the $m = 4$ tasks (areas on Earth).

satellites, and [37] relies on learned cooperation between each beam to avoid being assigned to the same tasks, rather than any deterministic mechanism. Because of this, it is unclear how these approaches might scale to constellations of many multi-beam satellites, where at each time step a wide variety of beams might be competing for the same task, and where learned strategies of assignment might break down.

In the following section, we outline the general problem formulation which attempts to generate solutions to the task assignment problem while accounting for handover constraints and other state-dependent aspects.

## *1.3 Sequential Assignment Problem Formulation*

We have stated that the task assignment problem involves solving an assignment problem at each time step. Assignment problems in general can be described as the task of assigning

$n$ agents to $m$ tasks so as to maximize utility subject to some constraints—see Figure 1.3 for a typical example. The simplest version of an assignment problem where $n \leq m$ can be formulated as follows:

$$\max_{x \in X} \sum_{i=1}^{n} \sum_{j=1}^{m} \beta_{ij} x_{ij} \tag{1.1}$$

where:

- $\beta \in \mathbb{R}^{n \times m}$ is the *benefit matrix*, where $\beta_{ij}$ corresponds to the benefit of agent $i$ completing task $j$.
- $x \in X \subset \{0,1\}^{n \times m}$ is the *assignment matrix*, where $x_{ij} = 1$ if agent $i$ is assigned to task $j$, and $x_{ij} = 0$ otherwise.
- $X := \{x \in \{0,1\}^{n \times m} : \sum_{j=1}^{m} x_{ij} = 1 \, \forall i, \, \sum_{i=1}^{n} x_{ij} \leq 1 \, \forall j\}$ is the set of valid assignments. This corresponds to the set of assignment matrices such that each agent completes 1 task, and each task is completed by at most 1 agent.

Consider instead the problem of generating $T$ assignments in sequence (i.e. over the course of an orbit). This problem could be formulated using sequences of benefit and assignment matrices, $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_T)$ and $\mathbf{x} = (x_1, \cdots, x_T)$ respectively.[1] Then, the sequential analog of (1.1) is written as follows:

$$\max_{\mathbf{x}} \sum_{k=1}^{T} \sum_{i=1}^{n} \sum_{j=1}^{m} [\beta_k]_{ij} [x_k]_{ij} \tag{1.2a}$$

$$\text{s.t. } x_k \in X \text{ for all } k \tag{1.2b}$$

As we will show Section 2, existing methods can easily obtain optimal solutions to Equation 1.2 [36]. However, (1.2a) is rarely expressive enough for realistic problems. Often, the value of an assignment $x_k$ depends on some concept of the "state" of the environment at that time, denoted $s_k \in S$, where $S$ is the set of all possible states. We can then express the benefit matrix as a function of state, $\hat{\beta}(s_k) \in \mathbb{R}^{n \times m}$.

[1] As such, boldface letters are used to denote sequences over time.

A concrete example of this state dependence could emerge when the value of an assignment depends on the previous assignment $x_{k-1}$—in satellite applications a change in satellite assignment can require a change in orientation and thus incur fuel costs. In such a scenario, $x_{k-1}$ would be contained in the state $s_k$.

This state also evolves over time according to the chosen assignments. We define a state transition function $\mathcal{T} : S \times X \to S$ such that we can calculate the evolution of the state over time via $s_{k+1} = \mathcal{T}(s_k, x_k)$.

Using these definitions, the Sequential Assignment Problem (SAP) can be written as follows:

$$\max_{\mathbf{x}=(x_1,\cdots,x_T)} \sum_{k=1}^{T}\sum_{i=1}^{n}\sum_{j=1}^{m}[\hat{\beta}(s_k)]_{ij}[x_k]_{ij} \tag{1.3a}$$

$$\text{s.t. } x_k \in X \text{ for all } k \tag{1.3b}$$

$$s_k = \mathcal{T}(s_{k-1}, x_{k-1}) \text{ for all } k \tag{1.3c}$$

Here, the assignment constraint (1.3b) implies that each agent is assigned to exactly one task in every time step and that each task is assigned to at most one agent per time step, as defined above.[2] Equation 1.3c ensures that the state evolves according to the transition dynamics $\mathcal{T}$.

### *1.4 Prior Work on the Sequential Assignment Problem*

To our knowledge, the SAP has never been specifically addressed in prior work. However, it shares several similarities to other works found in the more general operations research literature. There is a large body of work on the Quadratic Assignment Problem (QAP), which has the following structure:

$$\min_{\hat{x}\in X} \sum_{i=1}^{\hat{n}}\sum_{j=1}^{\hat{m}}\sum_{k=1}^{\hat{n}}\sum_{l=1}^{\hat{m}} a_{ik}b_{jl}\hat{x}_{ij}\hat{x}_{kl} + \sum_{i=1}^{\hat{n}}\sum_{j=1}^{\hat{m}} c_{ij}\hat{x}_{ij} \tag{1.4}$$

[2]Scenarios in which agents which can complete multiple tasks or where tasks can be completed by multiple agents can be addressed in this framework, but one needs to cleverly redefine agents and tasks such that these constraints remain satisfied—see Sections 4.3 and 4.4.

The general form of this problem is known to be NP-hard. Accordingly, optimal solutions have only been generated to problems of size $n \sim 30$ [7], and many heuristics have been proposed for generating suboptimal solutions [42].

The SAP can be reformulated into the form of Equation 1.4 by setting $\hat{n} = nT, \hat{m} = mT$, reshaping our assignment sequence $\mathbf{x} = (x_1, \cdots, x_T)$ into a single block-diagonal matrix $\hat{x} \in \mathbb{R}^{nT \times mT}$, and forming $a$, $b$, and $c$ accordingly.[3] However, to our knowledge the specific form of QAP arising from the SAP is also NP-hard.

When $\hat{\beta}(s_k)$ rewards assignments for staying consistent, the SAP also has a close relationship with robust assignment problems, which are formulated as the robust version of Equation 1.1,

$$\max_{x \in X} \min_{\beta \in \boldsymbol{\beta}} \sum_{i=1}^{n} \sum_{j=1}^{m} \beta_{ij} x_{ij} \tag{1.5}$$

where $\boldsymbol{\beta}$ is the set of benefit matrices the solution should be robust over. This problem, as with many of the problems addressed in this thesis, is NP-hard, but heuristic and dynamic programming based solutions have been proposed, and the problem admits pseudopolynomial solutions under specific conditions [35].

The formulation in (1.5) is overly restrictive, however, as it requires a single assignment $x$ for all $\beta \in \boldsymbol{\beta}$. In the sequential assignment problem, we can select a new assignment $x_k$ at each time step. This scenario is closer to the second stage recoverable robust assignment problem:

$$\max_{x \in X} \sum_{i=1}^{n} \sum_{j=1}^{m} \beta_{ij} x_{ij}$$

$$\text{s.t. } x \text{ shares at least k assignments with } x^{-}$$

This formulation allows for some assignments to change between time steps, but is again overly restrictive—in the case of the SAP, if benefits change significantly between time steps, it is desired that the assignments also change significantly. It also has no known polynomial

[3] Assuming that the state $s_k$ only depends on $x_{k-1}$ and components unaffected by transitions.

time solution, and can only make use of information from the current time step [22]. These factors make this formulation undesirable as a solution to the SAP.

Finally, the SAP can be thought of as a form of problem common in the optimization literature where agents aim to sequentially complete a group of tasks in any order, but where choosing to do one task can affect the reward of another task. These problems are also NP-hard and admit only heuristic solutions [10], and we will see in Chapter 4 that in practice, algorithms for this domain are far less suited to the SAP.

While reinforcement learning (RL) has shown great promise in solving a variety of highly complex, state-dependent sequential decision making problems, there are several barriers to applying cutting-edge algorithms from this field to the SAP, most of which relate to ensuring that chosen assignments lie within the set $X$. These difficulties will be discussed in more detail in Chapter 5.

Perhaps the most similar work to the RL approach described in Chapter 5 comes from the transportation network domain [61], in which RL is not directly applied but rather used to estimate values which are used as inputs to an optimal method bipartite assignment mechanism ($\alpha$ as defined in Section 2.1). This method was applied to real-time, global-scale task assignment at DiDi, and later applied to a similar problem at Lyft [4]. Despite similarities, our method uses neural networks rather than linear function approximation, bootstraps solutions using an optimal assignment mechanism, and uses a novel formulation to expose deeper theoretical insights into the workings of this method. These properties suggest that our algorithm can be more performant and efficient in many large-scale settings, including the satellite constellation setting.

To our knowledge, the only work to address a similar problem in the specific context of satellite constellations is [47]. In this work, the authors apply reinforcement learning to maximize connection quality and *connection duration* between satellites and user clusters (tasks). This can be thought of as an instance of the SAP. However, the algorithm proposed in this work does not come with performance guarantees, cannot easily generalize across multiple realistic scenarios, and is entirely centralized and greedy, assigning tasks to satellites

one at a time in sequence.

### *1.5 Contribution*

The Sequential Assignment Problem (1.3) can express a wide range of complex problems with practical relevance. Despite the lack of a formal proof, a review of similar problems in the literature strongly suggests that the SAP in its most general form is NP hard. Accordingly, this thesis will focus on principled heuristics and machine learning approaches which make use of the problem's unique structure to generate good solutions.

Our main contribution is to present two algorithms for generating solutions to the SAP. Our first algorithm, Handover-Aware Assignment with Lookahead (HAAL), is a deterministic, optimization based method. In settings where changes in task assignment should be minimized, we show that HAAL vastly outperforms all other state-of-the-art algorithms in generating solutions to (1.3). We show that HAAL has several other attractive properties: it is fully distributed, can be adjusted to trade off computational cost for performance, and is provably optimal within a bound under certain conditions.

We also introduce RL-Enabled Distributed Assignment (REDA), a RL-based method of generating solutions to the SAP. This method uses a theoretically justified approach to avoid pitfalls faced by other state-of-the-art RL algorithms, and can be successfully applied to a much wider class of SAPs. It is also fully distributed and can be tuned for a desirable computation-accuracy tradeoff. Although not provably optimal or deterministic, we show that REDA exhibits comparable performance to HAAL in settings where HAAL is applicable, and vastly improved performance when compared to HAAL and other RL algorithms in more general and complex domains.

Overall, this thesis presents several novel strategies that can generate solutions to critical real-life problems in domains such as satellite constellations, power grids, and transportation networks with a previously unprecedented level of quality.

### *1.6 Thesis Organization*

In Chapter 2, we present the necessary mathematical background on assignment problems and reinforcement learning. In Chapter 3, we present HAAL and its variants and prove several properties about the algorithm. In Chapter 4, we apply HAAL to a variety of realistic problem settings and analyze the results. In Chapter 5, we describe challenges faced in applying RL to the SAP, and introduce REDA and its theoretical properties. In Chapter 6, we apply REDA to several more problem settings and compare performance to HAAL and other algorithms. Finally, in Chapter 7, we provide some concluding remarks, discuss other applications of REDA and HAAL, and provide ideas on future research directions on the Sequential Assignment Problem.

# Chapter 2

# BACKGROUND AND MATHEMATICAL PRELIMINARIES

This chapter introduces some terminology and notation related to assignment problems and reinforcement learning.

## *2.1 Standard Assignment Problems*

Recall the simplest possible assignment problem,

$$\max_{x \in \{0,1\}^{n \times n}} \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{ij} x_{ij}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{for all } i$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{for all } j$$

While the integer constraint on $x$ may initially seem problematic, in fact we can exploit the unimodularity of the constraint matrix to ensure that optimal solutions to the unconstrained problem will naturally include entries which are either 0 or 1. This allowed a solution to this problem, dubbed the "Hungarian Method", to be first developed by Harold Kuhn in 1955 [36], motivated by personnel scheduling. While the algorithm initially had $O(n^4)$ complexity, later improvements in the algorithm modified it to run first in $O(n^3)$ time [62] and then fully parallelized [5].

These algorithms have several interpretations, with some implementations simply performing abstract matrix operations, while others drawing connections to finding paths on bipartite graphs or deep connections with duality theory. The result is that as of 2024, even extremely large assignment problems are almost trivial to solve (i.e. with a single Python

command). Throughout this work, we exploit this fact to great effect and use standard assignment problems as a building block in our other algorithms.

For convenience, we denote the solution to Equation 1.1 provided by these algorithms as the *assignment function* $\alpha : \mathbb{R}^{n \times m} \to \mathcal{X}$, which is a mapping from any benefit matrix to the assignment corresponding to the optimal solution of (1.1):

$$\alpha(\beta) := \underset{x \in X}{\operatorname{argmax}} \sum_{i=1}^{n} \sum_{j=1}^{m} \beta_{ij} \, x_{ij}. \tag{2.1}$$

Note that $\alpha$ can easily be used to solve Equation 1.2, where $\mathbf{x}^* = (\alpha(\beta_1), \cdots, \alpha(\beta_T))$.

Although the Hungarian Method describes how to solve (1.1) efficiently, it still requires a centralized node to compute optimal solutions. In a variety of applications such as satellite constellations, this centralization comes with increased latency and decreased robustness.

To address this issue, [68] presents an algorithm to compute $\alpha(\beta)$ in a completely decentralized setting, where agents can only communicate with neighboring agents in a graph structure. Task assignments are determined by running a decentralized auction, in which agents track prices and high bidders for each task, and are assigned to the task for which they are willing to pay the most after prices have reached equilibrium. This algorithm is guaranteed to converge to within $n\epsilon$ of the optimal solution, where $\epsilon$ is the bid interval.

## *2.2 Reinforcement Learning*

Reinforcement learning (RL) is a framework for solving sequential decision making problems by interacting with an environment and learning from experience [58]. Recently, the field of deep reinforcement learning has emerged, which combines principles of machine learning with those from classical RL to allow us to generate effective policies in highly complex domains like Go [54], robotics [6], and even protein folding [32].

We model a sequential decision making problem as a finite time Markov Decision Process (MDP) $\mathcal{M} = (S, A, S_1, \mathcal{T}, \mathcal{R}, \gamma)$, where:

- $S$ is the state space.

- $A$ is the action space.
- $S_1$ is the distribution of starting states (i.e. $s_1 \sim S_1$).
- $\mathcal{T} : S \times A \to S$ is the environment transition dynamics, i.e. $s_{t+1} \sim \mathcal{T}(s_t, x_t)$.
- $\mathcal{R} : S \times A \to \mathbb{R}$ is the reward function.
- $\gamma \in \mathbb{R}$ is the discount rate.

We define a policy $\pi : S \to A$ as a function which determines what action to take in a given state. Our goal is to generate an optimal policy $\pi^*$ such that reward is maximized:

$$\pi^* := \underset{\pi}{\operatorname{argmax}}\ \mathbb{E}^{\pi}\left[\ \sum_{k=1}^{T} \gamma^{k-1}\mathcal{R}(s_k, a_k)\ \right] \tag{2.2}$$

where the expectation $\mathbb{E}^{\pi}$ assumes that the initial state is sampled from $s_1 \sim S_1$, the agent follows the policy to select actions (i.e. $a_k \sim \pi(s_k)$), and that the state of the environment evolves according to $s_{k+1} \sim \mathcal{T}(s_k, a_k)$.

We can define $Q^{\pi} : S \times A \to \mathbb{R}$, the *Q-function of policy* $\pi$:

$$Q^{\pi}(s_k, a) := \mathbb{E}^{\pi}[\mathcal{R}(s_k, a) + \sum_{t=k+1}^{T} \gamma^{t-k}\mathcal{R}(s_t, a_t)]. \tag{2.3}$$

This can be interpreted as the value of selecting an action $a$ in state $s_k$, given that all future actions are selected according to $\pi$. The optimal policy $\pi^*$ can also be stated by using the $Q$-function as follows:

$$\pi^* := \{\pi' : Q^{\pi'}(s, a) \geq Q^{\pi}(s, a)\ \ \forall \pi \in \Pi,\ s \in S,\ a \in A\} \tag{2.4}$$

Thus, two approaches to finding $\pi^*$ present themselves. First, one can attempt to find $\pi^*$ directly, by using *policy gradient methods* to optimize a parametrized policy based on some reward function, as in [59] and more recently [53].

Alternatively, one can first find $Q^{\pi^*}$ and use (2.4) to recover $\pi^*$ as $\pi^*(s) := \underset{a \in A}{\text{argmax}}\, Q^{\pi^*}(s, a)$. To find $Q^{\pi^*}$, we define the *Bellman Optimality Operator* $F : Q \to Q$ as follows:

$$(FQ^\pi)(s_k, a_k) := r(s_k, a_k) + \gamma \max_{a'} Q^\pi(s_{k+1}, a')$$

Note that by definition, $Q^{\pi^*}$ is a fixed point of $F$, and per Banach's Fixed Point Theorem [34], we can show that repeatedly applying $F$ to a $Q$-function will yield the $Q^*$ in the limit. Formally:

$$Q^\pi_{n+1} = FQ^\pi_n \implies \lim_{n \to \infty} Q^\pi_n = Q^{\pi^*} \text{ for all } Q^\pi \in \mathcal{Q}.$$

This process of repeatedly applying the Bellman Optimality Operator to an arbitrary $Q^\pi$ until reaching $Q^{\pi^*}$, and thus $\pi^*$, is the basis of *Q-learning methods* in RL [39, 64].

### *2.3 Multi-Agent Reinforcement Learning*

In some cases, there are $n$ independent agents acting in the environment, all of which we want to learn to control. This setting is called Multi-Agent Reinforcement Learning (MARL), and is often modeled as a partially observable Markov decision process (POMDP) $\mathcal{M} = (S, A, S_1, \mathcal{T}, \mathcal{R}, \gamma, O, \mathcal{O})$, where $S$, $S_1$, $\mathcal{T}$, and $\gamma$ are defined as before [3].

The action space $A$ is now a *joint action space* $A = A^1 \times ... \times A^n$, where the *joint action* $a = (a^1, \cdots, a^n)$ consists of the actions of each agent in the environment.

In a partially observed setting, we do not assume that each agent in the environment has access to the full state $s$ of the environment. Instead, we say that at every time step $k$, agent $i$ receives an *observation* $o^i_k \sim \mathcal{O}^i(s_k),\ o^i \in O^i$, where $O^i$ is the *observation space* and $\mathcal{O}^i$ is the *observation function* for agent $i$. Each agent thus conditions its policy on observations rather than states (i.e. $\pi^i(o^i) = a^i$.)

Finally, rather than a single global reward function, each agent has its own reward function $\mathcal{R}^i : S \times A \to \mathbb{R}$ which it is independently trying to maximize. Note that in the general case $\mathcal{R}^i$ depends on the joint action, not just $a^i$.

Using this definition of reward, we can define the $Q$-functions for each individual agent

as follows:

$$Q_i^{\pi^i}(s_k, a_k) := \mathbb{E}^{\pi^i}\left[\mathcal{R}_i(s_k, a_k) + \sum_{t=k+1}^{T} \gamma^{t-k}\mathcal{R}_i(s_t, a_t)\right]$$

This is in general a far more challenging optimization problem than standard RL for several reasons:

1. **Non-stationarity.** From the perspective of a single agent, other agents can be modelled as part of the environment. However, this means that over the course of training as other agents update their policies, the environment is changing, potentially leading to training instability.

2. **Multi-agent credit assignment.** In an environment where multiple agents are acting simultaneously, it is often difficult for an agent to disentangle whether their own actions caused a positive reward, or whether it was the actions of another agent.

Despite these challenges, MARL offers an improved ability to scale to multi-agent environments where centralized control is undesirable (i.e. for latency reasons) or infeasible (i.e. where the size of the joint action space scales exponentially with the number of agents [3]).

# Chapter 3

# A CLASSICAL OPTIMIZATION APPROACH TO THE SEQUENTIAL ASSIGNMENT PROBLEM

In this chapter we show that the SAP (1.3) admits a deterministic, fully distributed algorithm with optimality guarantees under certain conditions.[1]

## *3.1 Preliminaries*

Before outlining the algorithm, we define some relevant terminology and notation.

For practical and theoretical reasons, it is helpful to split the state-dependent benefit matrix $\hat{\beta}(s_k) \in \mathbb{R}^{n\times m}$ into two parts:

$$\hat{\beta}(s_k) := \beta_k(s_k) + \eta(s_k).$$

where $\beta_k(s) \in \mathbb{R}^{n\times m}$ is the component of the value of an assignment that satisfies the following property:

$$\beta_k(s) = \beta_k(s') \text{ for all } s, s' \in S \tag{3.1}$$

Thus, we drop the dependence on state, and simply define $\beta_k \in \mathbb{R}^{n\times m}$ as the *benefit matrix* at time step $k$. $\eta(s) \in \mathbb{R}^{n\times m}$ is the *handover matrix* which *is* dependent on the state. Formally, $\eta(s_k)$ is defined as all remaining components of the value of an assignment,

$$\eta(s_k) := \hat{\beta}(s_k) - \beta_k.$$

In a constellation setting, $\eta(s_k)$ might be the contribution of satellite charge levels or orientations, which clearly do depend on the state of the constellation. By contrast, $\beta_k$ might

[1]Much of this section is the generalization of the problem statement used in the author's previous work. See [31] for a more detailed treatment of the algorithm discussed in this chapter.

be the distance between tasks and satellites at time $k$; assuming satellites cannot change their orbits, this is independent of state transitions. Importantly, property 3.1 implies that we can have an accurate estimate of $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_T)$ before the optimization process begins.

We can now quantify the value contributions of each part of $\hat{\beta}(s_k)$. Given benefit and assignment matrices, the *time step benefit* $B : \mathbb{R}^{n\times m} \times X \to \mathbb{R}$ is the sum of the constant benefits over each assigned agent-task pair:

$$B(\beta, x) := \sum_{i=1}^{n}\sum_{j=1}^{m} \beta_{ij}\, x_{ij}$$

Similarly, the *time step handover value* $H : S \times X \to \mathbb{R}$ is the value generated by making assignment $x$ which depends explicitly on the state $s$:

$$H(s, x) := \sum_{i=1}^{n}\sum_{j=1}^{m} [\eta(s)]_{ij} x_{ij} \tag{3.2}$$

Bringing these definition together, we can say that the *time step value* $V : \mathbb{R}^{n\times m} \times X \times X \to \mathbb{R}$ is the total benefit subject to the handover value for the pair of assignments:

$$V(\beta, s, x) := B(\beta, x) + H(s, x) \tag{3.3}$$

With these definitions, we can express the total value (1.3a) for the constellation as a sum of time step values

$$\mathcal{V}(\mathbf{x}; \boldsymbol{\beta}) = \sum_{k=1}^{T} V(\beta_k, s_k, x_k), \tag{3.4}$$

where states evolve according to the transition dynamics $s_k = \mathcal{T}(s_{k-1}, x_{k-1})$. Using these definitions, we can define the following lemma:

**Lemma 3.1.1.** *Fix $s \in S$ and $\beta \in \mathbb{R}^{n\times m}$. Then,*

$$\alpha(\beta + \eta(s)) = \operatorname*{argmax}_{x\in X} V(\beta, s, x).$$

*where $\alpha : \mathbb{R}^{n\times m} \to X$ is the optimal assignment function as defined in Section 2.1.*

*Proof.* This follows by plugging in the definition of $V(\cdot)$ and simplifying: see [31]. □

### *3.2 Algorithm Intuition*

Lemma 3.1.1 suggests a naive approach to generating a solution $\mathbf{x}$ to the SAP (1.3)—simply aim to maximize time step value (3.3) at each time step, without regard to future time steps. This strategy is outlined in Algorithm 1.[2]

**Algorithm 1** Handover-Aware Assignment (HAA)

**Given**: $\boldsymbol{\beta} = (\beta_1, \beta_2, ..., \beta_T), s_1 \in S$

1: **for** $1 \leq k \leq T$ **do**

2: $\quad x_k \leftarrow \alpha(\beta_k + \eta(s_k))$

3: $\quad s_{k+1} \leftarrow \mathcal{T}(s_k, x_k)$

4: **end for**

**return** $\mathbf{x} = (x_1, \ldots, x_T)$

Algorithm 1 only uses the benefit matrix at time $k$ ($\beta_k$) in order to compute the assignment matrix $x_k$. The algorithm does not take into account transition dynamics or the information contained in future benefit matrices, of which at least an estimate is often available. For example, in satellite applications the well understood nature of orbital dynamics allows us to compute accurate satellite positions (and thus benefit matrices) hours or even days ahead of time. Such information can be used to select a sequence of assignments that perform better according to total value (3.4), rather than strictly maximizing the time step value (3.3). In the following section, we will introduce the Handover-Aware Assignment with Lookahead (HAAL), an algorithm that leverages this future information in order to improve the total value of assignments at the expense of increased computational cost.

HAAL operates under several informal assumptions on the handover matrix and transition dynamics which hold in many practical applications.

**Definition 3.2.1** (Assignment-dominated transition function)**.** *A transition function* $\mathcal{T}$ :

[2]Algorithm 1 also comes with optimality guarantees—see Appendix A.

*$S \times X \to S$ is "assignment dominated" when $||\mathcal{T}(s,x) - \mathcal{T}(s',x)|| \leq D$ for all $s, s' \in S$, where $||\cdot||$ is some arbitrary measure of distance between states.*

In words, this means that the state of the system is mostly determined by the previous assignment. This might be the case if your state is defined by the orientations of the satellites, as in [31].

**Definition 3.2.2** (Strict penalty handover matrix)**.** *A handover matrix $\eta(s)$ is a "strict penalty" function when it induces a time step handover value function $H(\cdot)$ such that:*

$$||\mathcal{T}(s,x) - s|| \geq ||\mathcal{T}(s,x') - s|| \implies H(s,x) \leq H(s,x').$$

Intuitively, this means that the system penalizes assignments for changing the state too much, as is the case when state transitions come with an associated energy cost.

Consider an assignment $x_k^H$ computed by Algorithm 1 maximizing (3.3) at state $s_k$, in a system with assignment-dominated transition dynamics and strict penalty handover matrices. If $x_k^H$ leads to a state $s_{k+1} = \mathcal{T}(s_k, x_k^H)$ which is poor such that the value of the next assignment $V(s_{k+1}, x_{k+1}^H)$ is relatively small, a better strategy might be to select some $\tilde{x} \in X$ that satisfies the following inequalities:

$$V(\beta_k, s_k, \tilde{x}) \leq V(\beta_k, s_k, x_k^H) \tag{3.5a}$$

$$V(\beta_k, s_k, x_k^H) + V(\beta_{k+1}, s_{k+1}, x_{k+1}^H) < V(\beta_k, s_k, \tilde{x}) + V(\beta_{k+1}, \tilde{s}_{k+1}, \tilde{x}) \tag{3.5b}$$

That is, $\tilde{x}$ yields slightly less value in time step $k$ (3.5a) but performs significantly better in time step $k+1$ (3.5b). Since $\tilde{x}$ remains constant between $k$ and $k+1$ and the transition dynamics are assignment-dominated, $s_k \approx \tilde{s}_{k+1}$ and we can rewrite the time step value as follows:

$$V(\beta_k, s_k, \tilde{x}) + V(\beta_{k+1}, \tilde{s}_{k+1}, \tilde{x}) \approx V(\sum_{t=k}^{k+1} \beta_t, s_k, \tilde{x}) \tag{3.6}$$

Now, we have a single value function that can be used to determine $\tilde{x}$ optimally using standard assignment problem methods (Lemma 3.1.1), without having to resort to more expensive QAP or multi-assignment algorithms.

This is the operating principle behind HAAL: a "model-predictive-control-like" approach instead of a greedy one [51]. First, the algorithm partitions the time interval $[1, L]$, where $L$ is a user-defined lookahead window, into a sequence of sub-intervals (Figure 3.1). For example, $([1, 2], [3, L])$. Then, a subroutine of Algorithm 1 computes a sequence of assignments by requiring the assignments on each sub-interval to be constant. Next, the algorithm repeats this process over each possible partitioning of $[1, L]$. The first assignment of the most valuable assignment sequence is then executed. Last, the process repeats for each time step. We formalize this algorithm in the following sections.

### *3.3 Handover Aware Assignment with Lookahead (HAAL) Algorithm*

In this section, we present Handover Aware Assignment with Lookahead (Algorithm 2), an algorithm for generating solutions to the SAP. First, a few preliminaries.

In our discrete setting, *time intervals* are defined as integer tuples where $a \leq b$. A *time interval sequence* is a sequence of time intervals $([a_1, b_1], [a_2, b_2], \ldots, [a_N, b_N])$ such that $a_{k+1} = b_k+1$. For example, $([1, 1], [2, 3], [4, 6])$ is a valid time interval sequence, but $([1, 1], [2, 3], [5, 6])$ and $([1, 1], [2, 3], [3, 6])$ are not. A time interval sequence *of length* $L$ is a time interval sequence such that $a_1 = 1$ and $b_N = L$, where $N$ is the number of time intervals in the time interval sequence. We denote the collection of all time interval sequences of length $L$ as $Q_L$. For example, $Q_2 = \{([1, 1], [2, 2]), ([1, 2])\}$, while Figure 3.1 demonstrates $Q_4$. We can think of $Q_L$ as the collection of all partitionings of $[1, L]$ in which each partition is a time interval. Note that the cardinality of this set can be computed for a given $L$ as $|Q_L| = 2^{L-1}$.

We can also define the set $I_L := \{[a, b] : a, b \in \mathbb{N}, 1 \leq a \leq b \leq L\}$, which can be interpreted as the set containing all unique time step intervals in $Q_L$; note that $|I_L| = L(L + 1)/2$.

We are given the same inputs as Algorithm 1 with the addition of a lookahead window $L$. This parameter indicates that at time $k$, the algorithm only uses information from at most the next $L$ benefit matrices $\beta_k, \beta_{k+1}, \ldots, \beta_{\min(T,k+L)}$. To manage computational cost we can set $L \leq T$, because $|Q_L|$ (and thus the number of assignment problems we need to solve)

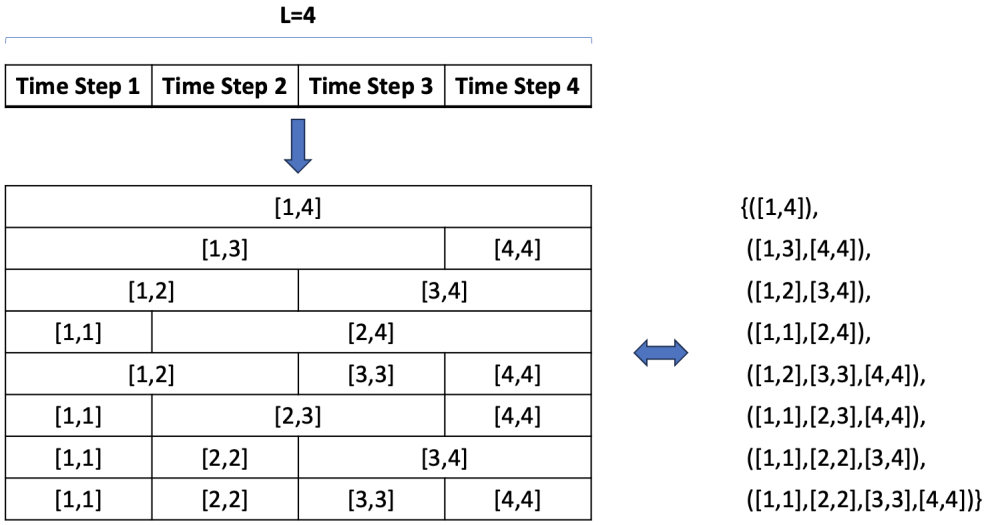



Figure 3.1: Contents of the set $Q_4$.

scales exponentially in $L$. We will see later that the algorithm does not need to consider information from all $T$ time steps at once in order to perform well.

At each time step $k$, the algorithm first determines the length of the effective lookahead window $L' = \min(L, T - k)$ in line 2. Next, the algorithm iterates over each time interval sequence in $\mathbf{q} \in Q_{L'}$, aiming to find the $\mathbf{q}^* \in Q$ which maximizes value using constant assignments in each of its constituent time intervals.

Accordingly, for each $\mathbf{q}$, the algorithm iterates over each time step $t$ in the lookahead window $L'$. At the start of each new time interval $[a, b]$, we compute the assignment $x^{\mathbf{q}}_{[a,b]}$ yielding the highest value when held constant over the interval. In order to find $x^{\mathbf{q}}_{[a,b]}$, we apply the same insights used in Equation 3.6 to note that we can compute an optimal, constant assignment over the time step interval $[a, b]$ by maximizing value using the sum of all the benefit matrices in the time interval ($\beta_{[a,b]}$, line 8). Thus, we solve a standard assignment problem using $\beta_{[a,b]}$ (line 9) and execute this assignment throughout the time step interval, tracking the value this yields.

Once this process is complete for every time interval sequence $\mathbf{q} \in Q_{L'}$, we execute

$x^{\mathbf{q}^*}_{[1,b]}$, the first assignment from the highest value time interval sequence (line 16). Note that assignments from other time intervals in $\mathbf{q}^*$ are discarded, to be recomputed in the next time step when the algorithm has more information about the future. This process continues until an assignment is generated at each time step $k$.

Note that when $L = 1$, Algorithms 1 and 2 are equivalent, and when $L = T$, Algorithm 2 is a strict improvement over Algorithm 1.

**Algorithm 2** Handover-Aware Assignment w/ Lookahead (HAAL)

**Require:** $\boldsymbol{\beta} = (\beta_1, ..., \beta_T), s_1 \in S,\ L \geq 1$

1: **for** $1 \leq k \leq T$ **do**
2: $\quad L' \leftarrow \min(L, T - k)$
3: $\quad$ **for** $\mathbf{q} \in Q_{L'}$ **do**
4: $\quad\quad s^{\mathbf{q}}_1 \leftarrow s_k,\ \ v^{\mathbf{q}} \leftarrow 0$
5: $\quad\quad$ **for** $t = 1, ..., L'$ **do**
6: $\quad\quad\quad [a, b] \leftarrow \{[a', b'] : a' \leq t \leq b',\ [a', b'] \in \mathbf{q}\}$ *(get the current time interval)*
7: $\quad\quad\quad$ **if** $t = a$ *(this is the first time step in the time interval)* **then**
8: $\quad\quad\quad\quad \beta_{[a,b]} \leftarrow \sum_{t'=a}^{b} \beta_{k+(t'-1)}$
9: $\quad\quad\quad\quad x^{\mathbf{q}}_{[a,b]} \leftarrow \alpha(\beta_{[a,b]} + \eta(s^{\mathbf{q}}_t))$
10: $\quad\quad\quad$ **end if**
11: $\quad\quad\quad v^{\mathbf{q}} \leftarrow v^{\mathbf{q}} + V(\beta_{k+(t-1)}, s^{\mathbf{q}}_t, x^{\mathbf{q}}_{[a,b]})$
12: $\quad\quad\quad s^{\mathbf{q}}_{t+1} \leftarrow \mathcal{T}(s^{\mathbf{q}}_t, x^{\mathbf{q}}_{[a,b]})$
13: $\quad\quad$ **end for**
14: $\quad$ **end for**
15: $\quad \mathbf{q}^* \in \underset{\mathbf{q} \in Q_{L'}}{\operatorname{argmax}}\ v^{\mathbf{q}}$
16: $\quad x_k \leftarrow x^{\mathbf{q}^*}_{[1,b]}$ *(execute the assignment corresponding to the first time interval in $q^*$)*
17: $\quad s_{k+1} \leftarrow \mathcal{T}(s_k, x_k)$
18: **end for**

**return** $\mathbf{x} = (x_1, \ldots, x_T)$

### *3.4 Distributed HAAL (HAAL-D) Algorithm*

One attractive feature of Algorithm 2 is that we can design a completely distributed version of the algorithm that can operate without central control and under limited communications. Algorithm 3 is the distributed version of Algorithm 2.

We model a group of agents as an undirected graph $\mathcal{G} = ([n], \mathcal{E})$. The edge set $\mathcal{E}$ is constructed by $\{i, j\} \in \mathcal{E}$ if and only if agents $i$ and $j$ can exchange information. This graph formulation is clearly motivated when generating solutions to realistic problems such as satellite constellations, as we will see in Chapter 4.

Only a few key differences distinguish Algorithms 2 and 3. First, Algorithm 3 computes assignments for each time step interval $[a, b] \in I_{L'}$ in parallel rather than in series. This is feasible since while in Algorithm 2 $\eta(\cdot)$ is computed using the most recent state in that time interval sequence (Algorithm 2 line 9), in Algorithm 3 we instead generate a handover matrix based solely on the state at the beginning of the time interval sequence, $s_k$ (line 6). This means that handover matrices in one time interval never depend on the results of the optimization for another time interval, and thus we can solve all $L(L+1)/2$ assignment problems at the same time. This is an approximation (which could also be used to parallelize Algorithm 2) and increases the size of satellite communications with $O(L^2)$, but importantly allows us to run multiple assignment auctions without meaningfully increasing the *number* of communications needed. By contrast, solving these assignment problems sequentially would result in the number of communications growing with $O(L\,2^{L-1})$.

Additionally, rather than solving the assignment problems using a centralized method, we can use the distributed method discussed in Section 2.1. Thus, Algorithm 3 initializes $L(L+1)/2$ auctions, one for each time interval in $\mathcal{I}_{L'}$, and solves them in a distributed fashion. When communicating with neighbors, the satellites share their prices and bidders for each of the $L(L+1)/2$ auctions simultaneously. Once all auctions have converged (line 8), all satellites will have computed identical near-optimal assignments $x_{[a,b]} \in X$ for all time intervals $[a, b] \in I_{L'}$.

**Algorithm 3** Distributed Handover-Aware Assignment w/ Lookahead (HAAL-D)

**Require:** $\boldsymbol{\beta} = (\beta_1, ..., \beta_T), s_1 \in S,\ L \geq 1$

**for** $1 \leq k \leq T$ **do**
  $L' \leftarrow \min(L, T-k)$
  **for** $j = 1 \dots n$ *(run in parallel on each satellite)* **do**
    **for** $[a,b] \in I_{L'}$ **do** *(run in parallel)*
      $\beta_{[a,b]} \leftarrow \sum_{t=a}^{b} \beta_{k+(t-1)}$
      $x_{[a,b]} \leftarrow \alpha(\beta_{[a,b]} + \eta(s_k^j))$ *(solve in a distributed way i.e. [68])*
    **end for**
    Wait for auctions for all $[a,b] \in I_{L'}$ to converge
    **for** $\mathbf{q} \in Q_{L'}$ **do**
      $s_1^{\mathbf{q}} \leftarrow s_k^j,\ v^{\mathbf{q}} \leftarrow 0$
      **for** $t = 1, ..., L'$ **do**
        $[a,b] \leftarrow \{[a',b'] : a' \leq t \leq b',\ [a',b'] \in \mathbf{q}\}$
        $v^{\mathbf{q}} \leftarrow v^{\mathbf{q}} + V(\beta_{k+(t-1)}, s_t^{\mathbf{q}}, x_{[a,b]})$
        $s_{t+1}^{\mathbf{q}} \leftarrow \mathcal{T}(s_t^{\mathbf{q}}, x_{[a,b]})$
      **end for**
    **end for**
    $\mathbf{q}^* \in \underset{\mathbf{q} \in Q_{L'}}{\operatorname{argmax}}\ v^{\mathbf{q}}$
    $x_k^j \leftarrow x_{[1,b]}^{\mathbf{q}}$ *(execute the assignment corresponding to the first time interval in $q^*$)*
    $s_{k+1}^j \leftarrow \mathcal{T}(s_k^j, x_k^j)$
  **end for**
**end for**

**return** $\mathbf{x}^i = (x_1^i, \dots, x_T^i),\ \forall i = 1 \dots n$

With access to this information, each satellite independently computes the time interval sequence $\mathbf{q}^* \in Q_{L'}$ that provides the most value and executes the first assignment from this sequence (lines 9 to 18), as in Algorithm 2. This process continues until an assignment is generated at each time step $k$.

### *3.5 Assumptions and Theoretical Guarantees*

One key assumption we make in writing Algorithm 3 is that we assume all agents have knowledge of $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_T)$ and $x_0$, and that each agent can simulate the state of the entire system (Line 19). However, this can be relaxed such that each agent only needs to be able to assess the value of an assignment for itself (i.e. have knowledge of its own constant benefits and be able to simulate aspects of the state which contribute to its own value).

In such cases, agents know how valuable a time interval sequence is for themselves, but not for others: accordingly, an extra stage of the algorithm where agents communicate the value they receive from each time interval sequence must be added after auctions converge. Once this information is known by all agents, each agent selects the sequence and corresponding assignment which provides the most value to the group as a whole. However, this is still an important limitation, and serves as motivation for an RL approach.

Based on the properties of $\hat{\boldsymbol{\beta}} = (\hat{\beta}_1(\cdot), \cdots, \hat{\beta}_T(\cdot))$, we can also make strict guarantees on the performance of Algorithms 2 and 3 and determine the conditions under which they perform similarly.

We begin by defining a property of handover matrices:

**Definition 3.5.1** (Vanishing handover matrix)**.** *A handover matrix $\eta(s)$ is "vanishing" if:*

$$x_{ij} = 1 \implies [\eta(\mathcal{T}(s,x))]_{ij} \geq 0 \text{ for all } s \in S.$$

In words, this means that no penalty is applied when assignments stay the same. Note that this implies:

$$H(\mathcal{T}(s,x),x) \geq 0 \quad \forall s \in S,\ x \in X. \tag{3.7}$$

Let $|H| = \max_{s \in S,\ x \in X} H(s,x) - \min_{s \in S,\ x \in X} H(s,x)$ be the maximum difference between any two handover values. Let $\underline{B} = \min_{x \in X,\ k} B(\beta_k, x)$ be a lower bound on the time step benefits.[3] Then, let $c = \min\{\tilde{c} : \underline{B} \geq |H|\tilde{c}\}$ be the ratio of the minimum time step benefit to the maximum difference in handover values.

**Theorem 3.5.1** (Optimality of Algorithm 2)**.** *Suppose $c > 0$ and $\eta(\cdot)$ is vanishing. Let $x_0 \in X$ be an arbitrary initial assignment. Let $\mathbf{x}^H = (x_1^H, \cdots, x_T^H)$, $\mathbf{s}^H = (s_1^H, \cdots, s_T^H)$ be the assignments and states returned by Algorithm 2 using the lookahead window $L$. Let $\mathbf{x}^* = (x_1^*, \cdots, x_T^*)$, $\mathbf{s}^* = (s_1^*, \cdots, s_T^*)$ be optimal assignments and states with respect to (1.3). Then, $\mathcal{V}(\mathbf{x}^H) \geq \frac{c}{1+c}\mathcal{V}(\mathbf{x}^*)$.*

*Proof.* The proof leverages the fact that assignments produced by Algorithm 2 will never be more than $|H|$ value away from optimality at each time step, which bounds the overall distance from the optimal solution. See Appendix A for details. □

Intuitively, this means that the less meaningful the state dependent portion of value ($\eta(s)$) is compared to the static benefits which we can characterize ahead of time ($\beta$), the closer our solution is to optimality.

Unfortunately, no such guarantees can be made about Algorithm 3 due to the fact that handover values are computed with respect to the existing state, rather than the previous state in the time interval sequence (Algorithm 3 Line 6) in order to enable parallelization. While no formal proof exists, we can hypothesize about the conditions under which this simplification is valid, and accordingly when a similar optimality bound can be expected to hold.

**Conjecture 3.5.1** (Validity of Algorithm 3 Parallelization Assumption)**.** *As performed in Algorithm 3, computing a handover value with respect to the previous state of the agent ($s_k$) rather than the current state in the time interval sequence ($s_t^{\mathbf{q}}$) is a valid approximation if and only if $\eta(s)$ is a strict penalty handover matrix for all $s \in S$.*

---

[3] One can replace $\underline{B}$ with $\underline{\beta} = \min_{i,j,k}[\beta_k]_{ij}$ for a more conservative but easier to compute bound.

The intuition for this conjecture is that if $\eta(\cdot)$ is a strict penalty handover matrix, changes in state will be disincentivized and the states will tend to change slowly. Thus, there is reason to believe that in these scenarios, $s_k$ and $s_t^{\mathbf{q}}$ will be very similar and thus $\eta(s_k) \approx \eta(s_t^{\mathbf{q}})$.

As we will see in Chapter 4, empirically HAAL and HAAL-D perform almost identically when $\eta(s)$ is a strict penalty handover matrix for all $s \in S$.

# Chapter 4

# CLASSICAL OPTIMIZATION EXPERIMENTS

In this Chapter, we apply HAAL and HAAL-D to a variety of realistic application areas and show that it has vastly improved performance compared to the current state of the art.

## *4.1 Satellite Constellation Experiment Preliminaries*

We use satellites in a constellation as agents and points on the Earth's surface as tasks. Simulated satellites are in circular orbits with an altitude of 550 km and are able to complete tasks within a 120° FOV assuming the satellite is nadir pointing. We form a communication graph $\mathcal{G}$ by assuming satellites can communicate with satellites within 2500 km via inter-satellite laser links (ISLs). See Figure 4.1 for a visualization of a typical orbital configuration.

We define $\mathbf{P} = (P_1, \cdots, P_T)$ where $[P_k]_{ij}$, the *proximity* of satellite $i$ to task $j$ at time $k$, is defined as follows:

$$[P_k]_{ij} := \begin{cases} \exp\left(-\frac{[\theta_k]_{ij}^2}{2\sigma^2}\right) & \text{if } [\theta_k]_{ij} \leq \theta_{\text{FOV}} \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

where $[\theta_k]_{ij} \in [0°, 360°)$ is the angle between satellite $i$ and task $j$ at time step $k$ (shown in Figure 4.1) and $\sigma := \frac{\theta_{\text{FOV}}}{\sqrt{-2\log(0.05)}}$ is chosen so that when $[\theta_k]_{ij} = \theta_{\text{FOV}}$, we will have $[P_k]_{ij} = 0.05$. We chose a Gaussian function to mimic the deterioration of Gaussian beam intensity as pointing angle increases.

This proximity $[P_k]_{ij}$ defines how "close" a satellite is to a given task, with $[P_k]_{ij} = 1$ representing a maximum closeness. Often, this proximity is an important part of the static benefits of an assignment ($\beta$), with more benefit for tasks which are closer to the satellite.

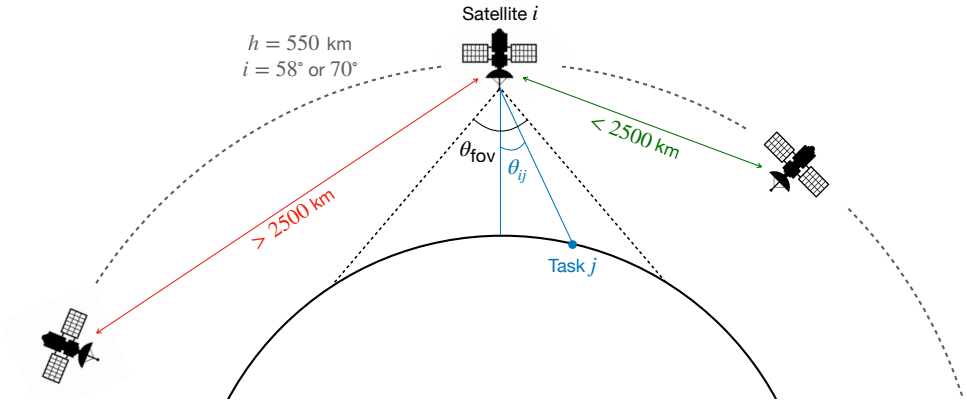



Figure 4.1: Diagram of orbital configuration for satellite constellation experiments; the satellites in the constellations are at an altitude of 550 km with evenly distributed orbital planes at the inclination of 58 or 70 degrees. Furthermore, it is assumed that the satellites can exchange messages over inter-satellite links within the range of 2500 km.

Thus, we define the static component of benefits as a function of this proximity as follows:

$$[\beta_k]_{ij} = \beta_j^{\max}[P_k]_{ij} \quad \forall i, j, k. \tag{4.2}$$

where $\beta_j^{\max}$ is a measure of the priority associated with a given task $j$.

Using a high-fidelity orbital mechanics simulation, we compute $\mathcal{G}$ and $P_{ij}$ for every satellite-task pair at 1-minute intervals over the course of an entire orbit ($T = 93$), and use this information to build the benefit matrices $\boldsymbol{\beta}$ with which we compute assignments.

In this section, we compare Algorithm 2 and 3 to the following centralized and distributed task assignment procedures[1]:

- *No handover ($\alpha(\beta)$)*: The output assignment sequence is computed as $\mathbf{x} = (\alpha(\beta_1), \ldots \alpha(\beta_T))$, without regard for the handover penalty.

[1]Python implementations of HAAL and comparison algorithms can be found at https://github.com/Rainlabuw/handover-aware-assignment.

- *Greedy (GA)*: Each satellite chooses the task of maximal benefit and sequentially executes that same task until the task is out of view. Then, the satellite again chooses the assignable task of maximal benefit, with conflicts resolved centrally by assigning tasks to the satellite of lower index.

- *Consensus-Based Bundle Algorithm (CBBA)*: We apply CBBA (a fully distributed algorithm) to our problem setting, modifying it to only consider a lookahead window of length $L \leq T$ at any given time step. Each "bundle" is then size $L$, and tasks are scored according to handovers induced from previously selected tasks in the bundle. The reader is directed to [10] for further details on the CBBA algorithm.

### *4.2 Simple Earth Coverage Experiments*

We test HAAL in several simple environments to demonstrate its performance and ability to scale to large constellations.[2]

Consider the simple handover penalty where a penalty of $\lambda$ is applied when a satellite changes the task it is assigned to. Thus, the state consists of the previous assignment, $s_k := [x^{s_k}]$, $x^{s_k} := x_{k-1}$, where the superscript $s$ denotes it is a component of state $s_k$. Formally,

$$[\eta(s_k)]_{ij} = \begin{cases} 0 & \text{if } x_{ij}^{s_k} = 1 \\ \lambda & \text{otherwise} \end{cases}, \quad \forall k.$$

Note that this handover matrix is vanishing and a strict penalty and thus preserves the optimality guarantee and (suspected) validity of the parallelization assumption mentioned in Chapter 3.

---

[2]This experiment is adapted from the author's prior work in [31].

### *4.2.1 Midsize LEO internet constellation with randomly placed tasks*

Our first experiment consists of 324 satellites (18 evenly spaced orbital planes at 58° inclination, 18 satellites each) which aim to complete 450 tasks randomly placed on the surface of the Earth—this simulates users of an internet service, which might be inconsistently placed across the Earth's surface (see Figure 4.2 for placement of tasks).

Benefits for satellite-task assignments are calculated according to Equation 4.2, where $\beta_j^{\max} \sim U(1, 2)$, and $\lambda = 0.5$.

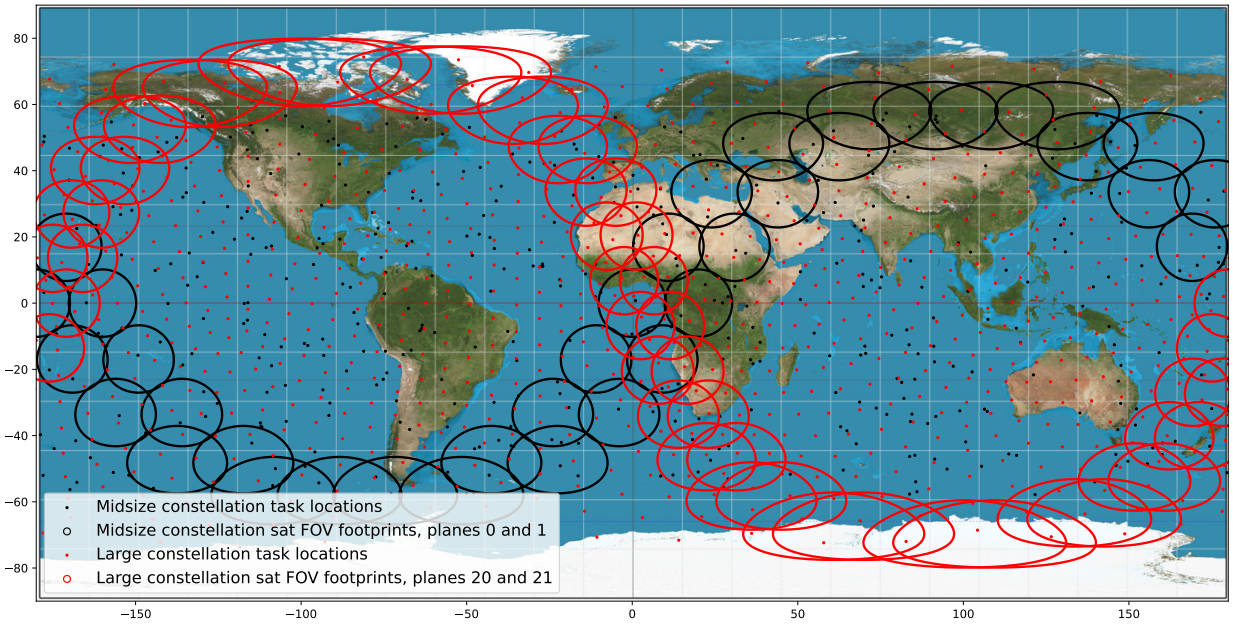



Figure 4.2: Sample ground tracks and task locations for constellation experiments. 2/18 and 2/40 orbital plane field-of-views are shown for the midsize and large constellations, respectively.

*Value performance*

Figure 4.3 demonstrates that Algorithms 2 and 3 deliver far higher total value than other state-of-the-art task allocation algorithms. Both Algorithm 2 and Algorithm 3 perform significantly better than all other tested algorithms at all values of $L$, yielding $\approx 50\%$ higher value than CBBA, the next best algorithm in this setting. Also note that despite the several slight differences between HAAL and HAAL-D, performance does not suffer when moving to the distributed setting. This supports Conjecture 3.5.1.

*Communication requirements*

In the distributed auction subroutine ([68]) we use on Algorithm 3 Line 6 (as well as in CBBA), agents need to repeatedly communicate with neighbors in order to converge on a near-optimal, conflict-free assignment in finite time. Thus, in Figure 4.3, we measure the number of rounds of communications necessary for convergence as $L$ increases.

Because we run our assignment problems in parallel (Algorithm 3 Line 4), we observe that communication requirements appear to scale at a rate far less than the rate of growth of the number of assignment problems being solved, $L(L+1)/2$. The increase in necessary communications that does exist can be attributed to the fact that as the lookahead window increases in satellite constellation scenarios, the satellites bidding on the same tasks are further away from each other in the communication graph. Note that even with 324 satellites and $L = 6$, the communication requirements are minimal—in order to assign tasks within the available 60 seconds, satellites need only to send approximately 80 messages to neighbors.

### *4.2.2 Large-scale Earth-coverage observation constellation*

To demonstrate the scalability and performance of the algorithm at realistic constellation sizes, we apply Algorithm 3 to a constellation of 1000 satellites (40 planes of 25 satellites each, $i = 70°$). We place 812 tasks evenly across Earth from latitude $-70°$ to $70°$, with 188 "dummy" tasks providing zero benefit added to ensure that $n \leq m$ (see Figure 4.2).

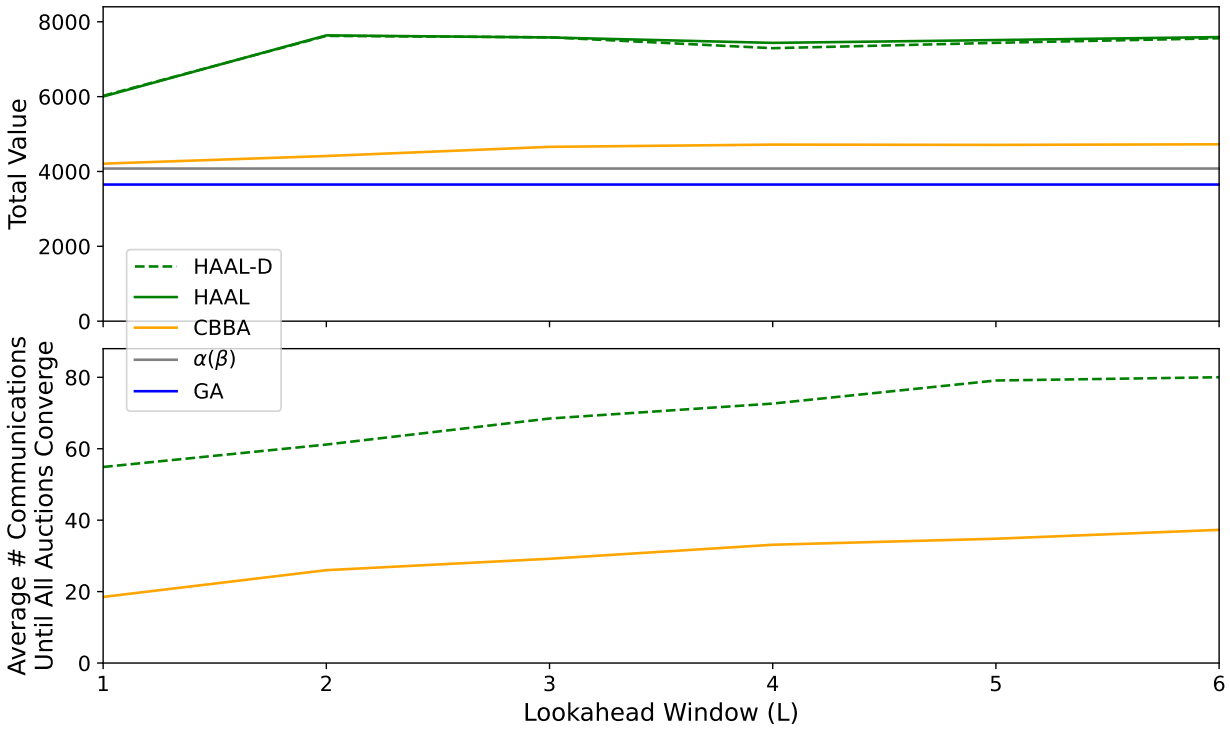
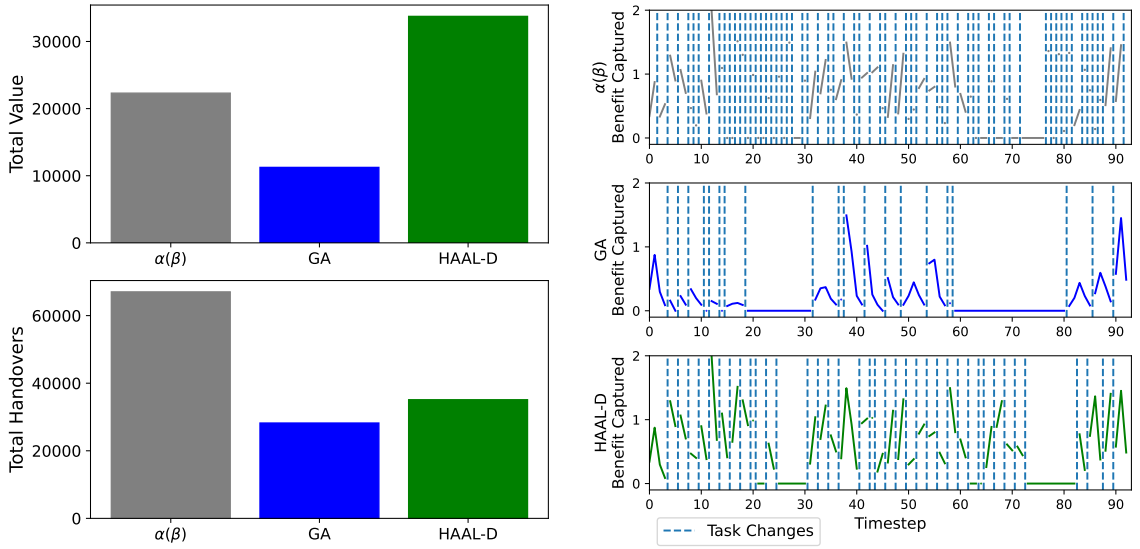



Figure 4.3: Value captured (higher is better) and number of communications needed for convergence (lower is better) for Algorithm 3 and other algorithms in a midsize satellite constellation.

This mimics a constellation designed to continuously observe the entire surface of Earth. We again randomly set the benefits of each non-dummy task according to Equation 4.2, $\beta_j^{\max} \sim U(1,2)$, and use $\lambda = 0.5$. We then apply Algorithm 3 (with $L = 6$), GA, and $\alpha(\beta)$ over a full orbit, tracking captured value and total number of task handovers (CBBA is not tested as it becomes computationally infeasible for large constellation sizes).

Figure 4.4 provides results of these experiments. As shown in Figure 4.4a, although GA achieves slightly fewer handovers, Algorithm 3 again obtains far more value than other algorithms in the literature. Figure 4.4b provides a visual explanation of how Algorithm 3 achieves this. The y-axis of each plot tracks the raw benefit (not including handover penalty)




(a) Value obtained (higher is better) and number of handovers induced (lower is better) for $\alpha(\beta)$, GA, and HAAL-D.

(b) History of benefits captured by a single satellite at each time step for $\alpha(\beta)$, GA, and HAAL-D. Each dotted blue line indicates a change in task assignment.

Figure 4.4: Performance of HAAL applied to large-scale constellation.

of the task achieved by a representative satellite at each time step, with task handovers indicated by vertical lines. Although $\alpha(\beta)$ consistently assigns satellites to tasks with high benefit, it requests a large number of task handovers (i.e. compare $\alpha(\beta)$ and HAAL-D in time steps 19-28). Conversely, GA requests slightly fewer task handovers than HAAL-D, but consistently obtains less benefit in each time step and is characterized by long periods of satellite inactivity (i.e. compare GA and HAAL-D in time steps 0-40). Visually we can see that our algorithm strikes a balance, handing over tasks only when it can obtain several time steps of benefit.

### *4.3 Object Tracking Experiment (Multi-Robot Tasks)*

The SAP framework can also be applied creatively to many other types of problems. The following example will demonstrate how to create a more complicated handover matrix $\eta(\cdot)$, as well as how to extend the assignment problem framework so that each task can be accomplished by more than one agent (i.e. ST-MR-TA settings as defined in [25]).

Suppose it is desired to create a constellation of satellites which can continuously monitor the entire United States and track the movement of planes as they take off and move through US airspace, even when their departure time is not known a priori.

In such a scenario, one could cover the US with hexagonal regions and associate each with a task that should be completed every time step. When a target is in the associated region, observing the region provides a large benefit proportional to $\beta^t$, while observing empty regions provides benefit proportional to $\beta^e << \beta^t$ to ensure that agents have an incentive to observe regions in the absence of targets. This scenario is depicted in Figure 4.5.

Additionally, in this scenario we assume that $N$ satellites can meaningfully observe a region at a single time. To allow for this in the formulation of the SAP (Equation 1.3), we add additional *synthetic tasks* for each duplicated observation of a region. Note that in this new formulation, there are again $n$ satellites and $T$ time steps, but there are now $mN$ tasks.

We need to redefine our task labeling to account for these synthetic tasks. Thus, we define a mapping from each synthetic task $g \in [mN]$ to a real task, $T : [mN] \to m$, and a mapping from each synthetic task to the observation number of the task it corresponds to, $O : [mN] \to [N]$.

Agents receive diminishing returns for completing tasks with higher observation numbers to incentivize them to only duplicate observations when targets are present. Then, the state-dependent rewards are defined as follows:

$$[\hat{\beta}(s_k)]_{ij} = \begin{cases} \gamma^{O(j)-1}[P_k]_{iT(j)}(\beta^e + \mathcal{N}^t_{T(j)k}\beta^t) & \text{if } \exists g \text{ s.t. } [x_{k-1}]_{ig} = 1,\ T(g) = T(j) \\ 0 & \text{otherwise } \textit{(i.e. region not observed previously)} \end{cases} \tag{4.3}$$

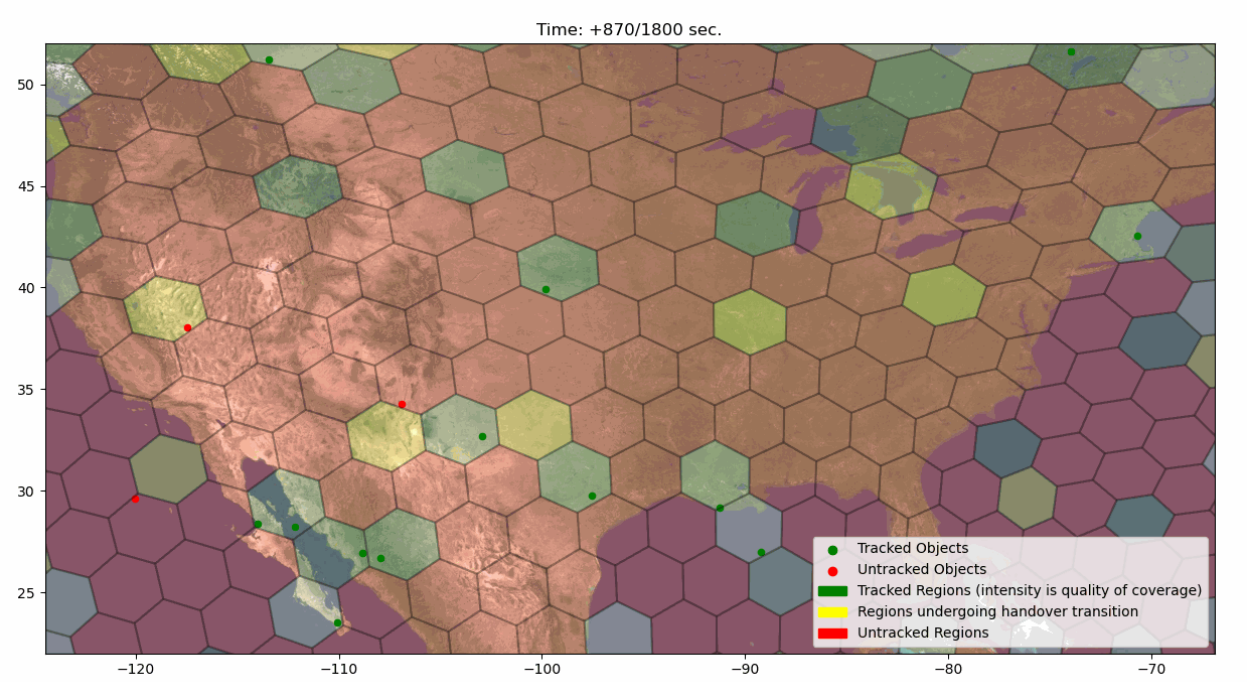



Figure 4.5: Visualization of target-tracking scenario. Targets are pictured as dots, which move horizontally or vertically through the grid regions. Regions can either be observed (green), unobserved (red), or currently in handover (yellow, i.e. case two of Equation 4.3).

where each region $g$ contains $\mathcal{N}_{gk}^t$ targets at time $k$.

In our experiment, we use $N = 2$, $\beta^e = 1$, $\beta^t = 10$, $\gamma = 0.2$, and $\lambda = 0.5$. HAAL exhibits strong performance "out of the box" when applied to this problem setting, where it has no knowledge of the targets until the moment they appear. A target is considered "tracked" if the region it is in is observed, and had also been observed in the previous time step (i.e. satellites had time to "set up" before beginning to observe the target).

Results comparing HAAL to $\alpha(\beta)$ in this setting are presented in Table 4.1. Clearly, HAAL represents a near doubling of performance in both metrics. Qualitatively, we observe that satellites using HAAL observe regions that targets are about to enter before the target reaches the region in order to avoid receiving the penalty corresponding to setting up.

| | $\alpha(\beta)$ | **HAAL** |
|---|---|---|
| % of objects tracked | 42.7% | **85.4%** |
| Total value captured | 2141.6 | **4367.2** |

Table 4.1: Object tracking results.

### *4.4 Multi-Beam Internet Satellite Experiment (Multi-Task Robots)*

We can also apply HAAL, and the assignment problem framework more generally, to problems where satellites can complete multiple tasks each (i.e. MT-SR-TA settings as defined in [25]). This problem formulation is critical in applications like internet constellations, where a single satellite can often provide internet to multiple regions at once via phased array antennas [20].

To formulate a MT-SR-TA problem in the standard assignment problem framework with the previously used valid assignment set $X$, one can simply add duplicate agents to the environment for each separate task the agent can complete. For example, if each agent can complete at most $M$ tasks, then our environment would contain $nM$ agents. If $nM > m$, one can add $nM - m$ dummy tasks to the environment with uniformly zero benefit to ensure that $nM \leq m$. We call these extra agents added to account for agents' ability to complete multiple tasks *synthetic satellites*.

We need to redefine our agent labeling to account for these synthetic satellites. Thus, we define a mapping from each synthetic satellite $l \in [nM]$ to a real satellite, $R : [nM] \to [n]$, and a mapping from a synthetic satellite to the beam number on the real satellite, $B : [nM] \to [M]$.

Finally, to incentivize sharing of the tasks among satellites, agents receive diminishing returns for completing more tasks. This means that synthetic satellites corresponding to higher beam numbers receive less benefit for completing tasks. Then, we define the state-

dependent benefit matrix for this setting as follows:

$$[\hat{\beta}(s_k)]_{ij} = \begin{cases} \gamma^{B(i)-1}[\beta_k]_{R(i)j} & \text{if } \exists l \text{ s.t. } [x_{k-1}]_{lj} = 1,\ R(l) = R(i), \\ \gamma^{B(i)-1}[\beta_k]_{R(i)j} - \lambda & \text{otherwise.} \end{cases}$$

We apply this state-dependent benefit matrix to a group of satellites over the United States, using $M = 10$, $\gamma = 0.9$, and $\lambda = 0.5$, and where static benefits are computed according to Equation 4.2 with $\beta_j^{\max} \sim U(1, 2)$. Results for this experiment are shown in Table 4.2, indicating that HAAL performs almost 3x better than a greedy approach in this setting.

| | $\alpha(\beta)$ | **HAAL** |
|---|---|---|
| Value captured | 1232.2 | **3334.6** |

Table 4.2: Multi-beam satellite results.

In this chapter we have shown that HAAL exhibits strong performance across several types of satellite assignment problem where task handovers are generally undesirable. We have also shown that the assignment problem framework can naturally be extended to cases where agents can complete more than one task, or where tasks can be completed by more than one agent, by adding synthetic satellites and tasks. However, in more general SAP settings, a learning-based approach can be helpful, as will be described in Chapter 5.

# Chapter 5

# A LEARNING-BASED APPROACH TO THE SEQUENTIAL ASSIGNMENT PROBLEM

While we have seen in Chapter 4 that HAAL exhibits good performance on several complex problems, it is still limited in several important aspects.

1. **Assumption that task handovers are disincentivized.** HAAL makes the significant assumption that the state dependent component of value, $\eta(s)$, is a strict penalty handover matrix. This makes sense for situations where a change in assignments leads to high fuel costs, but breaks down when agents should be switching tasks regularly (i.e. when agents are taking sensor measurements which yield diminishing returns when measurements are already accurate).

2. **Difficulty with long-term planning.** HAAL has difficulty in acting according to the long-term effects of actions. Although HAAL scales polynomially in the number of agents and tasks, it scales exponentially in the time window $L$, meaning that there will inevitably be future information that HAAL cannot consider. This can be problematic in scenarios with elements which require very long-term planning (i.e. when managing satellite power states).

3. **Scaling issues in distributed case.** In the distributed case, HAAL-D requires that each agent can simulate all aspects of the state that affect its own value. While this may be the case in simple scenarios such as when value depends only on the agent's own position, in other cases the entire state might be required. This is clearly undesirable in large problem settings.

4. **Challenges with handling stochasticity.** HAAL cannot be easily applied in settings where the environment is stochastic.

Reinforcement Learning (RL) is in many ways perfectly positioned to handle these deficiencies. In contrast to HAAL, RL approaches make no assumptions on the structure of the reward function $\mathcal{R}(s, x)$, estimate long-term effects via $Q$-functions, can be conditioned on only partial observations of the full state, and implicitly handle stochasticity through the presence of the expectation $\mathbb{E}^{\pi}$ in the reward function. In comparing the SAP objective (1.3) with the generic RL objective (2.2), we can see that the SAP can easily be expressed as a RL problem, where the equivalencies are made explicit in Table 5.1.

Thus, in this chapter, and based on the author's previous work in [30], we describe how to effectively apply RL methods to the sequential assignment problem.

| **Classic RL Formulation** | **SAP Formulation** |
|---|---|
| $A$ (action space) | $X$ (space of valid assignments) |
| $a \in A$ (action) | $x \in X$ (assignment) |
| $\mathcal{R}(s, x)$ (reward function) | $\sum_{i=1}^{n} \sum_{j=1}^{m} [\hat{\beta}(s)]_{ij} [x]_{ij}$ (time step value function) |

Table 5.1: Mapping between classic RL and SAP problem formulation.

## *5.1 Challenges of RL Formulation*

Although Table 5.1 demonstrates the natural relationship between the SAP and an RL problem, significant challenges arise when attempting to apply RL to such problems in practice. First, RL algorithms scale poorly to large action spaces—when using assignments as actions, $|X| = \frac{m!}{(m-n)!}$, which quickly becomes intractably large as $m$ and $n$ increase.

Thus, it is desirable to use Multi-Agent Reinforcement Learning (MARL) algorithms to handle the large action space. In the most naïve MARL setting, we define a *joint assignment*

$x = (x^1, \cdots, x^n)$ and a *joint policy* $\pi = (\pi^1, \cdots, \pi^n)$, $x^i \sim \pi^i$. The assignment space for a single satellite $x^i := \text{argmax}_j \ x_{ij} \in [m]$ is now a single integer denoting the task satellite $i$ is assigned to. Indeed, other approaches in the literature have applied MARL algorithms to the satellite task assignment problem in this way [37]. However, there are several fundamental issues with this approach.

The first challenge is deciding how to specify the reward function for an agent $i$, $\mathcal{R}_i(s, x)$. Given that our objective is a global maximization over tasks completed by *all* agents rather than any single agent, one might be tempted to use cooperative rewards, where $\mathcal{R}_i(s, x) = \mathcal{R}(s, x)$, as in [50, 57]. However, as we will see later, agents with this reward function struggle to disentangle the effect of their actions among the many other agents in the group, even when applying techniques like COMA, which are designed to enable a single agent to assess its counterfactual impact on the joint reward [23].

Conversely, one might take note of the recent success of completely independent agents in cooperative domains [19, 45, 67] and provide agents with the rewards they yield solely from tasks they complete, $\mathcal{R}_i(s, x^i) = [\hat{\beta}(s)]_{ix^i}$. When specifying rewards in this way, the learning problem is significantly easier because rewards correlate more directly with agent actions, and we hope that cooperation between agents in allocating tasks emerges naturally from the training process. Empirically, however, we see this behavior does not emerge easily, and that many agents simultaneously assign themselves to the most valuable tasks, even if the tasks can only be completed by a single agent.

This leads us to the second difficulty of applying MARL to assignment problems, which is that without an explicit constraint, joint assignments often consist of multiple agents completing the same task, meaning that $x \notin X$. Clearly, some centralization or communication between agents is necessary to ensure that agents do not duplicate assignments. Our method provides a principled approach for allowing agents to resolve conflicts and learn to make socially optimally and valid assignments $x \in X$.

### *5.2 Algorithm Intuition*

The assignment function $\alpha$ introduced in Equation 2.1 is a solution to both of these difficulties. First, note that given information about *the benefit to individual agents* $i$ for completing tasks $j$, $\beta_{ij}$, $\alpha$ yields assignments which are optimal *on the level of the entire group.* Additionally, $\alpha$ yields joint assignments $x \in X$—i.e. those that avoid assigning multiple agents to the same task—by definition.

Thus, we would like to use $\alpha$ as our joint policy, $\pi(s) = \alpha(\beta)$. However, in the SAP setting the state-dependent nature of the planning problem means that we do not a-priori know the long-term benefit of assignments, and thus cannot easily access a $\beta \in \mathbb{R}^{n \times m}$. Instead, we learn the expected future value of each assignment $i \to j$ from experience, and use these values as input to $\alpha$.

Taking a more mathematical approach to motivating this algorithm, recall that many $Q$-learning based algorithms such as DQN [39] require the ability to take actions ($\epsilon$-)greedily with respect to the current policy, $x_k = \text{argmax}_{x^* \in X} Q^\pi(s_k, x^*)$. However, in the assignment setting, acting greedily with respect to the joint policy $\pi$ becomes a difficult non-convex optimization problem. Our method provides a way of approximating this behavior.

A key part of why we can do this is because of a decomposition of the joint $Q$ function that exists in the assignment problem setting (similar to the one used in [57]), which we outline in Theorem 5.2.1.

**Theorem 5.2.1** (Decomposition of $Q^{\pi^\alpha}$ into $Q_i^{\pi^\alpha}$)**.** *Let $\pi^\alpha : S \to X$ be a constant, deterministic joint policy. Define the Q-function for an individual agent with respect to this joint policy $\pi^\alpha$ as:*

$$Q_i^{\pi^\alpha}(s, j) := \mathbb{E}\left[\mathcal{R}_i(s_k, j) + \sum_{t=k+1}^{T} \gamma^{t-k} \mathcal{R}_i(s_t, x_t^i) \;\middle|\; s_k = s, s_{k+1} \sim \mathcal{T}(s_k, \pi^\alpha(s)); \pi^\alpha\right]$$

*Then, in the assignment problem setting, where $\mathcal{R}(s, x) = \sum_{i=1}^{n} \mathcal{R}_i(s, x^i)$,*

$$Q^{\pi^\alpha}(s_k, x) = \sum_{i=1}^{n} \sum_{j=1}^{m} Q_i^{\pi^\alpha}(s_k, x^i) x_{ij} \quad \textit{for} \;\; x = \pi^\alpha(s_k). \tag{5.1}$$

*Proof.* This follows from the fact that in assignment problems, $\mathcal{R}(s,x) = \sum_{i=1}^{n}\sum_{j=1}^{m}\hat{\beta}(s)x_{ij} = \sum_{i=1}^{n}\mathcal{R}_i(s,x^i)$ (that is, the sum of the reward for the individual agents is equal to the joint reward, and the reward for agent $i$ is conditioned only on agent $i$'s assignment, $x^i$). See Appendix B for the complete proof. □

In words, given a state $s_k$, $Q_i^{\pi^\alpha}(s_k,j)$ is defined as the total expected future reward that agent $i$ will obtain, given that agent $i$ is assigned to task $j$, and then that all agents follow the joint policy $\pi^\alpha$ for future assignments. Given the partial observability of our environment, in practice we make the approximation $Q_i^{\pi^\alpha}(s,j) \approx Q_i^{\pi^\alpha}(\mathcal{O}^i(s),j)$.

Here, we can begin to see the clear connection between Equations 1.1 and 5.1; if we define our benefit matrix (previously $\beta$) to be $\mathbf{Q}_k^{\pi^\alpha} \in \mathbb{R}^{n\times m}$ such that $[\mathbf{Q}_k^{\pi^\alpha}]_{ij} = Q_i^{\pi^\alpha}(o_k^i,j)$, then $\text{argmax}_{x^*\in X}\, Q^{\pi^\alpha}(s_k,x^*) \approx \alpha(\mathbf{Q}_k^{\pi})$. Then, we can make assignments according to:

$$x_k = \alpha(\mathbf{Q}_k^{\pi^\alpha}) \approx \underset{x^*\in X}{\text{argmax}}\, Q^{\pi^\alpha}(s_k,x^*). \tag{5.2}$$

While Equation 5.1 only holds for $x = \pi^\alpha(s)$, if we assume that policies change slowly during the learning process such that $\alpha(\mathbf{Q}_k^{\pi^\alpha}) \approx \pi^\alpha(s)$, then Equation 5.2 is a valid approximation. Thus, if we can learn estimates of $Q_i^{\pi^\alpha}(o_k^i,j)$ directly from experience, we can build $\mathbf{Q}_k^{\pi^\alpha}$ and have agents act not by picking assignments that are best for themselves, but through the mechanism $\alpha$ which is guaranteed to return a socially optimal outcome for the group. This motivates our algorithm which we fully specify in the next section.

## *5.3 RL-Enabled Distributed Assignment (REDA) Algorithm*

Algorithm 4 specifies the REDA algorithm (depicted in Figure 5.1) for generating solutions to the SAP, with key differences from a standard independent DQN algorithm (e.g. [3, 60]) highlighted in purple.

### *5.3.1 Bootstrapping from a greedy policy*

Sequential assignment problems are unique in that there always exists a sub-optimal, non-parametrized policy with which we can bootstrap our policy from; $\pi(s) := \alpha(\hat{\beta}(s))$, where

**Algorithm 4** RL-Enabled Distributed Assignment (REDA)

Initialize $Q$-network parameters $\theta$, target $Q$-network parameters $\bar{\theta} = \theta$
Initialize a replay buffer $D$
**for** episode $e = 1, 2, ...$ **do**
  **for** time step $k = 1, ..., T$ **do**
    Collect current joint observation $o_k = (o_k^1, \cdots, o_k^n)$, where $o_k^i \sim \mathcal{O}^i(s_k)$
    With probability $\epsilon$: $x_k \leftarrow \alpha(\hat{\beta}(s_k))$ *(act greedily w/r/t the current benefit matrix)*
    Otherwise:
      Build $\mathbf{Q}_k^\pi$ such that $[\mathbf{Q}_k^\pi]_{ij} \leftarrow Q_i^\pi(o_k^i, j; \theta)$, with $\mathbf{Q}_{\text{avg}} \leftarrow \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} |[\mathbf{Q}_k^\pi]_{ij}|$
      Generate perturbation matrix $\boldsymbol{\xi} \in \mathbb{R}^{n \times m}$ such that $\boldsymbol{\xi}_{ij} \sim N(0, 2\mathbf{Q}_{\text{avg}}\epsilon)$
      $x_k \leftarrow \alpha(\mathbf{Q}_k^\pi + \boldsymbol{\xi})$ *(act ∼optimally w/r/t the estimated values of $Q_i^\pi(o_k^i, j; \theta)$)*
    Collect joint assignment $x_k = (x_k^1, \cdots, x_k^n)$, where $x_k^i \leftarrow \text{argmax}_j [x_k]_{ij}$
    Collect joint reward $r_k = (r_k^1, \cdots, r_k^n)$, where $r_k^i \leftarrow \mathcal{R}_i(s_k, x_k^i)$
    Observe next state $s_{k+1} \sim \mathcal{T}(s_k, x_k)$ and collect joint observation $o_{k+1}$
    Store joint transition $(o_k, x_k, r_k, o_{k+1})$ in replay buffer $D$
    Sample random mini-batch of $B$ joint transitions $(o_t, x_t, r_t, o_{t+1})$ from $D$
    **if** $s_{t+1}$ is terminal **then**
      Targets $y_t^i \leftarrow r_t^i$ for all $i$
    **else**
      Build $\mathbf{Q}_{t+1}^\pi$ such that $[\mathbf{Q}_{t+1}^\pi]_{ij} \leftarrow Q_i^\pi(o_{t+1}^i, j; \theta)$
      $x_{t+1} \leftarrow \alpha(\mathbf{Q}_{t+1}^\pi)$, $x_{t+1}^i \leftarrow \text{argmax}_j [x_{t+1}]_{ij}$
      Targets $y_t^i \leftarrow r_t^i + \gamma Q_i^\pi(o_{t+1}^i, x_{t+1}^i; \bar{\theta})$ for all $i$
    **end if**
    Loss $\mathcal{L}(\theta) \leftarrow \frac{1}{B} \sum_{t=1}^{B} \sum_{i=1}^{n} \left( y_t^i - Q(o_t^i, x_t^i; \theta) \right)^2$
    Update parameters $\theta$ by minimizing $\mathcal{L}(\theta)$
    Update target network parameters $\bar{\theta}$ after a set number of time steps
  **end for**
**end for**

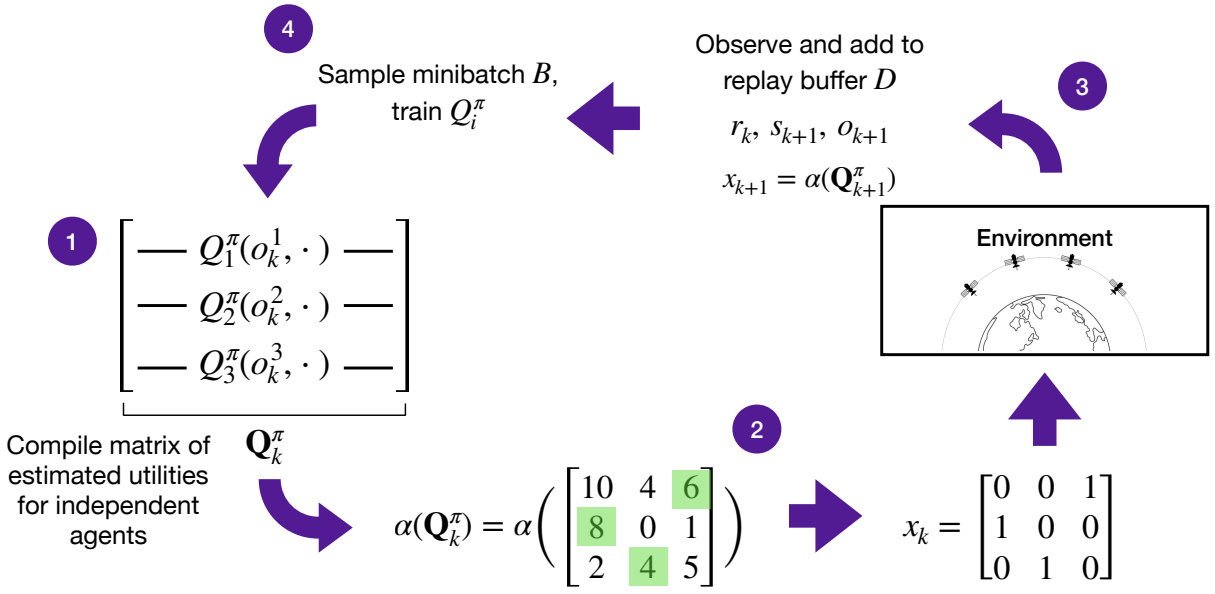



Figure 5.1: Architecture of REDA. 1) Calculate independent estimates of future utility for each agent, combine into a matrix. 2) Select joint assignment $x_k = \alpha(\mathbf{Q}^\pi_k)$ which maximizes social utility, not utility for any given agent. 3) Execute $x_k$ in environment and observe results. 4) Train agents' independent value estimates based on minibatch from replay buffer.

we simply make the greedy assignment with respect to the current benefit matrix at every time step, without regard for future benefits. Note that this approach is entirely analagous to HAA or HAAL with $L = 1$ as described in Chapter 3. At the beginning of training, we act with this greedy policy with probability $\epsilon$, filling our replay buffer with reasonable state-assignment pairs before beginning to learn to improve on this policy.

In problem settings where HAAL is expected to perform well, one could of course replace the non-parametrized policy used for pretraining with HAAL, $L > 1$.

### *5.3.2 Exploration*

To induce further exploration, we also add randomly distributed noise $\boldsymbol{\xi}$ to $\mathbf{Q}^\pi$, scaled by the current average magnitude of $\mathbf{Q}^\pi$, $\mathbf{Q}_{\text{avg}}$, such that sub-optimal joint assignments are selected

with some probability. This is a more effective exploration strategy than making entirely random joint assignments $x \in X$ given the size of the search space, $|X| = \frac{m!}{(m-n)!}$.

### *5.3.3 Target specification*

Another unique aspect of REDA is the way learning targets are specified. Because the policy $\pi$ can only select assignments $x \in X$, targets must also satisfy this constraint. In other words, Lines 19 through 21 express $y = r + \max_{x^* \in X} Q^\pi(s, x)$ rather than $y = r + \max_{x^*} Q^\pi(s, x)$. We find the best assignment $x^* \in X$ by again using $\alpha$—following the standard DQN paradigm [39], $\mathbf{Q}^\pi_{t+1}$ is generating using the value network with parameters $\theta$, but the assignment $x_{t+1}$ is evaluated using the target network with parameters $\bar{\theta}$.

## *5.4 Theoretical Motivation*

To further motivate why REDA produces sensible policies, we can show that the REDA target update causes $Q_i$ to converge to the true $Q^\pi_i$ under reasonable assumptions.

**Lemma 5.4.1.** *Let $Q_i \in \mathcal{Q}$ be an arbitrary Q-function. Let $F : \mathcal{Q} \to \mathcal{Q}$ be the operator corresponding to the REDA target update in the tabular case, without target networks or the greedy guide policy:*

$$(FQ_i)(o^i_k, j) = \mathbb{E}^\pi \left[ \mathcal{R}_i(s_k, j) + \gamma Q_i(o^i_{k+1}, x^i_{k+1}) \right]$$

*Assume each observation-assignment pair $(o^i, j)$ is visited infinitely often under a constant policy $\pi$. Then, if $Q^{n+1}_i \leftarrow FQ^n_i$, $\lim_{n\to\infty} Q^n_i = Q^\pi_i$.*

*Proof.* The REDA target update is analogous to a SARSA update [55], so this can be proven by showing that $F$ is a $\gamma$-contraction on the space of $Q$-functions, and that $Q^\pi_i$ is the unique fixed point of this contraction. See Appendix B for the complete proof. □

The critical assumption inherent in Lemma 5.4.1 is that the policy does not change before each observation-assignment pair is visited infinitely many times. However, assuming that

a sufficiently small learning rate is chosen, Lemma 5.4.1 will approximately hold for REDA and it can be assumed that the $Q$-values used in the mechanism $\alpha$ will converge to their desired values, $Q_i^\pi$.

This has several important implications. First, in situations where agents are providing information to centralized mechanisms for assignment, one often has to consider incentive-compatibility and whether agents are being truthful about the information they provide [52]. Lemma 5.4.1 shows that based on REDA's training process, the values $Q_i(o^i, j)$ used in the assignment mechanism $\alpha$ will indeed converge to $Q_i^\pi(o^i, j)$ as desired, and that agents are not able to act strategically or lie for personal benefit, a claim which we verify by experiment in Chapter 6.

Second, it motivates that REDA is a method of approximating DQN on the joint $Q$-function $Q^{\pi^\alpha}$. Because Lemma 5.4.1 states that given enough updates, $Q_i \to Q_i^{\pi^\alpha}$, Equation 5.2 holds and $\alpha(\mathbf{Q}^{\pi^\alpha}) \approx \text{argmax}_{x^* \in X} Q^{\pi^\alpha}(s, x^*)$. Then, acting according to $x_k = \alpha(\mathbf{Q}_k^{\pi^\alpha} + \boldsymbol{\xi})$ is approximately $\epsilon$-greedy action selection with respect to $Q^{\pi^\alpha}$, and target updates $\sum_{i=1}^n y_k^i = \sum_{i=1}^n r_k^i + \alpha_i(\mathbf{Q}_{k+1}^{\pi^\alpha})$ approximate an optimal target update $y_k = r_k + \max_{x^* \in X} Q^{\pi^\alpha}(s_{k+1}, x^*)$, where $\alpha_i(\beta)$ is the value provided to agent $i$ from assignment $\alpha(\beta)$. Thus, we can expect REDA to inherit similar properties relating to the convergence of $Q^{\pi^\alpha}$ to $Q^{\pi^*}$.

We have now described the REDA algorithm, which elegantly applies principles of MARL to the SAP. In Chapter 6, we will see empirically how REDA performs on realistic satellite constellation assignment problems.

# Chapter 6

# LEARNING-BASED EXPERIMENTS

In this Chapter, we will experimentally verify the performance of REDA in several environments, with Sections 6.1 and 6.4 adapted from our previous work in [30].

## *6.1 Small-Scale Environment Experiment*

We first apply REDA to an extremely simple environment in order to provide intuition about how REDA avoids pitfalls associated with other MARL algorithms.

### *6.1.1 Environment setup*

This environment contains three agents, three tasks, and three states. Agent 1 is the "dictator" in that its assignment fully dictates the state transition, $s_k = x_{k-1}^1$. $\hat{\beta}(s)$ is specified as follows:

$$\hat{\beta}(1) = \begin{bmatrix} 2 & 3 & 0 \\ 0 & 2 & 3 \\ 3 & 0 & 2 \end{bmatrix}, \quad \hat{\beta}(2) = \begin{bmatrix} 0 & 3 & 0 \\ 0 & 0 & 0.1 \\ 0.1 & 0 & 0 \end{bmatrix}, \quad \hat{\beta}(3) = \begin{bmatrix} 0 & 0 & 3 \\ 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \end{bmatrix}$$

Rather than requiring $x \in X$, which cannot be satisfied by existing algorithms, we specify that when agents assign themselves to a task, they receive benefit corresponding to what proportion of the task they completed (i.e. 50% when two agents complete the same task). This disincentivizes duplicated assignments, and means that in this SAP, the optimal policy is the joint assignment $x_k = (1, 2, 3)$ for all $k$, yielding 60 reward over 10 time steps. However, the greedy optimal policy for agent 1 is to continually assign itself to task 2, securing 3 reward for itself each time step but causing the system as a whole to receive far less benefit by driving the state to $s = 2$.

### *6.1.2 Benchmark algorithms*

One natural comparison is to the greedy approach used for pretraining in which $x_k = \alpha(\hat{\beta}(s_k))\ \forall k$. We denote this algorithm as $\boldsymbol{\alpha(\hat{\beta})}$ for clarity.

In addition to REDA and $\alpha(\hat{\beta})$, we also apply several state of the art MARL algorithms to problems in this section. Independent $Q$-learning (**IQL**, i.e. [60]) consists of independent agents acting with and updating a shared deep $Q$-network. Similarly, independent proximal policy optimization (**IPPO**, i.e. [67]) consists of independent agents acting with and updating a shared PPO policy. These algorithms are state-of-the-art across many challenging MARL tasks [45], yet use independent rewards. This means that we expect agents trained under this paradigm to exhibit selfish behavior.

Counterfactual Multi-Agent Policy Gradients (**COMA** [23]) is a strategy which uses fully cooperative rewards and a centralized critic to evaluate counterfactual situations and assess the impact of individual agent actions on the reward.

### *6.1.3 Results*

Performance of various algorithms in this environment is shown in Figure 6.1. Qualitative analysis of the results yields significant insights into the trade-offs of each method. While REDA immediately drives the group to optimal joint policy, both IQL and IPPO reliably converge to the greedy joint policy $x_k = (2, 3, 1)\ \forall k$. This corresponds to the dictator acting selfishly and driving the system to state 2 at the expense of other agents.

COMA avoids this greedy behavior, but still has difficulty disentangling the effect of agents on the joint reward. In experiments, it converges to the joint policy $x_k = (1, 3, 1)\ \forall k$, which suggests that it cannot determine whether agent 1 or 3 should complete task 1.

This experiment demonstrates that REDA can accurately learn the values of assignments, yet force each agent to select an assignment that leads to an outcome beneficial for the entire group, rather than a selfish one.

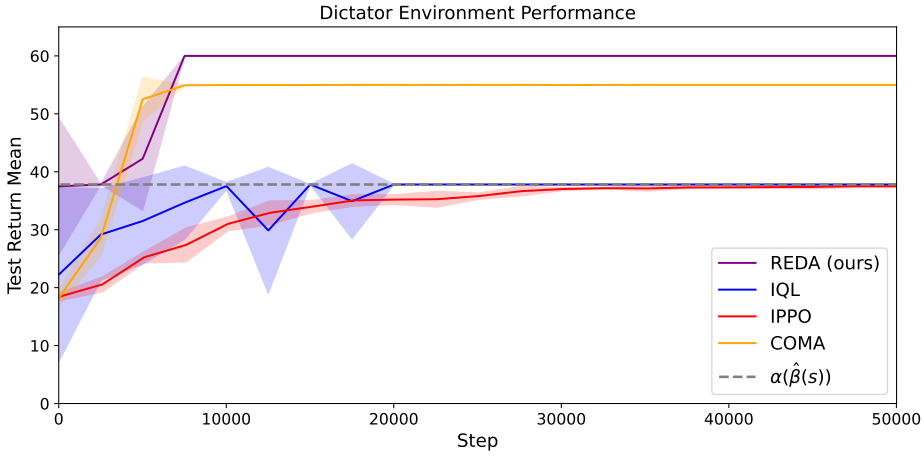



Figure 6.1: Performance over 5 runs of various algorithms in dictator environment, shown with standard deviation shaded. Note that after $\epsilon$ decays to 0 at $t = 10{,}000$, performance for REDA instantly approaches the theoretical maximum, while other algorithms remain significantly below the maximum.

## *6.2 Environment Setup for REDA Satellite Constellation Experiments*

We will now describe how the satellite constellation environment is specified to allow us to apply REDA to several large-scale constellations.

### *6.2.1 System state*

At a baseline, the full state of the system $s_k$ contains the following information:

- Static benefits of task assignments $[\beta_k]_{ij} \in \mathbb{R}$ for all $i, j$ (Equation 4.1), as well as benefits from the following $L-1$ time steps $[\beta_{k+1}]_{ij}, \cdots, [\beta_{k+L-1}]_{ij}$.
- Previous satellite assignments $x_{k-1} \in X$.

In simple environments such as the one seen in Section 4.2, this provides enough information to fully specify the state dependent benefits $\hat{\beta}(s)$. In more complicated environments, further elements are added to the state.

### *6.2.2 Observation space*

In large constellations, it becomes prohibitively expensive and unrealistic to provide individual satellites with all of this information when making assignment decisions. Instead, we condition agent policies on a partial observation of the system state. One key property of satellite assignment problems (and indeed assignment problems in many settings where tasks are associated with some spatial dimension) is that satellites can only see a small number of regions on Earth at a given time, and thus that the vast majority of $i \to j$ assignments have zero benefit and are in practice identical.

Based on this, we limit satellite observations to just the $M$ tasks with the highest benefits to satellite $i$:

$$\mathcal{M}_k^i := \{j \ : \ \sum_{t=0}^{L-1} [\beta_{k+t}]_{ij} \text{ is one of the } M \text{ largest of any } j = 1, ..., m\}.$$

$M$ should be selected such that it is larger than the maximum number of tasks that could provide non-zero benefits for a single satellite—for the experiments in this thesis, $M = 10$. This provides satellites with information about the raw benefit they can expect from completing a task.

Satellites also need information about what tasks might be highly desired by other agents. Thus, we define the set of the $N$ satellites with the highest benefit for one of the tasks in $\mathcal{M}_k^i$ as the *neighbors* of satellite $i$:

$$\mathcal{N}_k^i := \{l \ : \ \max_{j \in \mathcal{M}_k^i} \sum_{t=0}^{L-1} [\beta_{k+t}]_{lj} \text{ is one of the } N \text{ largest of any } l = 1, ..., n\}.$$

We use $N = 10$ in our experiments. Because in the satellite setting the non-state-dependent component of benefits, $\beta$, is defined largely by proximity, neighboring satellites $l \in \mathcal{N}_k^i$ are likely to be physically near agent $i$ as well. This makes it realistic that satellites could obtain this information as part of their observation (i.e. through inter-satellite laser links).

Thus, the observation space for an agent $i$ in our experiments contains the following information:

- Static benefits for the best tasks for the agent and its neighbors, $[\beta_k]_{lj}, \cdots, [\beta_{k+L-1}]_{lj}$ for all $j \in \mathcal{M}_k^i$, $l \in \{\mathcal{N}_k^i \cap i\}$.

- The previous assignments for the agent and its neighbors, $x_{k-1}^l$ for all $l \in \{\mathcal{N}_k^i \cap i\}$.

### *6.2.3 "Action" space*

Given that agents only observe information related to the top $M$ benefit tasks $j \in \mathcal{M}_k^i$, we also define our agent $Q$-functions only in terms of potential assignments to the top $M$ tasks.

Based on our earlier observation that the majority of tasks are outside the FOV of satellites and thus yield zero benefit, all other assignments are interchangeable. Thus, taking any action $j \notin \mathcal{M}_k^i$ is evaluated using a single evaluation of the agent $Q$-function, $Q_i^\pi(o_k^i, M+1)$.

As a concrete example of this simplication, suppose $m = 4$, $M = 2$, $\mathcal{M}_k^i = \{3, 1\}$, where $[\beta_k]_{i3} > [\beta_k]_{i1} > [\beta_k]_{i2} = [\beta_k]_{i4} = 0$. Then, $\mathbf{Q}_k^\pi$ would be defined as follows:

$$\mathbf{Q}_k^\pi = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ Q_i^\pi(o_k^i, 2) & Q_i^\pi(o_k^i, 3) & Q_i^\pi(o_k^i, 1) & Q_i^\pi(o_k^i, 3) \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

where task 3 is the most valuable and is thus evaluated using $Q_i^\pi(o_k^i, 1)$, whereas tasks 2 and 4 are not in $\mathcal{M}_k^i$ and are thus evaluated with $Q_i^\pi(o_k^i, 3)$ (because $M + 1 = 3$).

These modifications to the observation and action space allow agents to work with a much smaller observation and action space than would typically be required ($N << n, M << m$) and thus allows us to apply REDA to environments that are far larger than those often found in the MARL literature.

## *6.3 Revisiting a Simple Earth Coverage Experiment*

Using this formulation of the observation and action spaces, we can test REDA's performance in a realistically-sized satellite constellation experiment similar to the one used in Section 4.2.

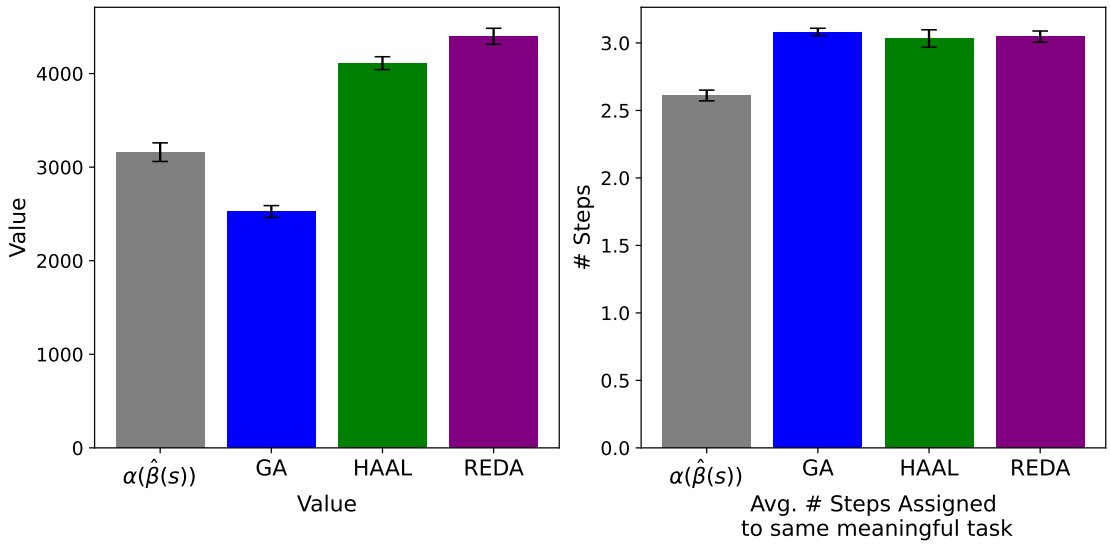



Figure 6.2: Performance of REDA in simple Earth coverage experiment. We can see that REDA outperforms HAAL to a statistically significant degree, while considering future information to make assignments that are just as consistent as HAAL.

We apply REDA to the midsize LEO internet constellation with randomly placed tasks ($n = 324$, $m = 450$). The only difference between this experiment and the one in Section 4.2 is that $\beta_j^{\max} = 1 \ \forall j$, and that handover penalties are only applied when switching to a *meaningful task* ($[\beta_k]_{ij} > 0$):

$$[\eta(s_k)]_{ij} = \begin{cases} \lambda & \text{if } x_{ij}^{s_k} = 0 \text{ and } [\beta_k]_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}, \quad \forall k.$$

Intuitively, this means that if a satellite assigns itself to a task that it cannot see, it remains idle and is not penalized for changing tasks. This behavior is more realistic and allows us to make the assumption that all tasks $j \notin \mathcal{M}_k^i$ are interchangeable and use the smaller action space as outlined in Section 6.2.

Results of these experiments are shown in Figure 6.2. Surprisingly, even in an environment for which HAAL is well-suited, REDA performs identically, if not better than, classical

approaches. This indicates that the REDA training process and our assumptions on the action and observation space still allow REDA to learn near-optimal policies.

## *6.4 Earth Coverage with Power Management Experiment*

As discussed in Chapter 5, REDA was designed to tackle even more complex problems for which HAAL is poorly suited. Thus, to investigate the relative performance of REDA and HAAL in an environment which requires long-term planning, we create an environment in which satellites have to manage their power state over the course of an orbit.

### *6.4.1 Modeling satellite power state*

Specifically, in addition to the standard elements of state, each satellite has a power state $p_k^i \in [0, 1]$ which starts at $p_1^i = 1$. When a meaningful task is selected, the satellite's charge decreases by 0.2, while when a task with zero benefit is selected (i.e. when agents are assigned to a task evaluated with $Q_i^\pi(o_k^i, M+1)$) charge increases by 0.1. When a satellite runs out of power (i.e. $p^i \leq 0$) for the first time, it receives no further benefit for any tasks completed in the episode, and the battery does not charge. This results in the following state-dependent benefits:

$$[\hat{\beta}(s_k)]_{ij} := \begin{cases} [\beta_k]_{ij} & \text{if } [x_{k-1}]_{ij} = 1 \text{ and } p_k^i > 0 \\ [\beta_k]_{ij} - \lambda & \text{if } [x_{k-1}]_{ij} = 0 \text{ and } [\beta_k]_{ij} > 0 \text{ and } p_k^i > 0 \\ 0 & \text{otherwise (i.e. } p_k^i \leq 0) \end{cases}$$

Accordingly, satellites also observe power states of each neighboring satellite, $p_k^l$ for all $l \in \{\mathcal{N}_k^i \cap i\}$.

### *6.4.2 Benchmark algorithms*

We again apply several standard MARL algorithms to this problem. To ensure a fair comparison, IQL was also provided with pretraining from the greedy policy $\alpha(\hat{\beta})$, while IPPO

was trained from scratch after behavior cloning on the greedy policy was found not to be beneficial.[1]

COMA is not tested because COMA requires training a centralized critic that can learn the value of all possible joint assignments, making it impractical for problems with a large joint state space.

Finally, because we expect classical approaches to struggle to manage their power states over the long term, we implement $\boldsymbol{\alpha(\hat{\beta})}$ **with power constraint**, which assigns greedily at each time step $x_k = \alpha(\hat{\beta}(s_k))$ under the constraint that no assignment is made which causes a satellite to run out of power.

### *6.4.3 Results*

Figure 6.3 shows the results of our algorithm in this environment. We find that REDA has low variance and consistently outperforms all other tested algorithms, beating IQL by $\sim 20\%$.

In Figure 6.4, we can see qualitatively why REDA performs so well on this task. Standard MARL approaches generate assignments with a significant amount of conflict, while classical approaches like HAAL fail to manage their power state over the course of the episode. REDA succeeds in all three areas; it entirely eliminates conflicting assignments, and minimizes changes in assignment while still successfully managing satellite power states over the entire episode.

## *6.5 Robustness of REDA to Constellation Size*

RL algorithms often exhibit poor performance when applied to problems outside of the distribution of their training data. However, we find that REDA exhibits strong performance across a range of distributions, even those outside of its training data.

Specifically, Figure 6.5 shows that if we take the network trained using REDA on an

[1] See [30] for further details on hyperparameter selection and network architectures.

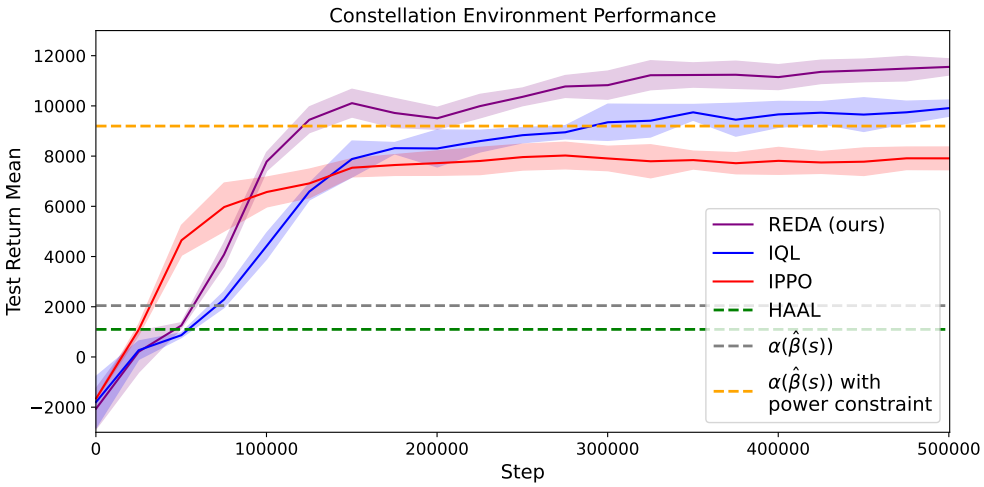



Figure 6.3: Performance of REDA in Earth coverage experiment with power management. $\epsilon$ is decayed to zero over 300,000 environment steps. We can see that after just 100,000 environment steps REDA outperforms all classical approaches, and consistently outperforms all other tested MARL methods.

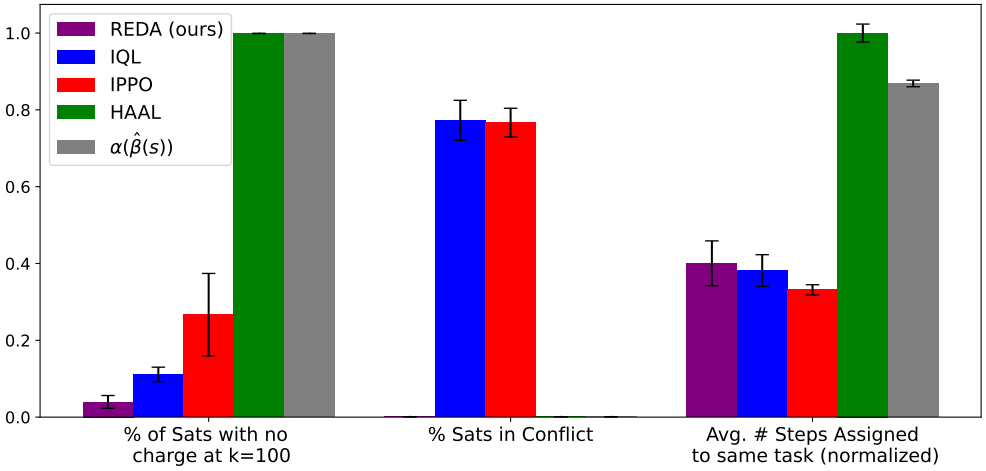



Figure 6.4: Performance of tested algorithms in power management environment across various secondary metrics. We can see that REDA outperforms IQL and IPPO across the board, while avoiding having satellites run out of power as when using classical methods like HAAL.

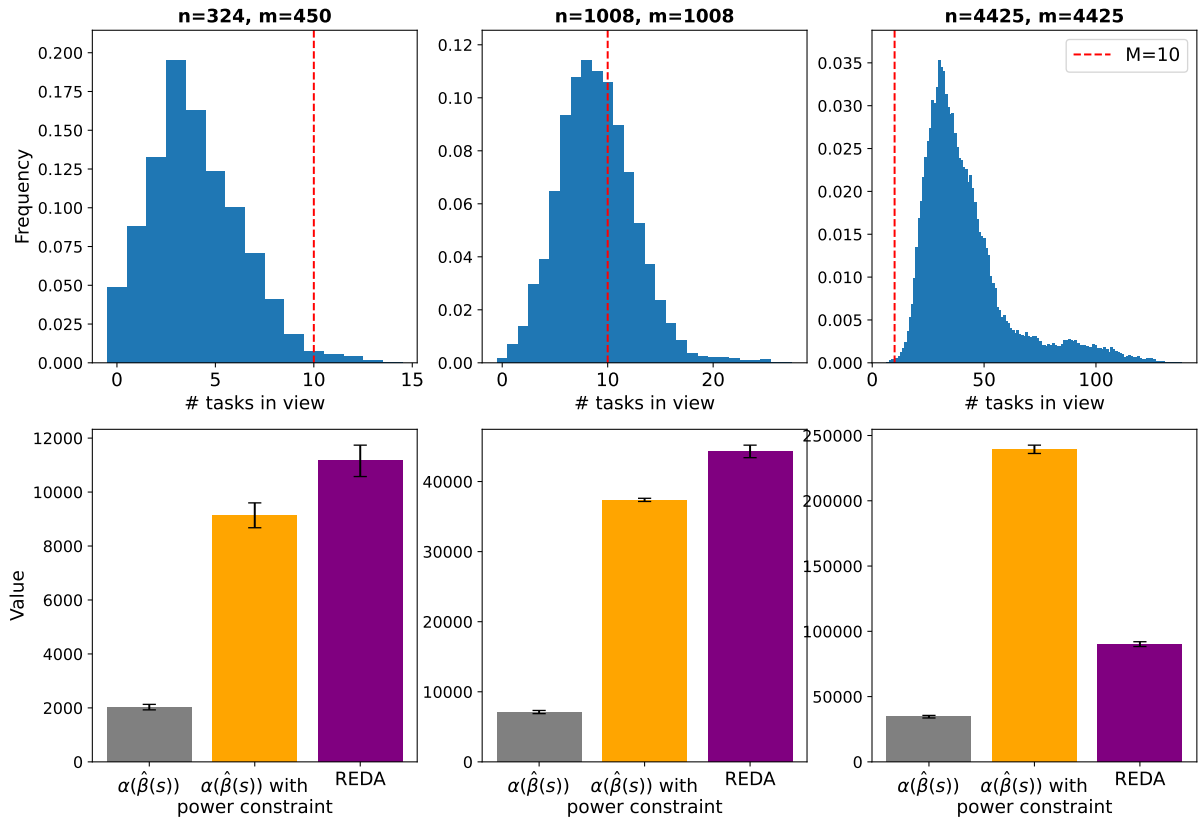



Figure 6.5: Performance of a REDA policy, trained on a 324 agent 450 task constellation, and applied to constellations of increasing size. When the average number of visible tasks is still below $M = 10$, performance of REDA is still better than all baselines, even in a constellation it has not been trained on. However, when the number of tasks in view for a satellite is far larger than $M$, performance is significantly degraded.

instance of the power management problem in Section 6.4 (trained using 324 agents and 450 tasks) and apply it directly to a much larger constellation with 1008 agents and 1008 tasks with no finetuning or other modifications, we find that HAAL still outperforms all other algorithms.

However, when applied to an even larger constellation, the trained REDA policy begins to degrade. Performance when applied to a 4225 satellite constellation with the nominal configuration of Starlink [20][2] is significantly worse than the greedy baseline with a power constraint.

Figure 6.5 also provides an explanation for this behavior. In the constellation with 1008 agents and tasks, many satellites still have less than $M = 10$ tasks in view at any given time. This means that agents can still evaluate most of the available tasks using their $(M + 1)$-dimensional action space, and that most of the tasks agents expect to correspond to charging indeed result in charging.

However, as the constellation continues to grow, the number of tasks in view also grows. Eventually, as in the Starlink case, the average number of tasks in view greatly exceeds $M$, causing several difficulties. First, in this case agents no longer have an action which can reliably evaluate all meaningful assignments available to them, limited the flexibility of the algorithms. Furthermore, in some cases action $M+1$ will result in performing a task which is in view and has nonzero benefit, thereby unexpectedly *reducing* charge. Qualitatively, we see that REDA agents trained on a smaller constellation where action $M + 1$ reliably increases the power state often run out of power in this new, out-of-distribution environment.

Despite these challenges, the performance of REDA in the constellation with $n = m = 1008$ is a promising result. This implies *that as long as the original agents are trained using action and neighbor space sizes $M$ and $N$ that are appropriate, REDA is highly robust to the number of agents in the environment.* Intuitively, REDA learns a policy which can effectively interact with neighboring agents to allocate tasks efficiently—these interactions don't change

[2]Orbital altitudes of $550, 510, 530, 675$, and 725 km respectively are used instead of the figures provided in the paper.

between a constellation with 300 or 30,000 satellites.

This promises to allow REDA to adapt to minor changes in constellation parameters without a need for retraining, and given that training on a constellation of 1,000 agents was completed on a single laptop, implies that REDA can be applied to even the largest constellations with more computational resources.

# Chapter 7

# CONCLUSION

In this chapter, we summarize the contributions of this thesis, and outline a vision for the application of this work to important real-world challenges.

## *7.1 Further Applications*

### *7.1.1 Further satellite constellation applications*

The success of REDA in complex satellite constellation tasking environments suggests that it can be used to create an entirely new paradigm of task allocation in this setting. Rather than the heuristic or naive approaches commonly used in current-day satellite constellation applications, REDA offers a principled approach to task allocation which nonetheless can handle the large variety of non-convex constraints which arise in practical settings.

An overarching vision for these next generation systems is depicted in Figure 7.1. By allowing algorithms to integrate the vast quantities of available data on weather and historical customer behavior, systems can develop much more efficient policies with respect to behavior of the real system and become more robust to changing weather conditions, and learn from experience in real time to meet changing customer demands.

Furthermore, much more complexity can be integrated into the satellite constellation simulations that are used in tasking problems. For example, satellite orientations were not modeled in the experiments in this thesis – instead, a constant penalty was applied for changing tasks, regardless of the difference in their required relative orientations. In future work, these differences could be explicitly modeled and factored into the calculation of the handover matrix.

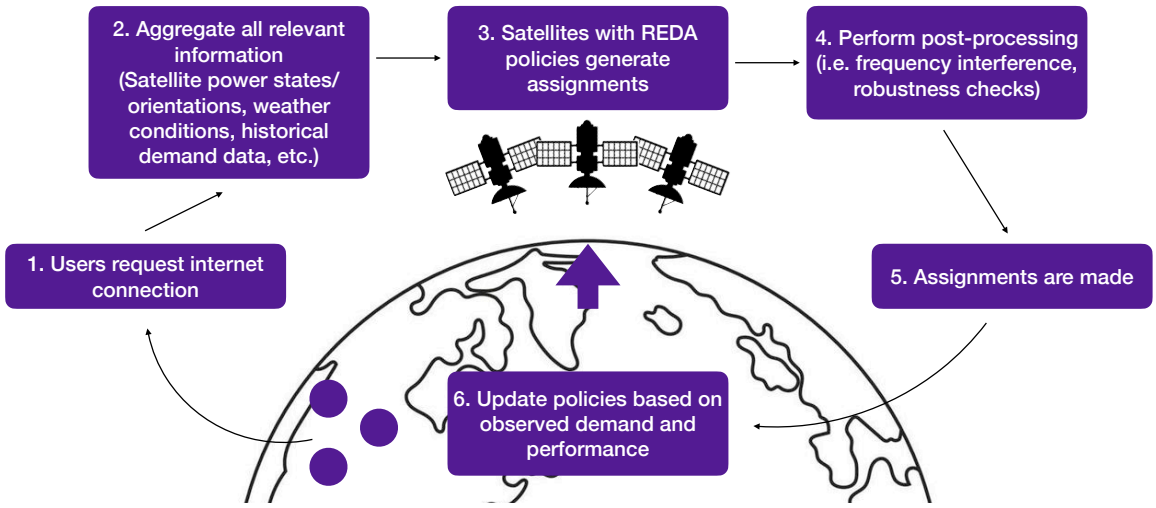



Figure 7.1: A visualization of an end-to-end satellite tasking pipeline. User internet requests are mapped directly to assignments based on a wide variety of data which can be collected during real-time operation.

### *7.1.2 Transportation network applications*

There is also a natural application of the SAP to distributed transportation networks like Uber and Lyft, in which drivers need to be matched to users requesting rides [1, 49]. Here, assignments are state dependent because their value depends on the current location of the driver and the rider.

With the success of REDA-like algorithms in real-world transportation networks serving as a proof of concept [4, 61], REDA itself could be applied in the transporation domain, leveraging its neural network representations, greedy pretraining, and novel exploration strategy to learn even more efficient and expressive policies in these settings.

### *7.1.3 Power grid applications*

Numerous power grid applications take the form of assignment problems, where power generation sources must be matched with users of that power. For example, in [38] the authors

use a linear programming relaxation to solve a constrained assignment problem in which they match substations with groups of customers, attempting to minimize the transmission distance and utilization of a group of substations.

A critical subproblem in the power grid setting is the *unit commitment problem*, in which generators need to be discretely turned on and off in order to match changing demand while minimizing startup and shutdown costs [41]. One could imagine a straightforward application of the SAP framework to this setting. Several RL algorithms have also seen success when applied to this domain [2, 18], but currently use centralized RL approaches rather than a decentralized approach, limiting their scaling potential.

One challenge with applying the SAP framework to this domain is that in addition to discrete on-off decisions, in many applications a choice must also be made about the *amount* of power that should be provided by a given power generator. However, one could imagine several ways of formulating the problem to account for this, including 1) having separate tasks corresponding to discrete levels of power, 2) applying SAP approaches to applications where power output must be constant (i.e. certain battery systems or natural gas generators), or 3) formulating this as a two-phase optimization problem in which a SAP method makes the on-off decisions, and an optimal solver is applied to the far easier, non-integer optimization problem.

## *7.2 Future Work*

Several areas of future work stand out. First, these algorithms should be applied to higher fidelity problems to showcase the power of the method and make progress on challenging applied problems across not just satellite constellations, but also transportation networks and power grids.

Another promising area of research is taking steps to apply HAAL and REDA to formulations of the SAP which are even more general. Despite showing in Sections 4.3 and 4.4 how the SAP can be applied to situations with a different set of valid assignments $X$, perhaps there are more natural ways to extend these algorithms to these settings.

Similarly, a major limitation of the sequential assignment problem is that the benefits can only depend on the state, not the other assignments. A more expressive version of the problem could be written as follows:

$$\max_{x_k \in X \ \forall k} \sum_{k=1}^{T} \sum_{i=1}^{n} \sum_{j=1}^{m} [\hat{\beta}(s_k, x_k)]_{ij} [x_k]_{ij}.$$

Here, the benefits depend on both the state of the system and the assignments of all agents in the system. This is a more accurate model in several scenarios, i.e. when satellite internet beams can cause interference with neighboring beams. It is possible that REDA in its current form can be applied to these scenarios, or perhaps new algorithmic advancements are necessary, such as formulating agent actions directly as bid amounts as in [8].

Finally, the theoretical aspects of both algorithms require more investigation. HAAL seems to have a deep but unexplored connection to model predictive control, which could yield stronger optimality bounds and algorithmic improvements. For REDA, while we provide intuition which motivates the connection with other centralized algorithms like DQN, we suspect there are deeper connections that may yield stronger convergence guarantees.

### *7.3 Summary of Contributions*

This thesis takes several steps towards increasing the efficiency of task allocation algorithms for satellite constellations and beyond.

First, we formulated the Sequential Assignment Problem (SAP), a NP-hard optimization problem which underlies much of the literature on satellite constellation task assignment, despite rarely being addressed directly. Next, we introduced a classical-optimization-based algorithm and its distributed version, Handover-Aware Assignment with Lookahead, to solve a useful class of SAPs with performance guarantees and polynomial-time scaling. Finally, we introduce a reinforcement-learning (RL) approach, RL-Enabled Distributed Assignment, which can learn experience to solve arbitrarily complex, stochastic instances of the SAP while drawing on data from a large variety of sources.

Each approach has clear strengths and weaknesses. In problem settings where handovers are generally undesirable and planning is mostly short term, HAAL provides a theoretically justified, computational efficient, and highly consistent approach to generating task assignments. However, in a more general setting with complicated transition dynamics or a requirement for long-term planning, REDA will tend to offer significantly better results.

Overall, both algorithms greatly improve performance over state of the art approaches in several complex problem settings, and promise to increase the robustness and efficiency of task allocation algorithms in the large-scale distributed systems of the twenty-first century.

# BIBLIOGRAPHY

[1] Niels Agatz, Alan Erera, Martin Savelsbergh, and Xing Wang. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2):295–303, 2012.

[2] Akshay Ajagekar and Fengqi You. Deep reinforcement learning based unit commitment scheduling under load and wind power uncertainty. *IEEE Transactions on Sustainable Energy*, 14:803–812, 2023.

[3] Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2024.

[4] Xabi Azagirre, Akshay Balwally, Guillaume Candeli, Nicholas Chamandy, Benjamin Han, Alona King, Hyungjun Lee, Martin Loncaric, Sébastien Martin, Vijay Narasiman, et al. A better match for drivers and riders: Reinforcement learning at Lyft. *INFORMS Journal on Applied Analytics*, 54(1):71–83, 2024.

[5] Dimitri Bertsekas. *Network Optimization: Continuous and Discrete Models*, volume 8. Athena Scientific, 1998.

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[7] Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. *Assignment Problems: Revised Reprint*. SIAM, 2012.

[8] Michael Chang, Sid Kaushik, S Matthew Weinberg, Tom Griffiths, and Sergey Levine. Decentralized reinforcement learning: Global decision-making via local economic transactions. In *International Conference on Machine Learning*, pages 1437–1447. PMLR, 2020.

[9] Doo-Hyun Cho, Jun-Hong Kim, Han-Lim Choi, and Jaemyung Ahn. Optimization-based scheduling method for agile earth-observing satellite constellation. *Journal of Aerospace Information Systems*, 15(11):611–626, 2018.

[10] Han-Lim Choi, Luc Brunet, and Jonathan P How. Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 25(4):912–926, 2009.

[11] Denise Chow. To cheaply go: How falling launch costs fueled a a thriving economy in orbit. *NBC News*, 2022.

[12] 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.

[13] Kyle Colton and Bryan Klofas. Supporting the flock: Building a ground station network for autonomy and reliability. In *Proceedings of the SmallSat Conference - Technical Session IX: Ground Systems*, pages 1–7. Planet Labs, 2016.

[14] Federal Communications Commission. Kuiper Systems LLC, Application for Authority to Deploy and Operate a Ka-band Non-Geostationary Satellite Orbit System. *FCC 20-102*, 2020.

[15] Kaixin Cui, Jiliang Song, Lei Zhang, Ying Tao, Wei Liu, and Dawei Shi. Event-triggered deep reinforcement learning for dynamic task scheduling in multisatellite resource allocation. *IEEE Transactions on Aerospace and Electronic Systems*, 59(4):3766–3777, 2023.

[16] Li Dalin, Wang Haijiao, Yang Zhen, Gu Yanfeng, and Shen Shi. An online distributed satellite cooperative observation scheduling algorithm based on multiagent deep reinforcement learning. *IEEE Geoscience and Remote Sensing Letters*, 18(11):1901–1905, 2021.

[17] Stéphane Dauzère-Pérès, Junwen Ding, Liji Shen, and Karim Tamssaouet. The flexible job shop scheduling problem: A review. *European Journal of Operational Research*, 314:409–432, 2023.

[18] Patrick de Mars and Aidan O'Sullivan. Applying reinforcement learning and tree search to the unit commitment problem. *Applied Energy*, 302:117519, 2021.

[19] Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

[20] Inigo del Portillo, Bruce G. Cameron, and Edward F. Crawley. A technical comparison of three low earth orbit satellite constellation systems to provide global broadband. *Acta Astronautica*, 159:123–135, 2019.

[21] Jens Eickhoff. *Onboard Computers, Onboard Software and Satellite Operations: an Introduction*. Springer Science & Business Media, 2011.

[22] Dennis Fischer, Tim A. Hartmann, Stefan Lendl, and Gerhard J. Woeginger. An investigation of the recoverable robust assignment problem. *CoRR*, 2020.

[23] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[24] Virginie Gabrel, Alain Moulet, Cécile Murat, and Vangelis Th. Paschos. A new single model and derived algorithms for the satellite shot planning problem using graph theory concepts. *Annals of Operations Research*, 69:115–134, 1997.

[25] Brian P Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 23(9):939–954, 2004.

[26] Vu Nguyen Ha, Eva Lagunas, Tedros Salih Abdu, Haythem Chaker, Symeon Chatzinotas, and Joel Grotz. Large-scale beam placement and resource allocation design for meo-constellation satcom. *arXiv preprint arXiv:2303.06372*, 2023.

[27] Ying He, Yuhang Wang, F. Richard Yu, Qiuzhen Lin, Jianqiang Li, and Victor C. M. Leung. Efficient resource allocation for multi-beam satellite-terrestrial vehicular networks: A multi-agent actor-critic method with attention mechanism. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):2727–2738, 2022.

[28] Adam Herrmann and Hanspeter Schaub. Reinforcement learning for the agile earth-observing satellite scheduling problem. *IEEE Transactions on Aerospace and Electronic Systems*, 59(5):5235–5247, 2023.

[29] Adam Herrmann, Mark A. Stephenson, and Hanspeter Schaub. Single-agent reinforcement learning for scalable earth-observing satellite constellation operations. *Journal of Spacecraft and Rockets*, 61(1):114–132, 2024.

[30] Joshua Holder, Natasha Jaques, and Mehran Mesbahi. Multi agent reinforcement learning for sequential satellite assignment problems. *Advances in Neural Information Processing Systems (Manuscript under review)*, 2024.

[31] Joshua Holder, Spencer Kraisler, and Mehran Mesbahi. Centralized and distributed strategies for handover-aware task allocation in satellite constellations. *Journal of Guidance Control and Dynamics (Manuscript under review)*, 2024.

[32] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[33] Andrew K Kennedy and Kerri L Cahoy. Performance analysis of algorithms for coordination of earth observation by cubesat constellations. *Journal of Aerospace Information Systems*, 14(8):451–471, 2017.

[34] Mohamed A Khamsi and William A Kirk. *An Introduction to Metric Spaces and Fixed Point Theory*. John Wiley & Sons, 2011.

[35] Panos Kouvelis and Gang Yu. *Robust Discrete Optimization and its Applications*, volume 14. Springer Science & Business Media, 2013.

[36] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[37] Zhiyuan Lin, Zuyao Ni, Linling Kuang, Chunxiao Jiang, and Zhen Huang. Dynamic beam pattern and bandwidth allocation based on multi-agent deep reinforcement learning for beam hopping satellite systems. *IEEE Transactions on Vehicular Technology*, 71(4):3917–3930, 2022.

[38] Bo Lyu, Shijian Li, Yanhua Li, Jie Fu, Andrew C. Trapp, Haiyong Xie, and Yong Liao. Scalable user assignment in power grids: a data driven approach. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPACIAL '16. Association for Computing Machinery, 2016.

[39] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[40] Philippe Monmousseau. Scheduling of a constellation of satellites: Creating a mixed-integer linear model. *Journal of Optimization Theory and Applications*, 191(2-3):846–873, 2021.

[41] Luis Montero, Antonio Bello, and Javier Reneses. A review on the unit commitment problem: Approaches, techniques, and resolution methods. *Energies*, 15(4):1296, 2022.

[42] Heiner Müller-Merbach. *Optimale Reihenfolgen*. Springer-Verlag, 1970.

[43] Nils Pachler, Edward F Crawley, and Bruce G Cameron. Beam-to-satellite scheduling for high throughput satellite constellations using particle swarm optimization. In *2022 IEEE Aerospace Conference (AERO)*, pages 1–9. IEEE, 2022.

[44] Nils Pachler de la Osa, Markus Guerster, Inigo del Portillo Barrios, Edward Crawley, and Bruce Cameron. Static beam placement and frequency plan algorithms for leo constellations. *International Journal of Satellite Communications and Networking*, 39(1):65–77, 2021.

[45] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. *arXiv preprint arXiv:2006.07869*, 2020.

[46] Shreya Parjan and Steve A. Chien. Decentralized observation allocation for a large-scale constellation. *Journal of Aerospace Information Systems*, 20(8):447–461, 2023.

[47] Wenduo Pei, Ruijie Zhu, Jingbo Wei, Yudong Zhang, Wenchao Zhang, Chao Xi, and Bo Yang. Stability and satisfaction index optimization for beam allocation in mega leo constellation. In *2023 Opto-Electronics and Communications Conference (OECC)*, pages 1–5. IEEE, 2023.

[48] Sean Phillips and Fernando Parra. A case study on auction-based task allocation algorithms in multi-satellite systems. In *AIAA Scitech Forum*, 2021.

[49] Zhiwei Tony Qin, Hongtu Zhu, and Jieping Ye. Reinforcement learning for ridesharing: An extended survey. *Transportation Research Part C: Emerging Technologies*, 144:103852, 2022.

[50] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

[51] J. Rawlings, D.Q. Mayne, and Moritz Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 01 2017.

[52] Tim Roughgarden. Algorithmic game theory. *Communications of the ACM*, 53(7):78–86, 2010.

[53] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[54] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[55] Satinder Singh, Tommi Jaakkola, Michael L Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38:287–308, 2000.

[56] Sara Spangelo, James Cutler, Kyle Gibson, and Amy Cohn. Optimization-based scheduling for the single-satellite, multi-ground station communication problem. *Computers and Operations Research*, 57:1–16, 2015.

[57] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[58] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

[59] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12, 1999.

[60] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017.

[61] Xiaocheng Tang, Zhiwei Qin, Fan Zhang, Zhaodong Wang, Zhe Xu, Yintai Ma, Hongtu Zhu, and Jieping Ye. A deep value-network based approach for multi-driver order dispatching. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1780–1790, 2019.

[62] Nobuaki Tomizawa. On some techniques useful for solution of transportation network problems. *Networks*, 1(2):173–194, 1971.

[63] Xinwei Wang, Guohua Wu, Lining Xing, and Witold Pedrycz. Agile earth observation satellite scheduling over 20 years: Formulations, methods, and future directions. *IEEE Systems Journal*, 15(3):3881–3892, 2020.

[64] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[65] Matthew Weinzierl, Prithwiraj Choudhury, Tarun Khanna, Alan MacCormack, and Brendan Rosseau. Your company needs a space strategy. Now. *Harvard Business Review*, 2022.

[66] Stephen Young. The meteoric rise in satellite numbers. *Union of Concerned Scientists*, 2022.

[67] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

[68] Michael M Zavlanos, Leonid Spesivtsev, and George J Pappas. A distributed auction algorithm for the assignment problem. In *2008 47th IEEE Conference on Decision and Control*, pages 1212–1217. IEEE, 2008.

# Appendix A

# PROOF OF OPTIMALITY OF HAAL

Proofs in this section are adapted from [31] but modified for the general state-dependent setting.

Recall that $|H| = \max_{s\in\mathcal{S},\ x\in\mathcal{X}} H(s,x) - \min_{s\in\mathcal{S},\ x\in\mathcal{X}} H(s,x)$, $\underline{B} = \min_{x\in\mathcal{X},\ k} B(\beta_k, x)$, and $c = \min\{\tilde{c} : \underline{B} \geq |H|\tilde{c}\}$.

First, we prove a useful lemma:

**Lemma A.0.1.** *Suppose $c > 0$ and $\eta(s)$ is vanishing for all $s \in \mathcal{S}$. Let $\boldsymbol{x}^* = (x_1^*, \ldots, x_T^*)$ be optimal assignments with respect to (1.3), with corresponding state $\mathbf{s}^* = (s_1^*, \cdots, s_T^*)$. Fix an arbitrary $s_k \in \mathcal{S}$ and set $x^\alpha := \alpha(\beta_k + \eta(s_k))$. Then for all $k$ we have,*

$$V(\beta_k, s_k, x^\alpha) \geq |H|c, \quad \textit{and} \quad V(\beta_k, s_k, x^\alpha) + |H| \geq V(\beta_k, s_k^*, x_k^*).$$

*Proof.* Note that when $x = x_{k-1}$, the definition of $\underline{B}$ and Equation 3.7 imply:

$$V(\beta_k, s_k, x) = V(\beta_k, \mathcal{T}(s_{k-1}, x_{k-1}), x) = B(\beta_k, x) + H(\mathcal{T}(s_{k-1}, x_{k-1}), x) \geq \underline{B}.$$

Thus, since $x^\alpha$ has been selected greedily with respect to value (Lemma 3.1.1) and per the definition of $c$, we have:

$$V(\beta_k, s_k, x^\alpha) \geq V(\beta_k, s_k, x_{k-1}) \geq |H|c, \quad \text{for all } k.$$

Next, we will show that at each time step $k$, $x^\alpha$ yields within $|H|$ value of the optimal assignment (Intuitively, if $\alpha(\beta_k + \eta_k(s_k))$ ever yielded an assignment that was sub-optimal by more than $|H|$, it could simply incur the maximum handover penalty of $|H|$ and switch to the better assignment.) First, rewrite the value gained at time step $k$ by the optimal assignment $x^*$ as:

$$V(\beta_k, s_k^*, x_k^*) = B(\beta_k, x_k^*) + H(s_k^*, x_k^*) = B(\beta_k, x_k^*) + H(s_k, x_k^*) + [H(s_k^*, x_k^*) - H(s_k, x_k^*)].$$

Bounding the difference between any two $H$ evaluations by $|H|$ per the definition,

$$V(\beta_k, s_k^*, x_k^*) \leq B(\beta_k, x_k^*) - H(s_k, x_k^*) + |H| = V(\beta_k, s_k, x_k^*) + |H|.$$

Lemma 3.1.1 now implies that

$$V(\beta_k, s_k, x^\alpha) + |H| \geq V(\beta_k, s_k^*, x_k^*).$$

□

Next, we show that Algorithm 1 admits an optimality bound under certain conditions.

**Lemma A.0.2** (Optimality Bound on Algorithm 1)**.** *Suppose $c > 0$. Let $x_0 = x_0^H = x_0^* \in \mathcal{X}$ be an arbitrary initial assignment. Let $\mathbf{x}^H = (x_1^H, \ldots, x_T^H)$, $\mathbf{s}^H = (s_1^H, \cdots, s_T^H)$ be the output assignments and corresponding states of Algorithm 1. Let $\mathbf{x}^* = (x_1^*, \ldots, x_n^*)$, $\mathbf{s}^* = (s_1^*, \cdots, s_T^*)$ be optimal assignments and states with respect to (1.3). Then,*

$$\mathcal{V}(\mathbf{x}^H) \geq \frac{c}{1+c}\mathcal{V}(\mathbf{x}^*).$$

*Proof.* Using lemma A.0.1 and $x_k^H = \alpha(\beta_k + \eta_k(s_k^H))$, we have:

$$V(\beta_k, s_k^H, x_k^H) \geq |H|c \quad \text{for all } k \tag{A.1a}$$

$$V(\beta_k, s_k^H, x_k^H) + |H| \geq V(\beta_k, s_k^*, x_k^*) \quad \text{for all } k \tag{A.1b}$$

Next, we use (A.1b) to find a lower bound on the ratio of the values of the assignments of Algorithm 1 to the optimal assignments:

$$\frac{V(\beta_k, s_k^H, x_k^H)}{V(\beta_k, s_k^*, x_k^*)} \geq \frac{V(\beta_k, s_k^H, x_k^H)}{V(\beta_k, s_k^H, x_k^H) + |H|} = \frac{V(\beta_k, s_k^H, x_k^H) + (|H| - |H|)}{V(\beta_k, s_k^H, x_k^H) + |H|} = 1 - \frac{|H|}{V(\beta_k, s_k^H, x_k^H) + |H|} \tag{A.2}$$

Plugging in (A.1a) into (A.2), we now conclude that,

$$\frac{V(\beta_k, s_k^H, x_k^H)}{V(\beta_k, s_k^*, x_k^*)} \geq 1 - \frac{|H|}{(1+c)|H|} \geq \frac{c}{1+c}, \quad \text{for all } k. \tag{A.3}$$

Hence, $\sum_{k=1}^T V(\beta_k, s_k^H, x_k^H) \geq \frac{c}{1+c} \sum_{k=1}^T V(\beta_k, s_k^*, x_k^*)$, completing the proof. □

Finally, we prove our main result. Our general strategy will be to prove that any time Algorithm 2 achieves less than $\frac{100c}{1+c}\%$ of the optimal value (the difference from $\frac{100c}{1+c}\%$ being defined as $\Delta$), it will earn back this extra value at some later time step, ensuring that by the end of $T$ time steps it has reached $\frac{100c}{1+c}\%$ of the optimal value.

**Theorem 3.5.1** (Optimality of Algorithm 2)**.** *Suppose $c > 0$ and $\eta(\cdot)$ is vanishing. Let $x_0 \in X$ be an arbitrary initial assignment. Let $\mathbf{x}^H = (x_1^H, \cdots, x_T^H)$, $\mathbf{s}^H = (s_1^H, \cdots, s_T^H)$ be the assignments and states returned by Algorithm 2 using the lookahead window $L$. Let $\mathbf{x}^* = (x_1^*, \cdots, x_T^*)$, $\mathbf{s}^* = (s_1^*, \cdots, s_T^*)$ be optimal assignments and states with respect to (1.3). Then, $\mathcal{V}(\mathbf{x}^H) \geq \frac{c}{1+c}\mathcal{V}(\mathbf{x}^*)$.*

*Proof.* First, we define some notation used throughout the proof. For $1 \leq k \leq T-L+1$ and $\mathbf{q} \in \mathcal{Q}_L$, the *assignment sequence* $\mathbf{x}^{\mathbf{q},k} = (x_1^{\mathbf{q},k}, \ldots, x_L^{\mathbf{q},k})$ *induced by* $q$ *at time* $k$ is defined as:

$$x_t^{\mathbf{q},k} := \begin{cases} \alpha(\sum_{i=a}^{b} \beta_{k+(i-1)} + \eta(s_t^{\mathbf{q},k})) & \text{if } \exists [a,b] \in \mathbf{q} : t = a \\ x_{t-1}^{\mathbf{q},k} & \text{otherwise} \end{cases} \tag{A.4}$$

where $x_0^{\mathbf{q},k} := x_{k-1}^H$, $s_0^{\mathbf{q},k} := s_{k-1}^H$, and $s_t^{\mathbf{q},k} := \mathcal{T}(s_{t-1}^{\mathbf{q},k}, x_{t-1}^{\mathbf{q},k})$.

Next, we define several time interval sequences $\mathbf{q} \in \mathcal{Q}_L$ of interest. Let $\mathbf{q}' = ([1,1],[2,2],\ldots,[L,L]) \in \mathcal{Q}_L$ be the sequence of length 1 time intervals. Let $\mathbf{q}_k^* = \operatorname{argmax}_{\mathbf{q} \in \mathcal{Q}_L} \sum_{t=1}^{L} V(\beta_{k+t-1}, s_t^{\mathbf{q},k}, x_t^{\mathbf{q},k})$ be the time interval sequence which induces assignments yielding the highest value over a lookahead window of length $L$, starting at time step $k$. Note that per the definition of Algorithm 2,

$$x_k^H = x_1^{\mathbf{q}_k^*,k}, \; s_k^H = s_1^{\mathbf{q}_k^*,k} \text{ for all } k \tag{A.5}$$

Thus,

$$\sum_{t=2}^{L} V(\beta_{k+t-1}, s_t^{\mathbf{q}_k^*,k}, x_t^{\mathbf{q}_k^*,k}) + V(\beta_k, s_k^H, x_k^H) \geq \sum_{t=1}^{L} V(\beta_{k+t-1}, s_t^{\mathbf{q}',k}, x_t^{\mathbf{q}',k}) \tag{A.6}$$

Set $v_k^* := \max(|H|c, V(\beta_k, s_k^*, x_k^*) - |H|)$, which implies that $v_k^*$ is the minimum value such that $v_k^* + |H| \geq V(\beta_k, s_k^*, x_k^*)$ and $v_k^* \geq |H|c$. Using the same reasoning as Lemma A.0.2, we can say that $\sum_{k=1}^{T} v_k^* \geq \frac{c}{1+c}\mathcal{V}(\mathbf{x}^*)$. It remains to show that $\mathcal{V}(\mathbf{x}^H) \geq \sum_{k=1}^{T} v_k^*$.

We can prove this via induction. First, note that when $k = 1$, we can rearrange Equation A.6 to obtain,

$$\sum_{t=2}^{L} V(\beta_t, s_t^{\mathbf{q}_1^*,1}, x_t^{\mathbf{q}_1^*,1}) \geq \sum_{t=1}^{L-1} V(\beta_t, s_t^{\mathbf{q}',1}, x_t^{\mathbf{q}',1}) + V(\beta_L, s_L^{\mathbf{q}',1}, x_L^{\mathbf{q}',1}) - V(\beta_1, s_1^H, x_1^H)$$

Furthermore, note that Lemma A.0.1 applies to any assignment generated on a time step interval of length one, which includes $x_t^{\mathbf{q}',k}$ for all $t = 1, \ldots, L$. Using this fact in conjunction with having $v_k^*$ as the *minimum* value which satisfies the conditions in Lemma A.0.1, we conclude that,

$$\sum_{t=2}^{L} V(\beta_t, s_t^{\mathbf{q}_1^*,1}, x_t^{\mathbf{q}_1^*,1}) \geq \sum_{t=1}^{L-1} v_t^* + \Delta_1 \tag{A.7}$$

where $\Delta_1 = v_L^* - V(\beta_1, s_1^H, x_1^H)$. Inequality A.7 is the base case of our inductive argument.

In the inductive step, we aim to establish the following relationship when $k \leq T - L$:

$$\sum_{t=2}^{L} V(\beta_{k+t-1}, s_t^{\mathbf{q}_k^*,k}, x_t^{\mathbf{q}_k^*,k}) \geq \sum_{t=1}^{L-1} v_t^* + \sum_{t=1}^{k} \Delta_t \implies \sum_{t=2}^{L} V(\beta_{k+t}, s_t^{\mathbf{q}_{k+1}^*,k+1}, x_t^{\mathbf{q}_{k+1}^*,k+1}) \geq \sum_{t=1}^{L-1} v_t^* + \sum_{t=1}^{k+1} \Delta_t \tag{A.8}$$

where $\Delta_i = v_{L+i-1}^* - V(\beta_i, s_i^H, x_i^H)$.

To prove this, consider the time interval sequence $\mathbf{q}_k^{*'} = \{[\max(1, a-1), b-1] : [a, b] \in \mathbf{q}_k^*,\ b \neq 1\} \cap \{[L, L]\} \in \mathcal{Q}$, which corresponds to $\mathbf{q}_k^*$ shifted forwards by one time interval, and with a time interval of length one added to the end of the sequence to cover the last time interval in $L$.

One could directly induce a new assignment sequence $\mathbf{x}^{\mathbf{q}_k^{*'},k+1}$ using $\mathbf{q}_k^{*'}$ at time $k+1$. Another way to generate an assignment sequence $\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_L)$ which has constant assignments over all time intervals of $\mathbf{q}_k^{*'}$ is to reuse the segment of $\mathbf{x}^{\mathbf{q}_k^*,k}$ which was not executed, $\hat{x}_{t-1} = x_t^{\mathbf{q}_k^*,k}, t = 2...L$, and let $\hat{x}_L = \alpha(\beta_{k+L} + \eta(s_{L+1}^{\mathbf{q}_k^*,k}))$, where $s_{L+1}^{\mathbf{q}_k^*,k} := \mathcal{T}(s_L^{\mathbf{q}_k^*,k}, x_L^{\mathbf{q}_k^*,k})$. Then, as assignments induced by $\mathbf{q}_k^{*'}$ at time $k+1$ are optimal under the condition that assignments

are constant over time intervals in $\mathbf{q}_k^{*'}$ (Lemma 3.1.1 and Equation A.4), it follows that

$$\sum_{t=1}^{L} V(\beta_{k+t}, s_t^{\mathbf{q}_k^{*'},k+1}, x_t^{\mathbf{q}_k^{*'},k+1}) \geq \sum_{t=1}^{L} V(\beta_{k+t}, \hat{s}_t, \hat{x}_t)$$
$$\geq \sum_{t=2}^{L} V(\beta_{k+t-1}, s_t^{\mathbf{q}_k^*,k}, x_t^{\mathbf{q}_k^*,k}) + V(\beta_{k+L}, s_{L+1}^{\mathbf{q}_k^*,k}, \alpha(\beta_{k+L} + \eta(s_{L+1}^{\mathbf{q}_k^*,k}))).$$

Then, Lemma A.0.1, Equation A.5, and the optimality of $\mathbf{q}_{k+1}^*$ when induced at time step $k+1$, lead to,

$$V(\beta_{k+1}, s_{k+1}^H, x_{k+1}^H) + \sum_{t=2}^{L} V(\beta_{k+t}, s_t^{\mathbf{q}_{k+1}^*,k+1}, x_t^{\mathbf{q}_{k+1}^*,k+1}) \geq \sum_{t=2}^{L} V(\beta_{k+t-1}, x_{t-1}^{\mathbf{q}_k^*,k}, x_t^{\mathbf{q}_k^*,k}) + v_{k+L}^*$$

Rearranging and plugging in $\Delta_{k+1}$ along with the inductive step (Equation A.8), it thus follows that,

$$\sum_{t=2}^{L} V(\beta_{k+t}, s_t^{\mathbf{q}_{k+1}^*,k+1}, x_t^{\mathbf{q}_{k+1}^*,k+1}) \geq \sum_{t=1}^{L-1} v_t^* + \sum_{t=1}^{k} \Delta_t + \Delta_{k+1} = \sum_{t=1}^{L-1} v_t^* + \sum_{t=1}^{k+1} \Delta_t$$

proving the inductive case. Note that this induction implies that at $k = T - L + 1$:

$$\sum_{t=2}^{L} V(\beta_{T-L+t}, s_t^{\mathbf{q}_{T-L+1}^*,T-L+1}, x_t^{\mathbf{q}_{T-L+1}^*,T-L+1}) \geq \sum_{t=1}^{L-1} v_t^* + \sum_{t=1}^{T-L+1} \Delta_t.$$

Adding $V(\beta_{T-L+1}, x_{T-L}^H, x_{T-L+1}^H)$ to both sides leads to

$$\sum_{t=1}^{L} V(\beta_{T-L+t}, s_t^{\mathbf{q}_{T-L+1}^*,T-L+1}, x_t^{\mathbf{q}_{T-L+1}^*,T-L+1}) \geq \sum_{t=1}^{L-1} v_t^* + \sum_{t=1}^{T-L} \Delta_t + v_T^*.$$

From this point forward, $L \geq T - k$, so the algorithm has full information and the assignments induced by $\mathbf{q}_{T-L+1}^*$ are the exactly the ones chosen by Algorithm 2 (formally, $(x_{T-L+1}^H, \ldots, x_T^H) = \mathbf{x}^{\mathbf{q}_{T-L+1}^*,T-L+1}$). Thus,

$$\sum_{k=T-L+1}^{T} V(\beta_k, s_k^H, x_k^H) \geq \sum_{t=1}^{L-1} v_t^* + \sum_{t=1}^{T-L} \Delta_t + v_T^*. \tag{A.9}$$

We now have enough information to bound $\sum_{k=1}^{T} V(\beta_k, s_k^H, x_k^H)$. Using the bound from Equation A.9 and the identity $V(\beta_i, s_i^H, x_i^H) = v_{L+i-1}^* - \Delta_i$, we find:

$$\sum_{k=1}^{T} V(\beta_k, s_k^H, x_k^H) = \sum_{k=1}^{T-L} V(\beta_k, s_k^H, x_k^H) + \sum_{k=T-L+1}^{T} V(\beta_k, s_k^H, x_k^H)$$
$$\geq \sum_{k=1}^{T-L} [v_{L+k-1}^* - \Delta_k] + \sum_{t=1}^{L-1} v_t^* + \sum_{t=1}^{T-L} \Delta_t + v_T^*.$$

Cancelling $\Delta_k$ terms and bringing all $v^*$ terms into one sum, we now conclude that,

$$\sum_{t=1}^{T} V(\beta_t, s_t^H, x_t^H) \geq \sum_{t=1}^{T} v_t^* \geq \frac{c}{1+c} \sum_{t=1}^{T} V(\beta_t, s_t^*, x_t^*) \quad \text{implying that} \quad \mathcal{V}(\mathbf{x}^H) \geq \frac{c}{1+c} \mathcal{V}(\mathbf{x}^*).$$

□

# Appendix B

# PROOFS FOR REDA

**Theorem 5.2.1** (Decomposition of $Q^{\pi^\alpha}$ into $Q_i^{\pi^\alpha}$)**.** *Let $\pi^\alpha : S \to X$ be a constant, deterministic joint policy. Define the Q-function for an individual agent with respect to this joint policy $\pi^\alpha$ as:*

$$Q_i^{\pi^\alpha}(s,j) := \mathbb{E}\left[\mathcal{R}_i(s_k,j) + \sum_{t=k+1}^{T} \gamma^{t-k}\mathcal{R}_i(s_t,x_t^i) \;\middle|\; s_k = s, s_{k+1} \sim \mathcal{T}(s_k,\pi^\alpha(s)); \pi^\alpha\right]$$

*Then, in the assignment problem setting, where $\mathcal{R}(s,x) = \sum_{i=1}^{n} \mathcal{R}_i(s,x^i)$,*

$$Q^{\pi^\alpha}(s_k,x) = \sum_{i=1}^{n}\sum_{j=1}^{m} Q_i^{\pi^\alpha}(s_k,x^i)x_{ij} \quad \textit{for} \;\; x = \pi^\alpha(s_k). \tag{5.1}$$

*Proof.* Recall the standard definition of $Q^{\pi^\alpha}(s,x)$:

$$Q^{\pi^\alpha}(s,x) := \mathbb{E}\left[\mathcal{R}(s_k,x_k) + \sum_{t=k+1}^{T} \gamma^{t-k}\mathcal{R}(s_t,x_t) \;\middle|\; s_k = s, x_k = x, s_{k+1} \sim \mathcal{T}(s_k,x_k); \pi^\alpha\right]$$

In the assignment problem setting where $\mathcal{R}(s,x) = \sum_{i=1}^{n} \mathcal{R}_i(s,x^i)$, we have:

$$Q^{\pi^\alpha}(s,x) = \mathbb{E}\left[\sum_{i=1}^{n}\mathcal{R}_i(s_k,x_k^i) + \sum_{t=k+1}^{T}\gamma^{t-k}\sum_{i=1}^{n}\mathcal{R}_i(s_t,x_t^i) \;\middle|\; s_k = s, x_k = x, s_{k+1} \sim \mathcal{T}(s_k,x_k); \pi^\alpha\right]$$

$$Q^{\pi^\alpha}(s,x) = \sum_{i=1}^{n}\mathbb{E}\left[\mathcal{R}_i(s_k,x_k^i) + \sum_{t=k+1}^{T}\gamma^{t-k}\mathcal{R}_i(s_t,x_t^i) \;\middle|\; s_k = s, x_k = x, s_{k+1} \sim \mathcal{T}(s_k,x_k); \pi^\alpha\right]$$

When $x = \pi^\alpha(s)$, then each term inside the summation is clearly equivalent to $Q_i^{\pi^\alpha}$, and we have:

$$Q^{\pi^\alpha}(s,x) = \sum_{i=1}^{n} Q_i^{\pi^\alpha}(s,x^i) \quad \text{for } x = \pi^\alpha(s).$$

Finally, noting that $j \neq x^i \implies x_{ij} = 0$ (an agent can only be assigned to a single task), we can say:

$$Q^{\pi^\alpha}(s,x) = \sum_{i=1}^{n}\sum_{j=1}^{m} Q_i^{\pi^\alpha}(s,j)x_{ij} \quad \text{for } x = \pi^\alpha(s).$$

and the proof is complete. □

**Lemma 5.4.1.** *Let $Q_i \in \mathcal{Q}$ be an arbitrary Q-function. Let $F : \mathcal{Q} \to \mathcal{Q}$ be the operator corresponding to the REDA target update in the tabular case, without target networks or the greedy guide policy:*

$$(FQ_i)(o_k^i, j) = \mathbb{E}^\pi\left[\mathcal{R}_i(s_k, j) + \gamma Q_i(o_{k+1}^i, x_{k+1}^i)\right]$$

*Assume each observation-assignment pair $(o^i, j)$ is visited infinitely often under a constant policy $\pi$. Then, if $Q_i^{n+1} \leftarrow FQ_i^n$, $\lim_{n\to\infty} Q_i^n = Q_i^\pi$.*

*Proof.* Starting from an arbitrary $Q_i, \tilde{Q}_i \in \mathcal{Q}$, and with $||Q|| = \sup_{o,j} Q(o,j)$, we have:

$$||FQ_i - F\tilde{Q}_i|| = \sup_{o,j} \gamma\left|\mathbb{E}^\pi\left[Q(o_{k+1}^i, x_{k+1}^i) - \tilde{Q}(o_{k+1}^i, x_{k+1}^i)\right]\right|.$$

Now, $o$ and $j$ only affect the reward in that they influence $o_{k+1}^i$ and $x_{k+1}$, so we can write:

$$||FQ_i - F\tilde{Q}_i|| \leq \sup_{o_{k+1}^i, x_{k+1}^i} \gamma|Q(o_{k+1}^i, x_{k+1}^i) - \tilde{Q}(o_{k+1}^i, x_{k+1}^i)| = \gamma||Q_i - \tilde{Q}_i||.$$

This proves the operator $F$ is a $\gamma$-contraction on $Q \in \mathcal{Q}$, and thus has a unique fixed point [34].

Moreover, its unique fixed point is $Q_i^\pi$. Starting from the operator $F$, we have:

$$(FQ_i^\pi)(o_k^i, j) = \mathbb{E}^\pi\left[\mathcal{R}_i(s_k, j) + \gamma Q_i^\pi(o_{k+1}^i, x_{k+1}^i)\right]$$

We abuse notation by writing $s_k \sim o_k^i$ to mean that $s_k$ is a randomly sampled state consistent with $o_k^i$. Then, substituting the definition of $Q_i^\pi$ and expanding $\mathbb{E}^\pi$,

$$(FQ_i^\pi)(o_k^i, j) = \mathbb{E}\left[\mathcal{R}_i(s_k, j) + \sum_{t=k+1}^{T} \gamma^{t-k}\mathcal{R}_i(s_t, x_t^i) \;\middle|\; s_k \sim o_k^i, s_{k+1} \sim \mathcal{T}(s_k, \pi(s_k)); \pi\right]$$

This is now the definition of $Q_i^\pi$, so $FQ_i^\pi = Q_i^\pi$ and per Banach fixed-point theorem [34], if $Q_i^{n+1} = FQ_i^n$, then $\lim_{n\to\infty} Q_i^{n+1} \to Q_i^\pi$. □