A Discretized Constraint-Aware Heuristic for Altimetry Data Acquisition in Satellites

Zhihao Jin

Thesis submitted to the University of Ottawa in partial fulfillment of
the requirements for the Master of Computer Science and
Concentration Applied Artificial Intelligence

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

uOttawa

ABSTRACT

As Earth observation missions continue to expand in both scale and complexity, optimization of satellite altimetry data acquisition has become a critical challenge. This thesis addresses continuing needs for scheduling hydrology-related measurements, such as those of oceans, lakes, and glaciers, efficiently while operating under strict constraints, including limited onboard memory, command execution limits, and overall satellite operability. To this end, we propose a constraint-aware heuristic that dynamically prioritizes and schedules altimetry targets, accounting for variations in spatial resolution and scientific priority.

The proposed methodology formulates mission planning as a multi-constraint optimization problem that incorporates satellite position and altitude, and to our knowledge, this is the first work to introduce such an approach. It adopts a knapsack-like formulation, accounting for variable data rates, target durations, and orbital geometry. The algorithm includes utility functions for memory management and spatially-aware target merging and supports dynamic mode downgrading to optimize data acquisition under resource limitations. A custom objective function is introduced to evaluate scheduling effectiveness, considering both data quality and target priority.

Experimental results, based on real and synthetic altimetry data, demonstrate the approach's scalability (handling up to 3,000 targets in minutes), high memory utilization efficiency, and superior performance compared to conventional optimization methods such as the Jaya algorithm. The algorithm delivers explainable, near-optimal target schedules with linear time complexity, making it a strong candidate for onboard autonomous planning.

ACKNOWLEDGMENTS

Time flies, and in the blink of an eye, two years of my graduate studies have passed. In this chapter, I would like to express my sincere gratitude for the journey I have experienced throughout this period of research and study.

First and foremost, I thank God for His guidance and blessings in my life and faith. Without His direction, I would not have been able to reach where I am today.

I am deeply grateful to my mother and my family for their unconditional support throughout my academic journey. Their constant encouragement and understanding made it possible for me to pursue my graduate studies abroad with confidence and peace of mind.

I would like to express my sincere gratitude to my thesis supervisor, Dr. Tom Cesari, for introducing me to the field of Artificial Intelligence and for his invaluable academic guidance, generous provision of research resources, and continuous support throughout my studies.

I would also like to express my appreciation to Jonathan Pergoli from the European Space Agency for providing valuable source data and insights, as well as for generously sharing his domain expertise, which greatly supported this work.

Finally, I extend my deepest gratitude to my beloved Xinrui Liu, whose unwavering support, understanding, and companionship carried me through countless nights of pressure and difficulty, making this journey both meaningful and memorable. I also thank my closest friends for their constant encouragement and thoughtful advice.

This thesis was prepared with limited assistance from artificial intelligence (AI) tools. The use of AI was restricted to auxiliary tasks related to the writing and technical preparation of the document, including reference formatting, citation management, paragraph polishing, and pseudo-code corrections. AI tools were employed to improve clarity, readability, and structural consistency of the manuscript, but they did not contribute to the generation of scientific ideas or methodological design. All research content, technical decisions, and conclusions presented in this thesis are the sole work of the author.

TABLE OF CONTENTS

Part I

Introduction and Literature Review

CHAPTER 1

INTRODUCTION

The growing use of satellite-based Earth Observation (EO) has changed the way scientists and engineers keep track of events happening on our planet. Among all the EO missions, altimetry ones are especially important—they measure things like sea level, river flow, glacier thickness, and how much water is stored on land. This information helps with climate studies, flood prediction, and water management. But the accuracy and detail that radar altimeters provide come with trade-offs: enormous amounts of data, limited onboard memory, and short communication windows with ground stations.

In modern EO missions, things are getting more complicated than ever. There are more observation requests, more data to handle, and smaller time windows to get everything done. Because of that, planning these missions has turned into a tricky balancing act. Traditional optimization algorithms often struggle here — they are powerful in theory, but the real world often brings too many moving parts and constraints, especially for Low Earth Orbit (LEO) satellites.

This thesis introduces a constraint-aware heuristic approach tailored to real-world altimetry operations. The method dynamically prioritizes targets based on scientific value, adjusts data acquisition modes to optimize memory usage, and merges spatially adjacent targets when necessary to reduce command overhead. The framework aims to maximize mission value while ensuring transparency, robustness, and operational feasibility, even in rapidly changing conditions, which we have already published in public. Furthermore in this thesis, targets are decomposed into smaller acquisition units, enabling more accurate memory modeling, flexible recording mode adjustment, and improved exploitation of spatial merging opportunities to reduce command overhead.

The approach was tested using both real and simulated data from Sentinel-6 missions and compared with common optimization methods like the Jaya algorithm. The results

suggest that this heuristic can handle large numbers of targets efficiently and still meet all satellite limits. Our algorithm is easy to understand, scales perfectly, and comes close to optimal results—making it a promising step toward more autonomous scheduling for future satellites.

CHAPTER 2

RELATED WORK

The literature surrounding satellite mission planning reflects a growing interest in handling the computational and operational challenges of Earth Observation (EO) scheduling. Mission planning for satellite altimetry must address various constraints, including onboard data storage limitations, downlink visibility windows, restricted command budgets, and the dynamic nature of user requests and orbital geometry.

2.1 Traditional optimization techniques

2.1.1 Linear Programming & Mixed Integer Programming

Traditional optimization techniques, such as **linear programming** and **mixed-integer programming (MIP)**, have been widely researched. However, these methods often fall short in space mission contexts due to the complexity of scheduling space tasks. The problem is non-convex and NP-hard, similar to classic problems like the knapsack problem [33]. In the context of satellite operations, additional complexities arise from position and time dependent constraints, resulting computationally infeasible for large-scale or real-time tasks.

In practice, linear and integer programming models have been widely applied to satellite scheduling problems in early day studies. In these methodologies, each observation targets request is typically represented as a binary variable showing whether it is selected or not. Additional constraints are then brought into to the mission requirements such as limited visibility windows for observations, onboard memory capacities for data storage, etc. By solving the optimization problem, those models and methodologies need to obtain an optimal schedule that maximizes data value while respecting mission constraints.

However, these formulations are only practical for small or medium-scale satellites missions. As the number of requests grows, the number of binary variables and constraints

increases dramatically which also results in the models growing exponentially. This computational explosion makes exact methods infeasible for large-scale operations, since it may require hours or even days to converge to a solution [49]. In real missions whose schedules must be generated quickly and frequently, such runtimes are unacceptable for all time.

The situation becomes even more challenging when scheduling agile satellites, which can rapidly rotate to acquire targets outside of their regular track. Modeling these systems requires additional constraints for satellite slew maneuvers, transition times, and attitude dynamics. Dealing with these factors dramatically increases the search space and increases the difficulty of the optimization, resulting the integer programming approaches nearly impossible to solve within operational time limits. As a result, while linear and integer programming approaches are theoretically powerful and provide valuable insights, they are often replaced in practice by heuristic or approximate methods which can produce near-optimal solutions within the strict time constraints of real-life satellite missions [22].

2.1.2 Branch and Bound (B&B)

One of the most well-known combinatorial optimization algorithms is **Branch and Bound (B&B)**. As illustrated in Figure 2.1, this algorithm operates by exploring the solution space through branching on decision variables and bounding each subproblem with linear programming relaxations by pruning unpromising branches [46].

In satellite scheduling, each observation target request can be handled as a binary decision variable, and B&B is used to evaluate whether including or excluding that request which could lead to a feasible and optimal schedule [6]. While this guarantees global optimality for small-scale cases, the exponential growth of the search tree makes it not practical for large-scale targets scheduling. [49, 22]. As a result, although B&B provides a foundational exact method in combinatorial optimization, its direct application in operational satellite missions is often replaced by heuristics, metaheuristics, or hybrid approaches that can provide near-

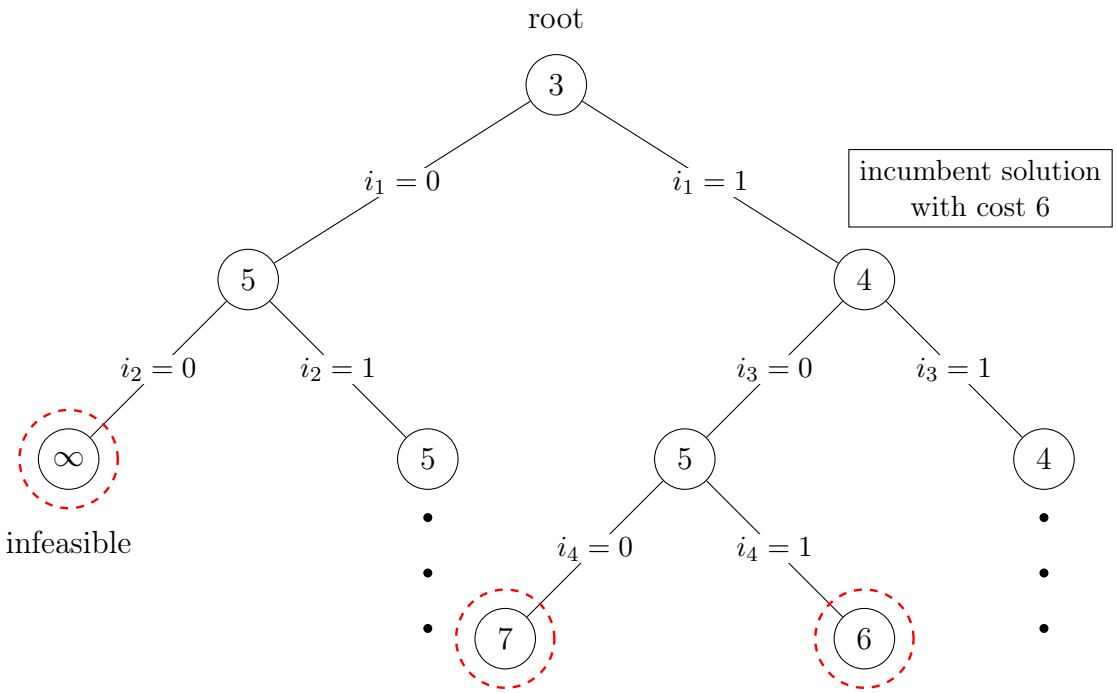


Figure 2.1: Illustration of the branch-and-bound search tree [93]

optimal solutions within practical time limits [88].

Researchers have also tried decomposition and relaxation methods, such as breaking the problem into smaller sub-problems or relaxing some constraints to make the model easier to solve. While these approaches can handle larger problems, they are still computationally expensive and often cannot guarantee solutions in real-time [6, 88]. Reviews of satellite scheduling consistently conclude that exact optimization is valuable for insight and benchmarking, but cannot scale to the size and speed required in real Earth observation missions [31].

2.1.3 Dynamic Programming

Dynamic Programming (DP) has long been recognized as one of the fundamental approaches for solving optimization problems that exhibit *optimal substructure* and *overlapping subproblems* properties [4]. By recursively decomposing a complex optimization task into smaller

subproblems, DP ensures that globally optimal solutions can be constructed from locally optimal decisions. DP considers the four most important factors in the algorithm, which are: **state, decision, transition function and base cases**. It has been extensively applied in areas such as resource allocation, scheduling, and routing optimization [**?**]. The fundamental idea of DP is to define a *value function* that represents the optimal cost (or reward) which is achievable from a given state. The general recursive form of DP can be expressed as:

$$V_t(s_t) = \begin{cases} \max\limits_{a_t \in A(s_t)} \left\{ R(s_t, a_t) + \gamma\, V_{t+1}(f(s_t, a_t)) \right\}, & \text{for maximization problems,} \\ \min\limits_{a_t \in A(s_t)} \left\{ C(s_t, a_t) + \gamma\, V_{t+1}(f(s_t, a_t)) \right\}, & \text{for minimization problems,} \end{cases} \tag{2.1}$$

where:

- t denotes the decision stage or timestep;
- s_t represents the system state at stage t;
- $a_t \in A(s_t)$ is the feasible action (or decision) set available in state s_t;
- $R(s_t, a_t)$ (or $C(s_t, a_t)$) is the immediate reward (or cost) obtained after applying action a_t;
- $f(s_t, a_t)$ defines the state transition function that maps the system to the next state s_{t+1};
- $\gamma \in [0, 1]$ is a discount factor that adjusts the contribution of future rewards;
- $V_t(s_t)$ is the value function, representing the optimal total reward (or minimum total cost) from state s_t onward.

The recursive relation in Equation (2.1) is commonly referred to as the *Bellman equation*. Solving a DP problem involves recursively evaluating $V_t(s_t)$ backward from the terminal stage T to the initial stage 0, which can ensure that globally optimal solutions are constructed from local optimal decisions.

However, when applied to large-scale real-world systems such as Earth observation satellite scheduling, DP often suffers from the *curse of dimensionality*, since its state space grows exponentially with the number of decision variables and constraints. This makes the direct application of traditional DP computationally impossible for problems that involve more than thousands of observation targets and interdependent time and resource constraints throughout the whole mission.

To avoid this issue, **bidirectional dynamic programming (BDP)** has been introduced as an effective strategy to reduce the state space explored during optimization [75]. Instead of performing a purely forward expansion from the starting state, BDP simultaneously explores the state space in both directions which is forward from the initial state and backward from the terminal state. The two searches meet at intermediate nodes, where the forward and backward subproblems are merged to construct complete feasible solutions. This dual exploration strategy significantly reduced unnecessary state evaluations and it cut off the number of transitions that need to be processed.

Righini and Salani (2009) applied the bidirectional dynamic programming (BDP) method to solve the *Orienteering Problem with Time Windows* (OPTW), which shares many similarities with satellite scheduling since both of them require selecting an optimal subset of targets while satisfying time and resource constraints. Their approach combined BDP with a *Decremental State Space Relaxation* (DSSR) technique. In this framework, the search begins with a relaxed state space that allows flexible or even infeasible paths, and gradually tightens the feasible region by enforcing visit constraints step by step. During each iteration, infeasible and dominated states are pruned, leading the algorithm to converge toward the

exact optimal solution while keeping the computation manageable. In addition, initialization heuristics were used to identify a set of *critical vertices* that guide the relaxation process and improve convergence speed. This combination of bidirectional search and progressive state relaxation shows that dynamic programming can be scaled to relatively large optimization problems when state expansion and convergence are carefully managed.

Building on these ideas, Wu and his team developed the *Adaptive-Directional Dynamic Programming (ADDP)* framework to further improve the efficiency of bidirectional approaches in 2020 [68]. The method introduces an adaptive, direction aware search mechanism which can dynamically adjusts how the search space is explored. In combination with a *Decremental State Space Relaxation (DSSR)* process, ADDP was designed specifically for the Agile Earth Observation Satellite Scheduling (AEOSS) problem. In this framework, the search space is divided according to the observation direction, and only transitions that are feasible within these directional limits are expanded. This selective or *directional pruning* greatly reduces unnecessary calculations, while still preserving the global optimal guaranteed by DP. Meanwhile, the DSSR component progressively tightens the state space by filtering out dominated or infeasible states in each iteration. Experimental results showed that the ADDP–DSSR framework could efficiently handle large-scale scheduling problems involving hundreds of observation targets in just a few minutes, achieving exact optimal solutions while overcoming the major limitations of classical dynamic programming [68].

This adaptive methodology highlights an important evolution in DP-based satellite scheduling algorithms, reducing the gap between exact optimization and computational possibility. The conceptual framework of the ADDP provides a valuable foundation for integrating DP principles and ideas with satellite data acquisition problems. However, the paper has one critical assumption which is on-board memory is large enough to observe all targets. Secondly, it lacks of targets flexibility operation. In other words, targets can't be considered to be merged with others since it may break the integrity of value function in dynamic

Table 2.1: Comparison of Adaptive-directional DP and Bidirectional DP [68]

Dataset Name	Adaptive-directional DP	Bidirectional DP	
	CPU time (s)	CPU time (s)	T_{dif}
500_38	2.17	3.18	-0.47
500_391	149.35	243.32	-0.63
500_60	64.66	49.73	0.23
500_117	25.62	440.84	-16.21
500_385	133.50	261.29	-0.96
Average	**75.06**	**199.67**	**-1.66**
600_71	12.18	12.65	-0.04
600_476	686.17	2244.74	-2.27
600_62	29.33	139.44	-3.75
600_144	141.15	208.94	-0.48
600_441	251.82	968.23	-2.85
Average	**224.13**	**714.80**	**-2.19**

T_{dif} denotes the normalized time difference compared to the Adaptive-directional DP baseline.

programming.

2.2 Heuristic algorithms

Compared with dynamic programming and other traditional optimization algorithms, heuristic methods stand out because of their key advantage: they can achieve near-optimal or sub-optimal solutions within acceptable computational time and resource consumption. In large-scale satellite scheduling problems, where the search space and time grow exponentially with the number of targets and constraints. Algorithms, such as mixed-integer programming or dynamic programming often become computationally impossible. By contrast, Heuristic algorithms provide a practical balance between solution quality and computational efficiency, which makes them particularly suitable for time sensitive or resource constrained mission planning scenarios.

2.2.1 Jaya algorithm

Among heuristic algorithms, the Jaya algorithm has gained attention for its simplicity and adaptability across a wide range of optimization domains [74]. It eliminates the need for algorithm-specific control parameters and relies instead on a straightforward update mechanism that moves candidate solutions toward the current best and away from the worst.

As shown in the Flowchart 2.2, parameter-free design makes it more appealing for general-purpose optimization tasks, especially when prior knowledge about the problem domain is limited. However, when applied to complex and dynamic systems such as satellite scheduling, several shortcomings become apparent.

First, the Jaya algorithm does not natively accommodate dynamically changing variables such as target merging, shifting observation priorities, or evolving resource constraints. In Earth observation missions, key parameters including onboard memory capacity, command limits, and ground station visibility can change rapidly as the satellite progresses along its orbit.

Without explicit mechanisms to adapt to these evolving conditions, Jaya can produce schedules that quickly become infeasible or inefficient. Secondly, despite being marketed as parameter-free, its performance is still influenced by its population size, iteration count, and stopping criteria those meta-parameters whose performance heavily depends on the meta-parameter tuning.

Inadequate or poor tuning of meta-parameters, can result to pre-mature convergence or excessive computation, especially for large-scale or highly constrained scheduling problems. These limitations make Jaya less suitable for real-time or safety-critical mission planning, where reliability, explainability and dynamically updating are essential and necessary.

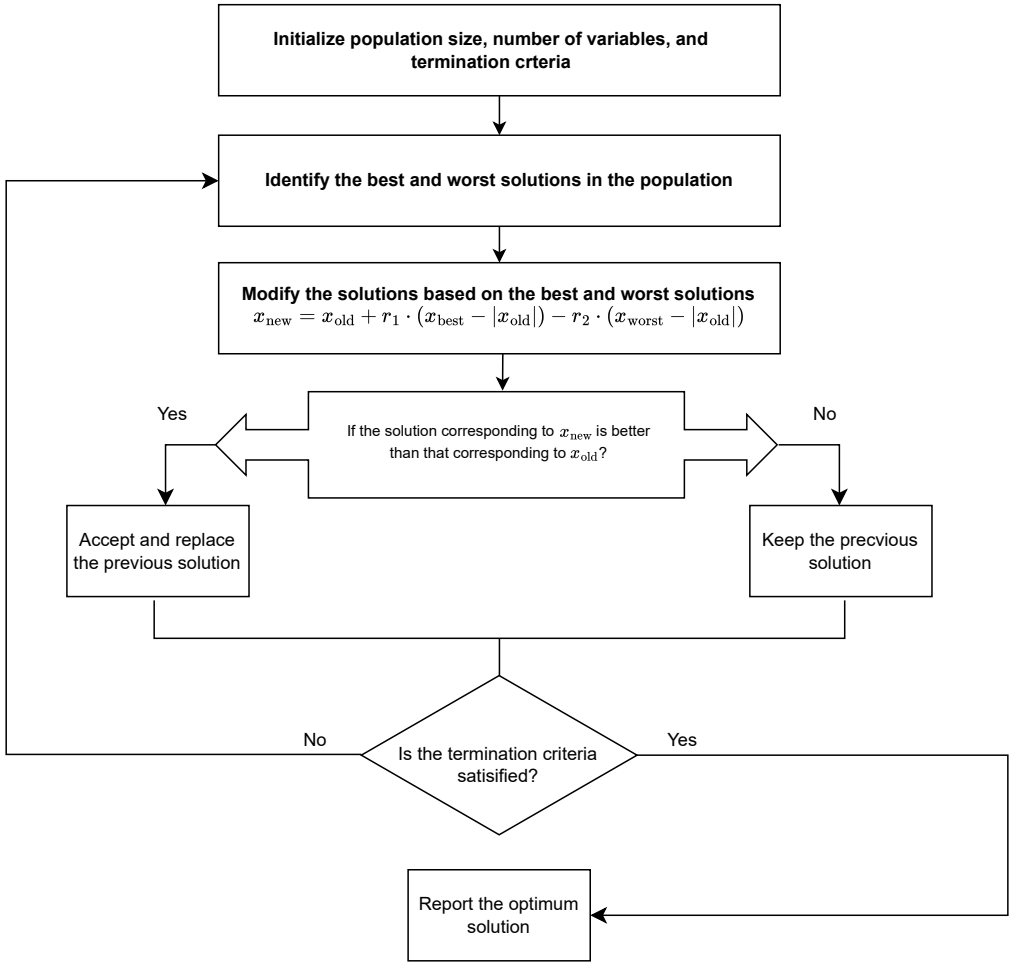


Figure 2.2: Flow chart of Jaya Algorithm[74]

2.2.2 Genetic Algorithm

Beyond Jaya, numerous heuristic and metaheuristic approaches have been explored in satellite scheduling research. Among the earliest are genetic algorithm (GA) and simulated annealing (SA), both of which have demonstrated success in handling large-scale and multi-objective scheduling problems. GA evolves a population of candidate schedules through selection, crossover, and mutation operators, gradually refining feasible task sequences [36].

It is particularly well suited for complex optimization problems because its search process mimics the biological evolution of species—maintaining a diverse population, inheriting

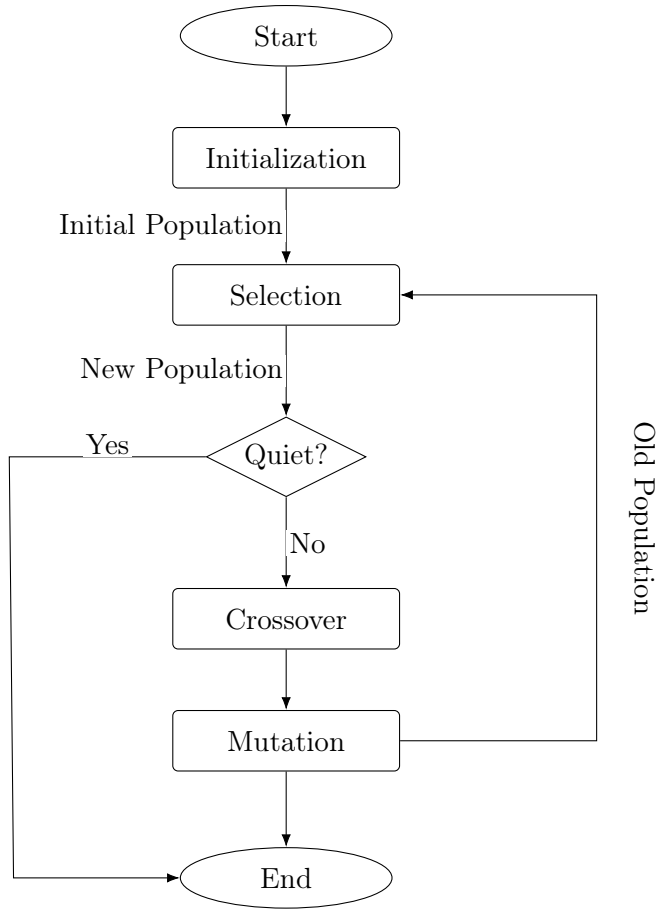


Figure 2.3: Flow chart of Genetic Algorithm [25]

useful traits, and discarding weak candidates over generations.

This evolutionary mechanism enables GA to efficiently explore large, discrete search spaces and balance exploration with exploitation.

In the context of Earth observation and agile satellite scheduling, GA has been widely adopted due to its flexibility in handling multiple constraints such as visibility windows, memory capacity, energy consumption, and attitude maneuver limitations.

Bianchessi et al. [9] developed a multi-satellite scheduling heuristic based on a hybrid GA framework, capable of coordinating multiple satellites under time-window and resource constraints. Similarly, Wolfe and Sorensen [95] applied GA to early Earth Observing System (EOS) scheduling problems, demonstrating its effectiveness compared with traditional exact

methods like integer programming.

Further improvements were made by Wu and Zhu [89], who combined GA with an ant colony optimization (ACO) component to accelerate convergence and improve global search capability for multi-satellite observation tasks.

Stefano and his team introduced a multi-objective GA to simultaneously maximize observation coverage and minimize transition energy, showing its adaptability to dynamic operational environments [83].

More recently, Wu et al. [**?**] proposed a hybrid GA that integrates local search and decomposition techniques, further enhancing the algorithm's robustness and computational efficiency for large-scale satellite scheduling.

Together, these studies confirm that GA provides a strong foundation for satellite mission planning; however, its performance often depends heavily on parameter tuning and problem-specific adaptations. Inappropriate choices of meta-parameters can lead to premature convergence to local optima in non-convex problems, motivating the need for more constraint-aware and adaptive approaches.

2.2.3 Tabu search

Tabu search has also been applied to satellite scheduling, using short-term memory structures to prevent cycling and encourage diversification [6]. By storing recently visited solutions, the algorithm avoids revisiting poor configurations and focuses exploration on promising regions of the search space.

While these methods are flexible and computationally tractable, their performance is often sensitive to parameter tuning and initialization strategies, which can limit their robustness across different mission scenarios.

Despite these advancements, a common challenge remains: most heuristic algorithms lack transparency and interpretability. In safety-critical satellite missions, operators must

understand why specific scheduling decisions are made and how trade-offs among competing objectives are handled. However, many heuristic methods operate as "black boxes", producing near-optimal results without offering insights into their internal decision-making processes. This opacity limits their operational acceptance, especially in missions that demand traceable and explainable scheduling outcomes.

2.3 Machine Learning & Deep Learning

The emergence of machine learning, especially deep learning (DL) and deep reinforcement learning (DRL) has begun to influence satellite scheduling research in both decision making and predictions. Recent studies considered scheduling as a sequential decision process with resource and temporal constraints, and use DRL to learn policies that maximize long term mission value under uncertainty. [40, 31].

2.3.1 Classical Machine Learning

Traditional statistical machine learning models, such as k-nearest neighbors (KNN), support vector machines (SVM), decision trees, and clustering algorithms like k-means, are primarily designed for prediction or classification tasks based on a large, labeled datasets. However, these assumptions do not stand for satellites scheduling problems. In our mission context, each scheduling instance corresponds to a unique optimization scenario rather than repetitive labeled tasks. Observing or acquiring targets do not have predefined "labels" which is used to indicate optimal choices. And more than that, the data are typically sparse, heterogeneous, and highly mission-specific [31, 52].

Moreover, satellite scheduling involves sequential decision-making under complex temporal, spatial, and resource constraints—features that classical statistical models cannot capture effectively. These models lack the ability to inference the dependencies across time or to optimize a global objective involving multiple conflicting constraints like attitude slews,

onboard memory, and visibility windows.

Consequently, classical ML algorithms are unable to generate feasible or near-optimal schedules in dynamic operational contexts [40]. Therefore, while traditional machine learning techniques may support limited predictive submodules (such as weather estimation or demand forecasting), they remain fundamentally unsuitable for end-to-end scheduling optimization. This limitation motivates the exploration of deep learning and reinforcement learning approaches that can directly model sequential, constraint-driven decision processes.

2.3.2 Online Learning

Before introducing reinforcement learning, it is essential to understand the concept of *online learning*, which forms one of its theoretical foundations. Traditional *batch learning* assumes to have a fixed and complete dataset before training and the model is optimized once and later then deployed to make predictions on unseen data. However, many real world problems, including satellite operations, operate in dynamic or nonstationary environments where new information continuously arrives over time. In such cases, the batch assumption breaks down, which motivates the development of *online learning*, where the learner updates its model or parameters sequentially after each observation [79, 63].

Core principle. In online learning, the learner interacts with the environment in rounds $t = 1, 2, \ldots, T$. At each round, the learner selects a decision h_t (e.g., a prediction or an action) based on the information available up to time t. Then, the environment reveals a loss function ℓ_t, and the learner incurs this loss. The goal is to minimize the cumulative loss compared to the best fixed decision in hindsight.

Parameters

Class $\mathcal{H}$ of predictors, **loss function** ℓ.

The algorithm outputs a default initial predictor $h_1 \in \mathcal{H}$.

For $t = 1, 2, \ldots$

1. The next example (x_t, y_t) is observed.
2. The loss $\ell(h_t(x_t), y_t)$ of the current predictor h_t is computed.
3. The online learner updates h_t, generating a new predictor $h_{t+1} \in \mathcal{H}$.

A characterizing feature of online learning is that the model update $h_t \to h_{t+1}$ is typically *local*. That is, it only involves the current predictor h_t and the current example (x_t, y_t).

Note that an online learner A generates a sequence $h_1, h_2, \cdots \in \mathcal{H}$ of predictors. We evaluate the performance of A through the notion of *sequential risk*:

$$\frac{1}{T}\sum_{t=1}^{T} \ell(h_t(x_t), y_t),$$

measuring, as a function of T, the average loss of the predictor sequence over the first T examples. The sequential risk is the online learning counterpart of the notion of statistical risk in statistical learning.

In what follows, we use the notation $\ell_t(h) = \ell(h(x_t), y_t)$ when the sequence $(x_1, y_1), (x_2, y_2), \ldots$ is understood from the context. This defines a sequence $\ell_1, \ell_2, \ldots$ of loss functions.

In keeping with the analogy between online and statistical learning, we also define the *regret* which is used to measure the quality of an online learning algorithm:

$$R_T = \sum_{t=1}^{T} \ell_t(h_t) - \min_{h^* \in \mathcal{H}} \sum_{t=1}^{T} \ell_t(h^*),$$

where R_T quantifies how much worse the online learner performs compared to the best

fixed decision h^*. An algorithm is considered effective if its regret grows sublinearly with time (i.e., $R_T = o(T)$), meaning that the average regret per step $\frac{R_T}{T}$ converges to zero as T increases.

Exploration in online decision making. In many practical scenarios, especially when feedback is limited, the learner must balance *exploration* (trying new actions to gather information) and *exploitation* (using the best-known option so far). This balance has been researched in the *multi-armed bandit* (MAB) problem, where a learner chooses among K actions ("arms") at each round, receiving stochastic rewards. The objective is to maximize cumulative reward, minimize cumulative regret—without knowing the reward distributions in advance [2, 48].

The ϵ-greedy algorithm. One of the simplest yet powerful strategies to address the exploration–exploitation trade-off is the ϵ-greedy algorithm. At each time step t, the agent selects:

$$a_t = \begin{cases} \text{a random action}, & \text{with probability } \epsilon, \\ \arg\max_a \hat{r}_a, & \text{with probability } 1-\epsilon, \end{cases}$$

where $\hat{r}_a$ denotes the estimated average reward of action a. The parameter $\epsilon \in [0, 1]$ controls the degree of exploration: higher ϵ values encourage trying less-explored options, while lower ϵ emphasizes exploitation of known high-performing choices. In dynamic systems such as satellite operations, ϵ can be decayed over time to favor exploitation once sufficient environmental knowledge is gained.

Upper Confidence Bound (UCB). An alternative approach, the *Upper Confidence Bound* (UCB) algorithm, handling exploration by associating each arm with a confidence

interval for its expected reward. At round t, the learner selects the arm that maximizes:

$$a_t = \arg\max_a \left(\hat{r}_a + c\sqrt{\frac{\ln t}{n_a}} \right),$$

where $\hat{r}_a$ is the empirical mean reward of arm a, n_a is the number of times arm a has been played, and c is a tunable parameter controlling the exploration intensity. The second term acts as an optimism bias—arms with higher uncertainty (smaller n_a) are favored early, ensuring that no potentially valuable action is neglected. UCB guarantees a logarithmic regret bound of $O(\log T)$ under stationary reward distributions, making it one of the most theoretically grounded algorithms in online learning.

Explore-Then-Commit (ETC). Another foundational algorithm in the online learning and multi-armed bandit literature is the *Explore-Then-Commit* (ETC) strategy [12]. As the name suggests, ETC explicitly divides the learning process into two distinct phases: an initial *exploration* phase followed by a *commitment* (or exploitation) phase.

In the exploration phase, each arm is sampled a fixed number of times—say, n_0 rounds per arm—to estimate its expected reward. Let $\hat{r}_a$ denote the empirical mean reward for arm a after n_0 samples:

$$\hat{r}_a = \frac{1}{n_0} \sum_{t=1}^{n_0} r_{a,t}.$$

After this exploratory stage, the algorithm *commits* to the arm with the highest estimated mean reward:

$$a^* = \operatorname*{argmax}_a \hat{r}_a,$$

and continues selecting a^* exclusively for the remaining $T - Kn_0$ rounds, where K is the total number of arms. Thus, the learner fully exploits the knowledge acquired during the exploration phase.

The simplicity of ETC makes targets selection tractable and easy to implement. Its regret

can be decomposed into two sources: (1) the cost of initial exploration (when suboptimal arms are deliberately played), and (2) the risk of committing to an incorrect arm if the initial estimates are noisy.

Conceptually, ETC mirrors many real-world decision processes: first, a system collects enough empirical evidence to assess possible actions; then, it locks onto the most promising option once confidence is sufficient.

In satellite operations, such a principle could be applied to early-mission phases, where different observation strategies are tested to estimate environmental conditions or target yields before committing to a steady observation policy for the remainder of the planning horizon.

However, ETC assumes a fixed reward distributions, which limit its capability in dynamic contexts such as Earth observation scheduling mission.

Relevance to satellite scheduling. For satellite task planning, online learning provides a conceptual framework for adapting to streaming data such as new observation requests or evolving priorities. For example, an ϵ-greedy policy could occasionally test alternative observation sequences to explore new combinations of targets which provides a fundamental area framework for space operation to find the optimal of sub-optimal combination of actions (policies) in a feasible time complexity. However, pure online learning assumes independent rounds and does not model the temporal or resource dependencies that characterize real satellite operations (e.g., attitude change, onboard memory, or overlapping visibility windows). Thus, while online learning introduces adaptivity and mathematical guarantees of diminishing regret, it lacks the sequential state transition and dynamic reward mechanisms needed for long-horizon optimization.

Reinforcement learning (RL) extends these ideas by embedding the learner within a dynamic environment, where each decision changes the future state of the system. Instead of minimizing regret, RL seeks to maximize long-term cumulative reward by learning an optimal

policy $\pi^*(a|s)$. In essence, RL can be viewed as *online learning with state transitions and delayed feedback*—a crucial property for complex, time-dependent problems such as satellite scheduling under uncertainty.

2.3.3 Deep Reinforcement Learning

After understanding online learning, now we turn to *reinforcement learning* (RL) which is a framework that generalizes the idea of sequential adaptation by explicitly modeling the interaction between a decision-making agent and a dynamic environment. In RL, the learner does not simply predict outcomes or minimize regret against a static comparator. Instead, it learns a *policy* that maps states to actions in order to maximize the expected cumulative reward obtained through repeated interaction with the environment [85, 43].

Core principle. Formally, an RL problem is modeled as a *Markov Decision Process* (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where:

- $\mathcal{S}$ is the set of states representing the environment,
- $\mathcal{A}$ is the set of possible actions available to the agent,
- $P(s'|s, a)$ is the state-transition probability from s to s' given action a,
- $R(s, a)$ is the expected immediate reward after taking a in s, and
- $\gamma \in [0, 1]$ is the discount factor controlling the importance of future rewards.

The agent's goal is to find an optimal policy $\pi^*(a|s)$ that maximizes the expected discounted return:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \right].$$

Deep Q-Network (DQN). In complex domains, the state or action spaces are high-dimensional and continuous, making traditional tabular RL infeasible. *Deep Reinforcement Learning* (DRL) addresses this limitation by using deep neural networks as function approximates for the policy and/or value functions.

Among the earliest and most influential breakthroughs in deep reinforcement learning is the *Deep Q-Network* (DQN), introduced by Mnih et al. [59]. DQN combines traditional Q-learning with deep neural networks to approximate the optimal action-value function $Q^*(s, a)$, enabling reinforcement learning to scale to high-dimensional and continuous state spaces that were previously intractable for tabular methods.

Concept and structure. In the classical Q-learning algorithm, the agent updates its Q values on the Q-table iteratively using the Bellman optimality equation as shown below:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha\big[r_t + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q_t(s_t, a_t)\big],$$

where α is the learning rate, and γ is the discount factor. This approach, however, becomes impractical when the state space is continuous or extremely large—as in satellite mission planning, where states must represent orbital position, onboard memory, attitude angle, and time-dependent target visibility.

DQN addresses this by introducing a neural network parameterized by θ to approximate the Q-function,

$$Q(s, a; \theta) \approx Q^*(s, a),$$

allowing the agent to generalize across similar states instead of storing explicit Q-values. The network takes as input a representation of the environment state and outputs estimated Q-values for each possible action.

The parameters θ are trained by minimizing the mean square root temporal-difference

(TD) loss:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\Big[\big(y_t - Q(s,a;\theta)\big)^2\Big],$$

where the target y_t is defined as

$$y_t = r_t + \gamma \max_{a'} Q(s', a'; \theta^-),$$

and θ^- represents the parameters of a separate *target network*, updated slowly to stabilize training.

Intuitive interpretation of TD loss. The *temporal-difference* (TD) loss measures how far the network's current prediction of the Q-value is from a target value derived from the next observed transition. In essence, it captures the "prediction error" between what the agent *expected* to happen and what it *actually* experienced after taking an action. If the two values are close, the TD loss is small, showing that the agent's understanding of the environment is accurate; if they differ greatly, the loss is large, signaling that the network should adjust its parameters to better match reality.

Mathematically, the term inside the square brackets in

$$L(\theta) = \mathbb{E}\Big[\big(y_t - Q(s,a;\theta)\big)^2\Big]$$

is the squared error between the **target value** y_t and the network's current **predicted value** $Q(s,a;\theta)$. The target y_t represents a one-step look-ahead estimate: it combines the immediate reward r_t the agent just received with the discounted estimate of future rewards from the next state,

$$y_t = r_t + \gamma \max_{a'} Q(s', a'; \theta^-).$$

In this way, the TD loss ensures that the network continuously corrects itself by using its

short-term predictions with longer-term outcomes.

To put it simply, the TD loss teaches the agent to refine its "guess" of how good each action is. Whenever the environment provides feedback that differs from the agent's expectations, the TD loss becomes nonzero, leading a correction through gradient descent. Over many interactions, these small corrections accumulate, allowing the agent to learn an accurate mapping from states and actions to expected future rewards.

For example, if the agent predicted that imaging a particular target would yield a reward of 5, but the resulting observation and future opportunities make the actual expected value closer to 8, the TD loss quantifies this 3-point discrepancy. By minimizing this loss, the network slightly increases the estimated value of that decision, improving its policy over time.

To further reduce correlation between sequential samples, DQN employs an *experience replay buffer* $\mathcal{D}$ that stores past transitions (s, a, r, s') and samples them randomly during training.

These two innovations—**experience replay** and the **target network**—are crucial for stable learning and prevent divergence in non-linear function approximation settings. The combination enables DQN to handle complex, high-dimensional tasks such as satellite attitude maneuvering, dynamic scheduling, where explicit enumeration of Q-values is impossible.

Applications in space operations. Deep Q-Networks have shown potential in solving decision making problems within aerospace and satellite operations. In mission scheduling scenarios, each state s_t can represent the satellite's instance configuration like position, attitude, onboard memory, remaining observation requests, and each action a_t may correspond to selecting the next target, adjusting acquiring mode, or initiating a targets downlink session. The reward r_t is typically designed to reflect mission objectives such as maximizing scientific value.

Recent studies have demonstrated that DQN based scheduler can outperform heuristic

and rule-based methods under dynamic or uncertain conditions. For instance, Ou et al. [64] formulated the *satellite range scheduling problem* as a Markov Decision Process (MDP) and proposed a DQN framework to dynamically allocate imaging windows among multiple requests. Their approach encoded temporal visibility constraints and task priorities into the state representation and used a multi-objective reward combining coverage and value gain. Their simulation results showed that the learned policy achieved higher total observation reward and better adaptability to task arrivals than greedy and GA-based baselines, particularly for high-demand scenarios.

In another representative work, Herrmann et al. [39] explored the use of DQN for *agile multi-satellite scheduling* under varying communication and visibility conditions. Their framework treats each scheduling step as a decision point where the agent selects which satellite should perform the next observation or downlink action. The state vector includes simplified operational variables such as satellite position, available memory, and communication link status, while the reward function balances task completion value and resource usage. Although they used a relatively lightweight DQN architecture, the agent learned adaptive policies which showed better performance than greedy and rule-based strategies, especially when communication delays or ground-station conflicts occurred. This study demonstrates that even with simplified state representations, reinforcement learning can effectively improve the responsiveness and flexibility of multi-satellite operations.

In a more recent contribution, Chun et al. [23] introduced the *Graph Attention Deep Q-Network* (GA-DQN) for AEOSS. In this model, observation targets and potential attitude transitions are represented as graph nodes and edges, respectively. A graph attention mechanism captures spatial relationships among targets, while the DQN framework learns to select the most promising target sequences given dynamic visibility and slewing constraints. This hybrid framework allows the model to generalize across target configurations and scale to hundreds of requests. Their experiments demonstrated better performance over classical

DP and heuristic based approaches in terms of total reward and computational efficiency which also achieving near real time scheduling for dense target scenarios.

In summary, these studies highlight that DQN serves as a general decision making framework which is capable of handling discrete actions and complex nonlinear state representations, which are good to real-world satellite mission planning.

When combined with constraint filters, heuristics, or encoder-decoder, DQN based schedulers can generate near-real-time decisions that improve space mission more efficiency and robustness.

Despite these advances, two main issues remain: (1) training deep models requires large datasets, which are not always available in space operations, and (2) most ML models act as black boxes, making it difficult for mission operators to interpret why a certain schedule was chosen. This lack of transparency limits their acceptance in safety critical phases.

This thesis is builds on these recent advances by introducing a dynamic, constraint-aware heuristic that not only respects memory, command, and geometric constraints but also allows merging of nearby targets and flexible mode assignment. Unlike many prior approaches, the method provides a transparent and explainable scheduling framework that performs well under dynamic conditions and is suitable for potential onboard implementation.

Part II

Satellite Mission Overview

2.4 Introduction to Sentinel-6 Mission

The Sentinel-6 satellite, which launched in November 2020, is one of the most important parts of the international Earth Observation program. It is a joint program between the European Space Agency (ESA), EUMETSAT, NASA, and NOAA. The main goal of Sentinel-6 is to keep the long record of radar altimetry going after the TOPEX/Poseidon and Jason satellite series. It mainly focuses on precise measurements of global sea level, ocean circulation, and different climate change indicators.

Sentinel-6 operates in a Low Earth Orbit (LEO) at around 1,336 km altitude and completes an orbit every 112 minutes. The orbit repeats every 10 days, which means it can cover the whole Earth in a consistent pattern. The orbit geometry is very stable and helps ensure that the satellite can compare measurements over time, which is important for detecting both long-term and short-term changes in sea level or river altitude.

2.5 Poseidon-4 Radar Altimeter and Observation Modes

The main scientific instrument on Sentinel-6 is the Poseidon-4 radar altimeter, which can work in several modes. These modes allow the mission to choose between better resolution and lower data volume, depending on what the situation needs.

- **Low Resolution Mode (LRM)**: Standard mode with moderate accuracy and efficient data compression, mainly used for open oceans.
- **Low Resolution Mode with Range Migration Compensation (LRMC)**: A middle option that improves resolution while keeping memory usage reasonable.
- **LX mode**: Records uncompressed, very detailed data that are most useful for studying coastlines, lakes, and rivers, but it uses much more onboard memory.

These modes are selected dynamically based on mission goals, geographic region, and

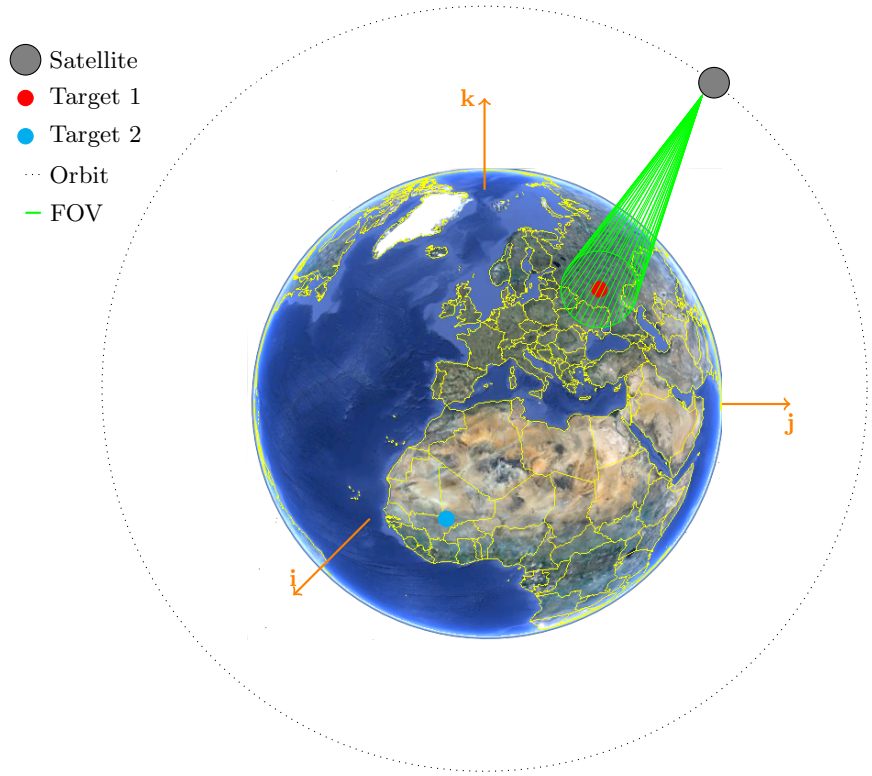


Figure 2.4: Example of target data acquisition. The satellite flies along its orbit and starts recording when the targets are inside the field of view (FOV).

available onboard resources. The RAW mode is the most valuable for scientific purposes, but also the most expensive in terms of memory usage. Because of this, it must be carefully scheduled so that the satellite memory does not overflow and all data can still be downloaded later.

2.6 Scientific Importance

Sentinel-6 plays a crucial role not only in ocean science but also across a wide range of Earth observation applications. The data collected by the Poseidon-4 altimeter are extensively used in:

- Numerical weather prediction and the improvement of wind and wave models.
- Flood monitoring and disaster response planning.

- Climate and hydrology studies, including sea-level rise and glacier monitoring.

These applications highlight how altimetry data benefit both scientific research and everyday life, particularly as climate change intensifies hydrological variability and environmental uncertainty.

2.7 Onboard Data Handling and Ground Operations

Sentinel-6 has a limited onboard mass memory unit that stores altimetry data until it can be sent to the ground. Data are usually downloaded when the satellite passes over one of EUMETSAT's ground stations in Europe or North America. Each downlink pass lasts only a few minutes, so the planning team must make sure that data are stored efficiently and that the memory buffer never gets full.

If the onboard memory becomes full before a scheduled downlink, newly acquired targets will overwrite previously stored data, leading to corrupted or lost measurements, which is unacceptable for scientific operations. To avoid this, ground operators must carefully plan recording sessions and downlink windows to ensure that the memory is efficiently utilized without ever exceeding its capacity.

2.8 Sentinel-6 Targets Dataset Overview

During the mission planning phase, scientists and schedulers select observation targets from a global dataset containing more than 3,000 predefined altimetry targets distributed across the Earth's surface over 20 orbits. Therefore, a comprehensive understanding and analysis of this dataset are essential before further mission planning steps. Table 2.2 presents a partial sample of the target records used in this research.

As shown in Table 2.2, each target record contains several descriptive parameters (with irrelevant columns omitted here), explained as follows:

Table 2.2: Sample of Satellite Target Records with Start and End Coordinates

start_latitude	start_longitude	end_latitude	end_longitude	pso	duration	duration_pso	entity	r_orb
40.406585	-3.400501	40.410892	-3.395674	33.590098	15.847397	0.122982	nadir	17
41.528379	-2.563479	41.534652	-2.557913	34.795143	2.309398	0.123227	nadir	17
41.528379	-2.563479	41.540298	-2.549821	34.918370	16.148573	0.123227	nadir	17
42.355076	-1.921616	42.360243	-1.915879	35.780036	2.321602	0.123877	nadir	17

- **start_latitude / start_longitude:** The geographic coordinates marking the starting point of each target along the satellite's ground track.
- **end_latitude / end_longitude:** The geographic coordinates marking the ending point of the target segment. Together with the start coordinates, they are used to calculate the target's spatial distance using the great-circle formula.
- **pso:** The position angle of the satellite with respect to the target, indicating its orientation during acquisition.
- **duration:** The time window available for data acquisition over the target, typically expressed in seconds.
- **duration_pso:** The corresponding duration in positional space, associated with the PSO angle.
- **entity:** The observation configuration or mode used for data collection (e.g., *nadir*).
- **r_orb:** The orbital revolution number representing the satellite's specific orbit pass when the observation occurs.

These fields form the core of the target dataset used for trajectory and scheduling optimization in subsequent algorithmic analysis.

Figure 2.5 below is the map that illustrates the targets from the dataset pinned on the Earth map. As shown in the map, different targets on different satellite tracks are indicated with different colors, and we have more than 20 tracks circling around the Earth. Some

of the targets are observation of the rivers, lakes, and reservoirs distributed across different countries and continents.

In addition to geographical distribution, it is also important to understand the spatial characteristics of the selected targets. Each altimetry target is defined by its start and end coordinates along the satellite ground track. The target size is computed as the great-circle distance between the start and end points on the Earth's surface, taking the planet's curvature into account.

Mathematically, the great-circle distance d between two points with latitude and longitude (ϕ_1, λ_1) and (ϕ_2, λ_2) is calculated using the haversine formula:

$$d = 2R \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

where R is the mean radius of the Earth (approximately 6371 km). It should be noted that the haversine formula assumes a spherical Earth model, which introduces a small approximation error which is within 0.1 km compared to more accurate ellipsoidal models. However, for the scale and resolution of the target acquisition problem considered in this work, this error is negligible and does not significantly affect the overall optimization results.

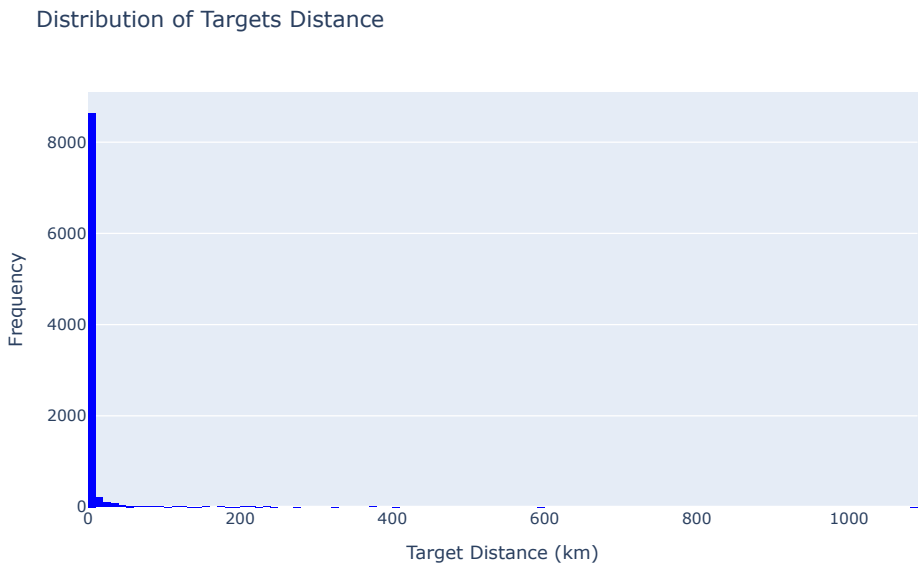
Table 2.6 summarizes the statistical results of all computed target distances in the dataset. These values indicate that most targets are short in spatial extent (less than 1 km), while a few large segments cover over 1000 km, typically representing oceanic or polar tracks.

More importantly, since all targets are distributed across the Earth's surface, it is also essential to analyze how far they are separated from one another. The distance between two adjacent targets along the same satellite ground track is defined as the **target gap**. Understanding these gaps provides valuable insight into the spatial continuity of observations and helps determine whether consecutive targets can be efficiently acquired within a single orbital pass.

Figure 2.5: Illustration of Targets Distribution Over Orbits.

Figure 2.6: Statistical Summary and Distribution of Target Size

Statistic	**Value (km)**
Count	9165
Mean	3.18
Standard Deviation	19.70
Minimum	0.00
25th Percentile	0.02
Median (50th Percentile)	0.10
75th Percentile	0.92
Maximum	1086.10



From an algorithmic perspective, the target gaps directly influence the configuration of several meta-parameters in the proposed optimization model, including the discretization step size, bin allocation strategy.

Smaller gaps indicate spatially continuous coverage, which is preferable for improving onboard memory utilization. On the other hand, larger gaps imply discontinuous observation segments, which require the algorithm to consider acquisition modes more carefully in future work.

An illustration of these target gaps along a representative orbital track is shown in Figure 2.7, where each gap corresponds to the distance between two consecutive targets. This visualization highlights how spatial distribution patterns vary across orbits and underscores the importance of adaptive parameter tuning in subsequent optimization stages.

The statistical summary shown in Figure 2.8 reveals several important characteristics of the target gap distribution. First, although the mean gap is approximately 978 km, the median is only 188 km, indicating that the distribution is highly right-skewed. This reflects the fact that most adjacent targets along a given orbital track are relatively close to one another, while a smaller number of exceptionally large gaps—up to nearly 19,100 km—account for the long tail of the distribution. These large gaps typically correspond to transitions between geographically separated regions, such as moving from continental landmasses to open-ocean areas or between distinct hydrological zones.

The histogram in Figure 2.8 provides a clearer visualization of this skewed behavior. The density of small gaps suggests that many targets lie within the same regional cluster or form continuous acquisition segments.

Overall, these results suggest that the majority of targets correspond to localized observation areas, whereas a smaller subset consists of long continuous ground tracks. This characteristic distribution will influence both the memory allocation and the prioritization strategy in the later optimization algorithm.

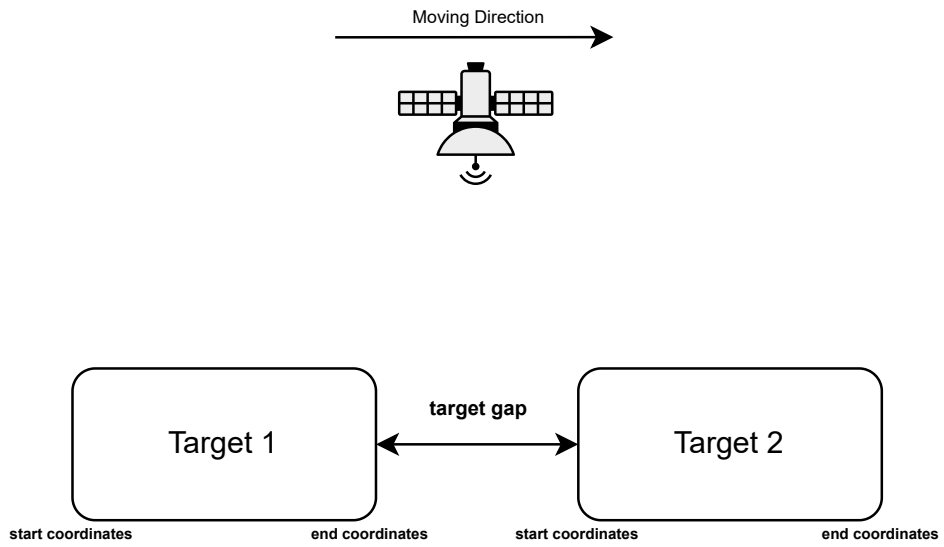


Figure 2.7: Illustration of Target Gaps.

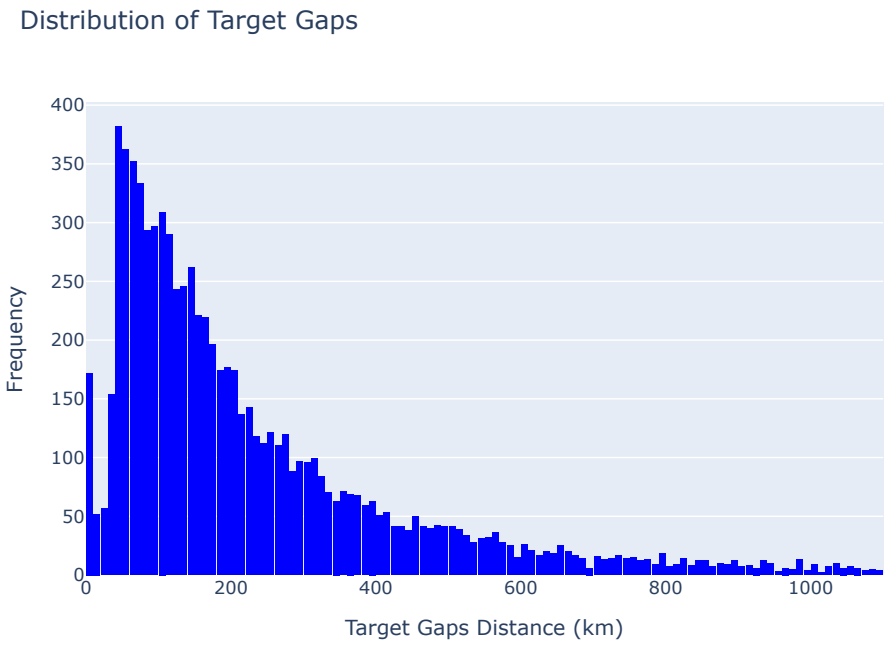
2.9 Altimetry Targets and Mission Planning Challenges

Altimetry targets are the designated locations on the Earth's surface that Sentinel-6 is required to observe, such as ocean sectors, coastlines, rivers, glaciers, or large reservoirs. Each target is given a *priority level* that represents how important it is scientifically or operationally. For example, a coastal region or a river basin might have a higher priority than open-ocean measurements if it is used for flood monitoring or climate validation. Every target also has an observation duration, a preferred data acquisition mode (for instance LX, LRMC, or LRM), and an associated memory cost depending on the selected mode.

Planning which targets to observe and which mode to use is far from simple. It is a large-scale combinatorial optimization problem that resembles the classic *knapsack problem* like we mentioned before, where planners must decide how to fill the limited satellite memory with the most valuable data. However, the problem here is even harder because the duration and the value of each observation are not fixed since they depend on orbital geometry, data

Figure 2.8: Statistical Summary and Distribution of Target Gaps

Statistic	**Value (km)**
Count	9164
Mean	977.82
Standard Deviation	2623.38
Minimum	0.00
25th Percentile	94.46
Median (50th Percentile)	188.11
75th Percentile	423.82
Maximum	19093.98



rate, and the available modes. As a result, this planning task is a non-convex and NP-hard problem.

The mission planners must take into account several operational and physical constraints when generating a plan:

- **Memory constraint:** The total volume of data generated by all selected targets must never exceed the onboard mass memory at any time. Since LX mode consumes much more space than LRM, this limits how often it can be used within a single orbit.
- **Commands limit:** There is a maximum number of instrument commands that can be uploaded for each planning cycle. And the typical number of commands for a operational cycle is 400 due to either mission cycle configurations or legacy satellites design. Too many mode switches or segment initiations can lead to operational errors or execution delays.
- **Geometric and temporal dependencies:** Selecting one target may block the opportunity to observe another, either because of overlapping time windows or insufficient buffer memory. These dependencies mean that local decisions (within one orbit) also affect the global plan across the entire planning cycle.

Because Sentinel-6 is not designed for real-time replanning, all of these scheduling decisions must be made in advance. The ground operation team generates a daily or weekly command plan that specifies when and how the instrument will record data, what modes will be used, and which targets will be skipped if there are conflicts. Once the plan is uplinked to the spacecraft, it runs autonomously for the duration of the cycle.

An inefficient or poorly optimized plan can cause serious data loss. For instance, if too many high-resolution targets are scheduled early in the orbit, the onboard memory may become full, preventing the recording of later high-priority targets. Similarly, if command or timing constraints are violated, the plan might be rejected or require manual re-editing,

delaying the whole operation.

Therefore, the mission planning team has to find a delicate balance between maximizing the total scientific reward and maintaining operational safety. This is made even more difficult by the constantly changing conditions: new target requests from scientists, unpredictable anomalies, and varying downlink opportunities. Because of this dynamic environment, it is not enough to rely on static or brute-force methods.

The problem requires an adaptive and constraint-aware approach that can adjust the selection of targets and recording modes while keeping memory usage, command counts, and orbital geometry under control. Developing such an approach is one of the main motivations of this thesis.

2.10 Motivation for Optimization Research

Because of all these constraints and goals, efficient scheduling becomes a key challenge. Traditional scheduling tools are often not flexible enough to deal with the increasing number of user requests and strict memory limits. Therefore, this thesis uses Sentinel-6 as the test mission and performance reference for developing a new **constraint-aware heuristic** algorithm.

The main idea is to create a method that can:

- Maximize the number of valuable data acquisitions.
- Respect memory and command constraints.
- Stay understandable and explainable for operators.

This approach aims to reduce manual work and make the planning more adaptive to changing situations. In the future, such heuristics could even be part of onboard autonomous scheduling systems for next-generation altimetry missions.

Part III

Algorithm Overview and Implementation Framework

As discussed in the previous chapters, space operation missions such as altimetry data acquisition require a high level of transparency and efficiency. In other words, it is not enough for the algorithm to simply produce an optimized plan or schedule since they must also be explainable. Mission scientists and schedulers need to understand how and why the algorithm makes certain decisions, especially when those decisions affect valuable observation opportunities or memory consumption. This transparency not only improves trust in the system but also enables human experts to fine-tune operational parameters when necessary.

Our proposed heuristic algorithm was developed with these principles in mind. It aims to provide a balance between performance and interpretability, ensuring that every scheduling decision can be traced back to its underlying constraints and objectives. By doing so, it allows mission operators to visualize how the satellite's onboard resources such as memory capacity, number of commands which are being utilized.

In general, our algorithm can be divided into three main components:

1. **Satellite Trajectory Initialization**, which import the orbit and potential data into the program, and models them in our program.

2. **Target Preprocessing**, which evaluates and filters the list of observation targets according to mission priority, and meets the memory constraint conditions.

3. **The Main Heuristic Algorithm**, which applies constraint-aware decision rules to select the most valuable targets and observation modes while respecting rest of all mission constraints.

These three parts work together to simulate a realistic decision-making process under operational constraints. The heuristic integrates information about onboard memory and command availability to make efficient and explainable scheduling choices. It is designed to maximize the number of high-priority altimetry data acquisitions while avoiding overuse of resources or command overload.

Furthermore, the heuristic is adaptable: it can dynamically adjust to mission parameters such as changing target distributions or updated priority scores. This flexibility makes the approach suitable for both static pre-planned schedules and semi-dynamic mission updates.

In the following sections, each part of the algorithm will be introduced in detail. We will begin by describing the satellite trajectory initialization process, followed by target preprocessing, and finally, the full structure of the constraint-aware heuristic that integrates all these components into a unified optimization framework.

CHAPTER 3

SATELLITE TRAJECTORY INITIALIZATION

3.1 Overview

Before the heuristic scheduling algorithm can be applied, the satellite trajectory and target structure must be properly initialized in the program. This process ensures that all observation targets are correctly mapped into the satellite's ground track, discretized into manageable bins, and associated with relevant operational metadata such as priority, observation modes, and estimated memory usage, which is also associated with satellite altitude.

The goal of this initialization stage is to establish a clear, structured representation of the satellite's potential observation timeline. Each step of the initialization directly contributes to ensuring that subsequent optimization and heuristic planning can operate on transparent and well-defined data structures.

3.2 Targets Discretizations

In real-world altimetry missions, a single observation target can be extremely large. For example, oceanic or river targets can extend over hundreds of kilometers. The footprint size of the satellite's radar altimeter varies with its orbital altitude and local surface conditions, meaning that a single "target" may cover a wide range of altitudes and geographical features. Consequently, acquiring each target as one indivisible observation unit can lead to poor optimization results, since different portions of the same target may require different data-acquisition modes or memory allocations.

To overcome this, the proposed system introduces the concept of `Bins`, where each target is divided into a set of smaller, spatially continuous segments along the satellite's trajectory. Each bin represents a segment between two discretized endpoints and is treated as an independent operational unit.

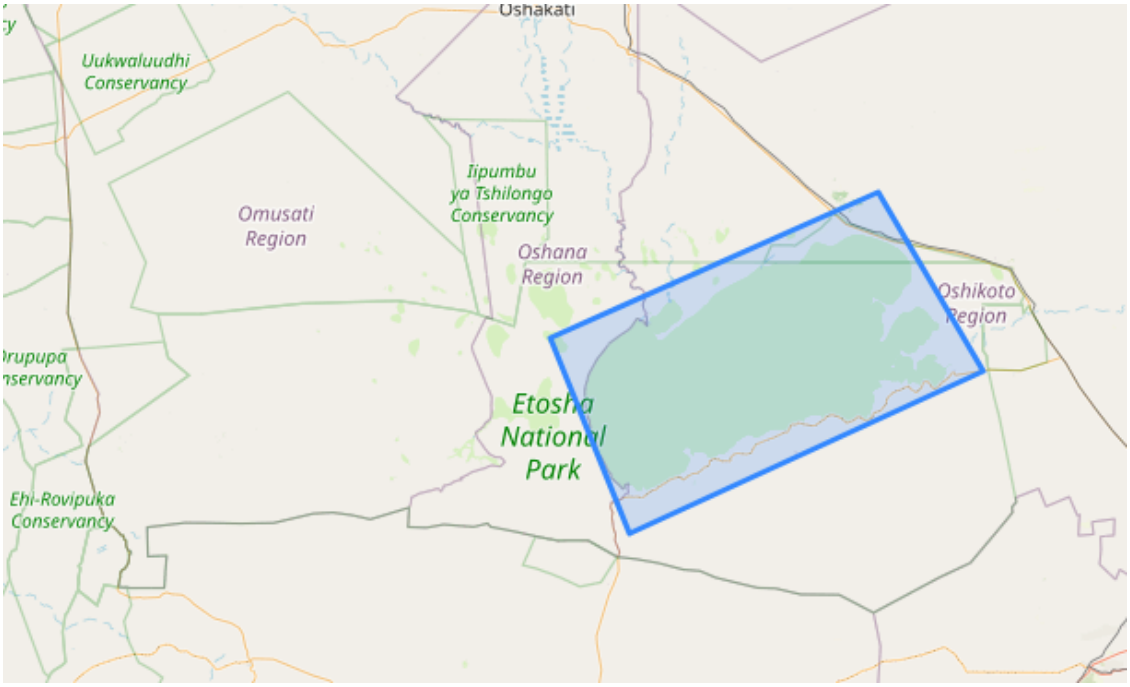


Figure 3.1: Example of target footprint. The Etosha lake is depicted as polygon.

In this implementation, all bins are generated with a fixed spatial width determined by a mission-defined parameter called the *step size*. The step size is a meta-parameter set by the Mission Operation Center (MOC), typically expressed in kilometers. It defines the ground distance between two consecutive endpoints along the satellite's ground track. As a result, each bin has a uniform width, ensuring consistent spatial coverage and simplifying subsequent processing and memory modeling.

- **Mode Acquiring Flexibility:** By representing each target as multiple bins, the algorithm can make finer-grained scheduling decisions. Some bins within a large target can be recorded in high-resolution mode (e.g., LX), while others can be downgraded to a lower-resolution mode (e.g., LRMC) if the available memory is limited.

- **Memory-Aware Optimization:** Different bins of the same target may occupy varying amounts of onboard memory depending on the observation mode. This variability allows the heuristic algorithm to balance scientific value against memory cost, achieving

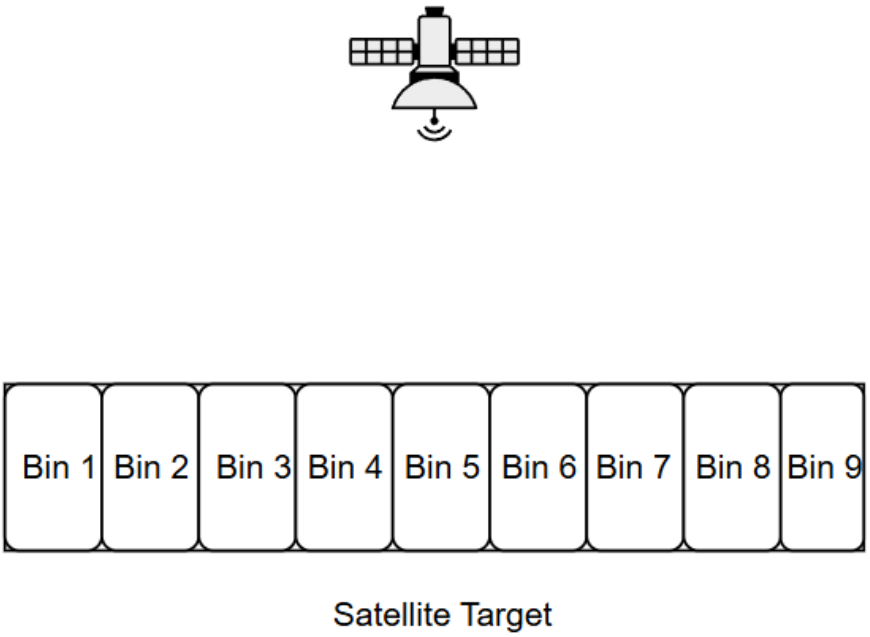


Figure 3.2: Example of targets Discretizations.

a more efficient overall plan.

- **Adaptation to Orbital Geometry:** The satellite's ground track changes slightly in each orbit, and the angle of incidence between the sensor and the target surface can affect data quality and duration. Modeling targets as bins allows the system to handle such geometric variations, since each bin is associated with its specific orbit index and latitude–longitude range.

- **Transparency for Analysis:** A bin-based structure enables mission planners to understand precisely which segment of a target is being observed and at what mode. This transparency is essential for post-analysis, since scientists can trace data quality back to specific trajectory segments.

In summary, the bin-based representation transforms the complex, continuous surface of the Earth into a structured, discrete model that is both physically meaningful and computationally tractable. This methodology not only enhances the explainability of the algorithm but also significantly improves its adaptability under real mission constraints.

3.3 Core Concepts and Data Structures

To support modular development and code transparency, several core data structures are defined in Python, each representing a key component of the observation system.

3.3.1 EndPoint Class

The `EndPoint` class represents a single geographical coordinate on the Earth's surface, defined by its latitude and longitude. These endpoints are used to discretize one target into smaller segments. Each endpoint is associated with a type (for example, a target name), allowing them to be grouped later for bin construction.

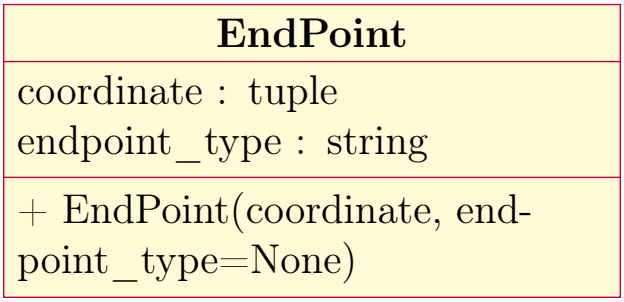


Figure 3.3: UML class diagram of the `EndPoint` data structure.

3.3.2 Mode Class

Each observation mode (for instance, LRMC or LX) is represented by the `Mode` class. This class stores both the mode type and its corresponding memory usage value. Higher-resolution modes, such as LX, consume more onboard memory but provide better data quality, while modes like LRMC are used when memory is limited.

3.3.3 Bin Class

The `Bin` class forms the backbone of the initialization process. Each bin represents a discrete observation segment along the satellite's path, defined by a pair of consecutive endpoints. It

Mode
mode : string memory_usage : float
+ Mode(mode, memory_usage)

Figure 3.4: UML class diagram of the `Mode` data structure.

contains attributes such as observation modes, target type, priority, and orbit index.

This structure provides the necessary abstraction for heuristic algorithms to later evaluate which bins to activate, downgrade, or skip based on memory and priority constraints.

Bin
fore_endpoint : Endpoint rear_endpoint : Endpoint bin_type : string bin_range: number bin_pso: number bin_pso_delta: number index : int mode_list : list priority : float
+ Bin(fore_endpoint, rear_endpoint, bin_type, bin_range, index, mode_list=None)

Figure 3.5: UML class diagram of the `Bin` data structure.

3.3.4 Target Class

The `Target` class provides a more constructive operational objects used in the command-constrained acquisition stage. While the pre-processing phase operates primarily at the bin level, the command constraint is naturally expressed at the target level, since each target corresponds to a contiguous recording segment (or a merged segment) and therefore directly contributes to the command budget. A `Target` aggregates a list of bins that share the same globally unique target identifier (`target_name`), and stores orbit-dependent metadata such as the `orbit_number` and `target_orbital_index`, which enables fast same-orbit neighbor

search during merging.

In addition, `Target` encapsulates mode-related and merging-related properties required by later stages. The flag `is_mode_hybrid` determines the command cost of acquiring the target, while the list `merged_targets_list` records intermediate targets that may have been unintentionally acquired when constructing a merged recording interval.

Furthermore, `start_pso` and `pso_duration` provide a compact representation of the target footprint (and are updated whenever merging occurs). Finally, utility methods such as `memory_usage`, `command_usage`, and `set_all_bins_to_mode` allow the planner to evaluate feasibility and enforce uniform recording modes when necessary.

Figure 3.6 illustrates the UML class diagram of the `Target` data structure.

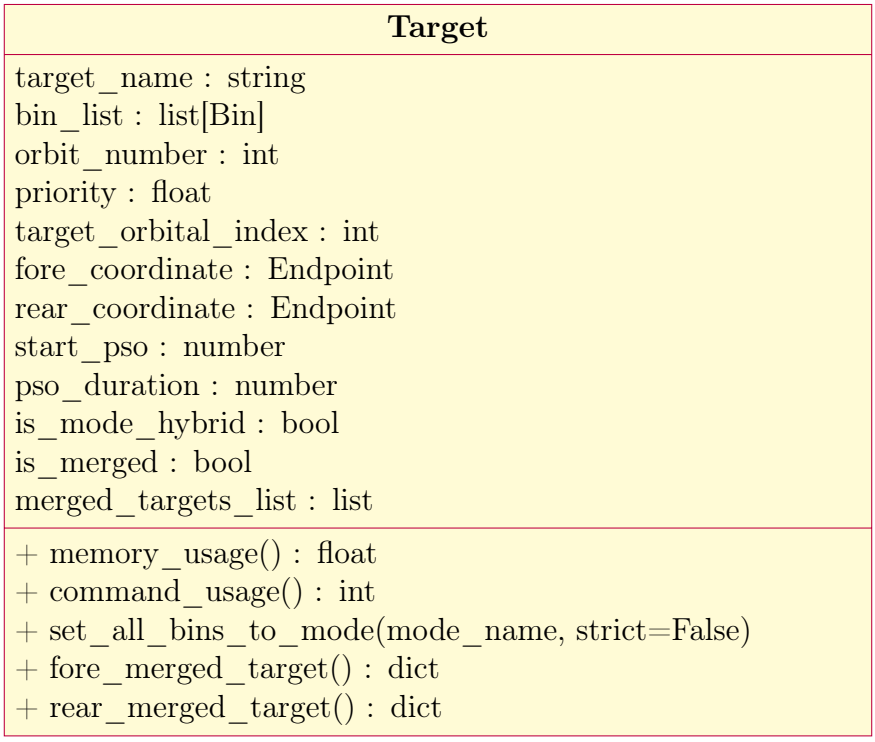


Figure 3.6: UML class diagram of the `Target` data structure.

3.4 Initialization Pipeline

The overall initialization is handled by the `trajectory_initialization()` function, which follows a sequential structure. It performs four major tasks:

1. Importing the targets' coordinates from the input data files.
2. Discretizing each target's trajectory into endpoints.
3. Creating bins between consecutive endpoints.
4. Assigning memory usage, observation modes, and priority based on the XML configuration file.

3.5 Endpoint Initialization

The first step involves reading target data and generating endpoints. For each target defined by its start and stop coordinates, a discretization function divides the line segment into evenly spaced points, with the interval defined by a user-selected step size (e.g., 10 km). Each point is stored as an `EndPoint` object.

The initialization process of algorithm 1 begins by extracting the subset of targets selected by the user. For each desired target, the algorithm retrieves its associated start and stop coordinates that define the target's ground track. Using these coordinates, the algorithm discretizes the line segment into a sequence of evenly spaced points, where the spacing is determined by a user-specified step size (e.g., 5 km).

Each discretized point is converted into an `EndPoint` object containing its latitude, longitude, and target label. These endpoints serve as the foundation for bin construction in later stages of the workflow. All generated endpoints are appended to a global list that preserves their order along the ground track. This ensures that subsequent steps—such as bin creation, range computation, and memory estimation—operate on a consistent and well-defined

spatial representation of each target.

Algorithm 1 Initialization of Target Endpoints

```
1: procedure InitializeTargetEndpoints(end_point_list, step_km)
2:     T ← GetAllTargets
3:     U ← GetAllWantedTargetsName
4:     for all t in T do
5:         if t.target_name ∈ U then
6:             start_coordinate ← (t.start_latitude, t.start_longitude)
7:             stop_coordinate ← (t.stop_latitude, t.stop_longitude)
8:             E ← DiscretizeLine(start_coordinate, stop_coordinate, t.target_name, step_km)
9:             for all e in E do
10:                append e to end_point_list
11:            end for
12:        end if
13:    end for
14: end procedure
```

This discretization ensures that each target can later be represented as a sequence of bins that closely follow the satellite's actual orbit geometry.

3.6 Bin Initialization and Attribute Assignment

After endpoints are generated, they are grouped by target name to construct bins. Each bin connects two consecutive endpoints, forming the smallest operational unit for scheduling.

In this Algorithm 2, it outlines the complete bin-construction procedure. For every user-selected target, the algorithm first extracts all associated endpoints and orders them along the track. The number of bins is determined by the starting and ending endpoint pairs. For each pair, the algorithm computes the bin duration, the starting PSO value, the PSO increment (pso_δ), and the relevant orbital parameters. A `Bin` object is then instantiated with all required attributes—start and end coordinates, bin index, bin PSO range, duration, and orbit number—and appended to the global bin list. This structured representation ensures that all memory estimation and heuristic scheduling, operate on a the bin set.

Algorithm 2 Initialization of Bins for All Targets

procedure InitializeTargetBins(bin_list, end_point_list)
    $T \leftarrow$ GetAllWantedTargetsName ▷ List of target names selected by the user
    for all t in T **do**
        $P \leftarrow \{\, edp \in \text{end\_point\_list} \mid edp.\text{endpoint\_type} = t \,\}$
        $n\_bins \leftarrow |P| - 1$
        if $n\_bins \leq 0$ **then**
            report that target t has no valid endpoints
            continue
        end if
        $duration \leftarrow$ GetBinDuration($n\_bins,\ t$)
        $start\_pso \leftarrow$ GetTargetStartPSO(t)
        $orbit \leftarrow$ GetOrbitByName(t)
        $target\_pso\_duration \leftarrow$ GetTargetDurationPSO(t)
        $pso\_delta \leftarrow \dfrac{target\_pso\_duration}{n\_bins}$
        for $i = 0$ to $n\_bins - 1$ **do**
            $bin\_start\_pso \leftarrow start\_pso + pso\_delta \cdot i$
            $starting\_endpoint \leftarrow P[i]$
            $ending\_endpoint \leftarrow P[i+1]$
            append **Bin**($starting\_endpoint,\ ending\_endpoint,$
              $bin\_type = t,\ index = i,$
              $bin\_pso = bin\_start\_pso,$
              $bin\_pso\_delta = pso\_delta,$
              $duration = duration,\ orbit = orbit$) to bin_list
        end for
    end for
end procedure

3.7 Target Memory Calculation Assignment

The calculation of target memory requires a dedicated explanation because it relies on the proposed bin–based discretization methodology, which differs significantly from other approaches. This section introduces the geometric quantities and measurement data that are essential to the construction of the memory model, and we will begin with a couple of concept explanations.

3.7.1 Range

In real-world satellite missions, the distance between the satellite and the target Earth's surface—commonly referred to as the *range*—is neither constant nor idealized as shown in simplified geometric figures. Instead, the range varies continuously due to the ellipsoidal shape of the Earth, gravitational effects, the surface altitude of the target, and small orbital perturbations. A simplified illustration of the concept of range is as in the Figure 3.7:

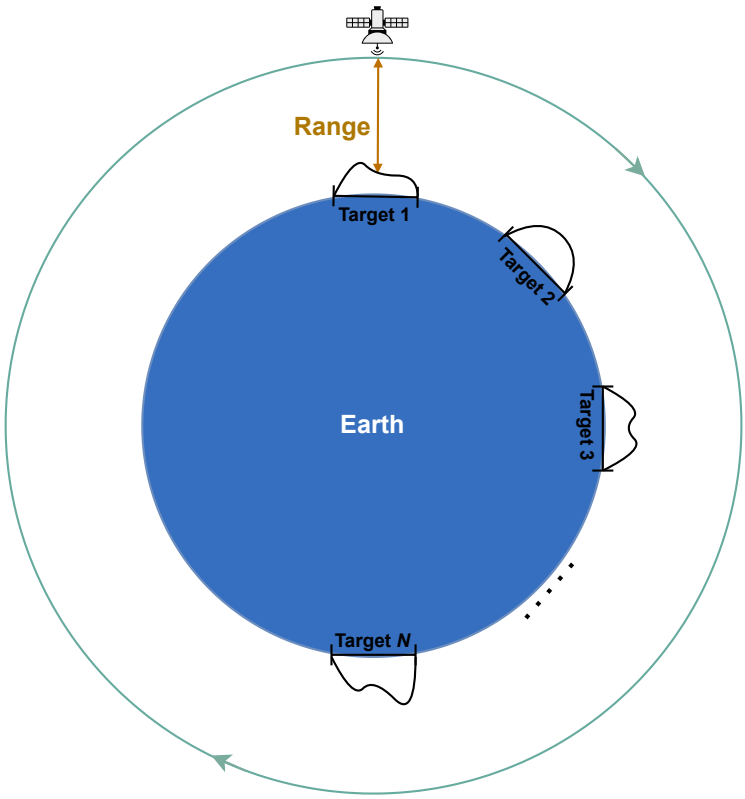


Figure 3.7: Illustration of Range.

3.7.2 Pointing Separation Angle (PSO)

In addition to range, the second key geometric quantity required for the memory calculation is the *Pointing Separation Angle* (PSO). In our context, the PSO represents the geocentric angular separation between the satellite and the target ground point at either the beginning or end of a visibility window. More precisely, the PSO is defined as the angle formed at the Earth's center between the satellite position vector and the vector pointing to the target location.

The value used at the beginning of the visibility interval is denoted as the *starting_pso*. This quantity describes the geocentric angular separation at the moment the bin first becomes

observable, and it defines the initial geometry of the bin–target configuration. Unlike the off-nadir angle, which is defined from the satellite's perspective, the PSO uses the Earth's center as the vertex and therefore captures the surface-relative geometry of visibility.

As the satellite moves along its orbit, the PSO evolves smoothly according to the relative motion between spacecraft and the surface point. The incremental change during the visibility window is represented by the parameter *duration_p*, and the end of the target PSO interval is computed as

$$\text{ending\_pso} = \text{starting\_pso} + \text{duration\_p}.$$

Together, as shown in Figure 3.8, these angles describe the geocentric angular sweep associated with each bin, and this sweep strongly influences the variation of range throughout the bin's visibility duration. As the parameter name *duration_p* suggests, the PSO is one way to measure the duration of a recording window: the bigger the angle, the bigger the window and therefore the higher the memory cost.

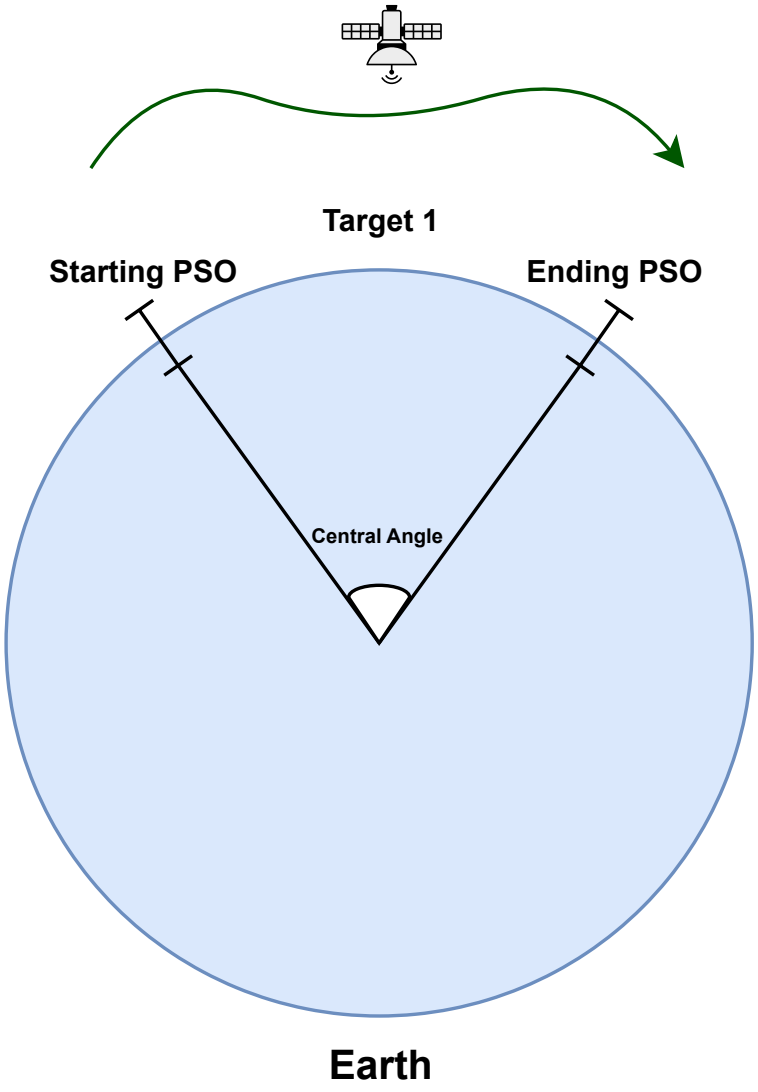


Figure 3.8: Illustration of Range.

3.7.3 Range and PSO Data From ESA

To accurately model how memory varies within a bin, our algorithm incorporates high–precision range and pointing–geometry measurements provided by the European Space Agency (ESA). These measurements include detailed range values across different orbital segments, together with their corresponding PSO values.

Since the on–board memory consumption of a target is highly dependent on the satellite–to–surface range, these measurements play a crucial role in computing the memory footprint of each discretized bin.

3.7.4 Bin-Level Memory Calculation

Before computing the memory of a full target, it is essential to compute the memory associated with each individual bin, because the bin is the smallest representational unit in our methodology. Each bin captures a small, spatially continuous segment of the target footprint, and the memory of the entire target is reconstructed by summing the memory associated with its bins.

The table below illustrates a subset of the range and pointing–geometry dataset used by our memory model.

orbit_number	pso_angle_centi_deg	range	pso_angle_deg
1	0	1337528.72	0
1	5	1337528.03	0.05
1	10	1337527.37	0.10
1	16	1338738.73	0.16
1	21	1338738.12	0.21
1	26	1338737.54	0.26
1	32	1337900.99	0.32
1	37	1337900.46	0.37
1	42	1337899.96	0.42
1	48	1337899.49	0.48
1	53	1339417.04	0.53
1	58	1339416.62	0.58

Table 3.1: Range and angular parameters for orbit 1 (altitude removed).

Column Descriptions:

- **orbit_number**: The orbital revolution index associated with the measurement. Each value corresponds to one full revolution of the satellite around the Earth.

- **pso_angle_centi_deg**: The geocentric central angle between the satellite nadir point and the surface point, expressed in centidegrees (0.01°). This value represents a uniformly sampled angular position along the visibility geometry, enabling fine-grained interpolation of the range profile.

- **range**: The satellite-to-surface distance (in meters) corresponding to the PSO angle shown in the same row.

- **pso_angle_deg**: The same geocentric central angle as *pso_angle_centi_deg*, but expressed in degrees for convenience.

As shown in the table above, the orbital range data has been discretized into rows sampled at increments of approximately 0.05°. Our task is therefore to determine the appropriate range value for each bin in the discretized target representation.

In the second phase of the pipeline, each bin is already assigned two quantities: *bin_pso* and *bin_pso_delta*, which together describe the angular interval spanned by the bin. To obtain a single representative geometric angle for range lookup, we compute the central pointing separation angle using

$$\text{central\_pso} = \text{bin\_pso} + \frac{\text{bin\_pso\_delta}}{2}.$$

This central angle serves as the reference for interpolating the satellite–surface range as shown in the Figure 3.9. We then locate the closest entry in the discretized PSO range table to obtain the corresponding range value for that bin. Once the range is determined, the memory consumption of the bin can be computed based on the user-provided observation mode list.

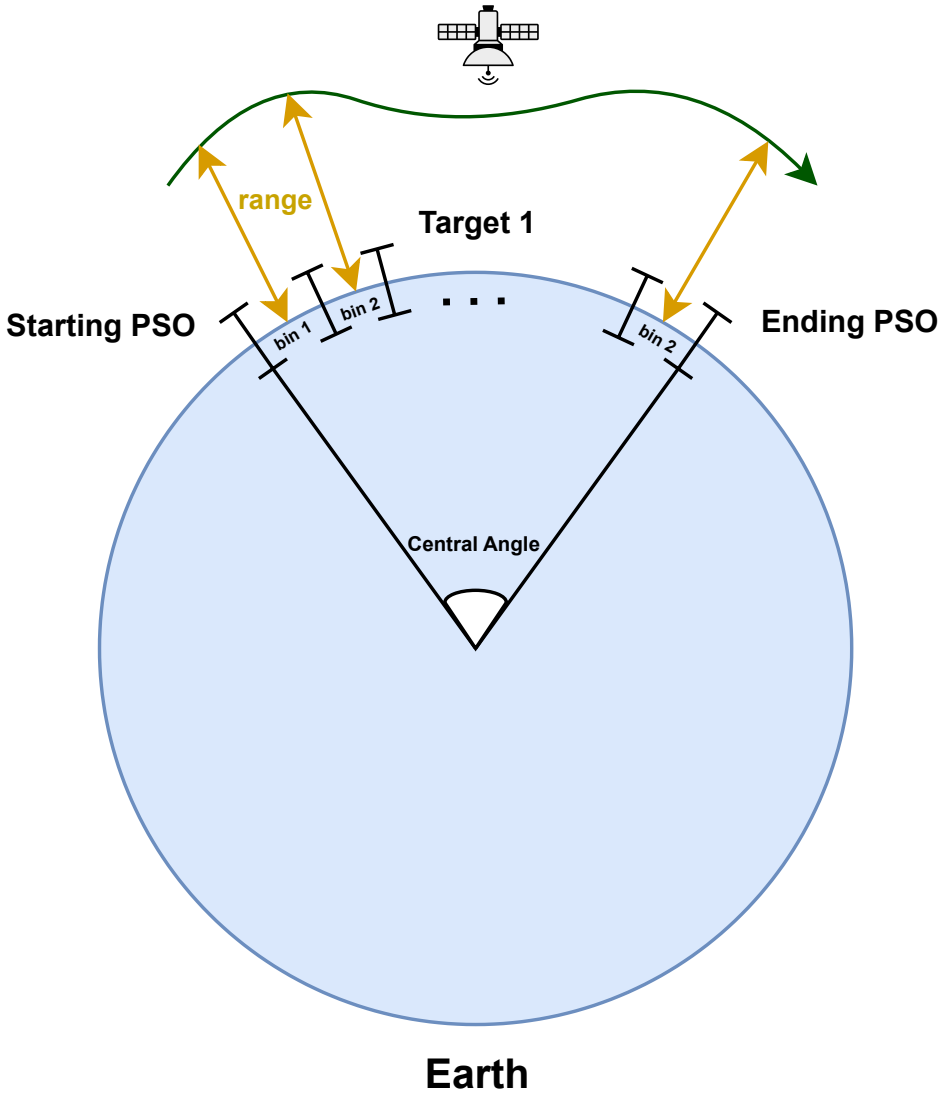


Figure 3.9: Illustration of Range in Targets

At this final stage of the initialization pipeline, each `Bin` object has already been fully constructed with its attributes, including start and end coordinates, range, duration, and

PSO interval. With these properties in place, the next objective is to determine the memory usage required to acquire each bin under the user-specified recording modes. This step is essential, as onboard memory is one of the dominant mission constraints of any acquisition plan.

The memory computation workflow consists of two components. First, as shown in Algorithm 3, the system assigns priorities and available recording modes to every bin based on the user-provided XML configuration file. Each target may specify one or multiple acquisition modes (e.g., high-resolution: LX or medium-resolution: LRMC), and the corresponding data rates are retrieved from mission parameters. For each bin–mode pair, the algorithm invokes the memory calculator.

The second component, detailed in Algorithm 4, computes the actual memory usage for a single bin. This calculation incorporates mission-specific constants, a geometric correction factor, the bin's spatial range, and the mode-dependent raw data rate given by the ESA. The resulting value represents the expected number of bytes required to record that bin in the specified mode. All memory values are then attached to the bin's internal mode list, enabling later stages of the heuristic to reason about memory feasibility, trade-offs, and target prioritization. The mission constants used in this computation are summarized below for clarity.

Mission and Mode Constants.

$$\text{HIGH\_MODE\_DATA\_RATE} = 4\,676\,840 \text{ bytes/s},$$

$$\text{MEDIUM\_MODE\_DATA\_RATE} = 2\,311\,680 \text{ bytes/s},$$

$$H_{0-\text{FACTOR}} = 0.005929439438291139.$$

Algorithm 3 Assignment of Priority and Modes to Bins

procedure BINSMEMORYASSIGNMENT(bin_list, xml_path)
    Parse the XML file at *xml_path* and obtain the root node
    Initialize an empty map: *target_map*
    for all target node in the XML **do**
        name ← target.name
        priority ← target.priority
        mode_list ← all mode entries under target.modes
        target_map[*name*] ← (*priority*, *mode_list*)
    end for
    for all *bin* in bin_list **do**
        if *bin.bin_type* is in *target_map* **then**
            bin.priority ← *target_map*[*bin.bin_type*].*priority*
            for all *m* in *target_map*[*bin.bin_type*].*mode_list* **do**
                if *m* is HIGH_MODE **then**
                    data_rate ← HIGH_MODE_DATA_RATE
                else if *m* is MEDIUM_MODE **then**
                    data_rate ← MEDIUM_MODE_DATA_RATE
                else
                    continue ▷ Unsupported mode
                end if
                memory ← MEMOCALCULATOR(*bin.range*, *data_rate*, *bin.duration*)
                append **Mode**(*m*, *memory*) to *bin.mode_list*
            end for
        else
            report that *bin.bin_type* is not found in the XML configuration
        end if
    end for
end procedure

Algorithm 4 Memory Usage Calculation for a Single Bin

procedure MEMOCALCULATOR(bin_range, mode_data_rate, duration)
    scale ← $10^{-6} \cdot \dfrac{463 \cdot 4}{395}$
    $H_0 \leftarrow \left(\dfrac{\text{bin\_range}}{H\_0\_FACTOR} \right) \cdot \text{scale}$
    $R_{\text{dep}} \leftarrow \dfrac{\text{mode\_data\_rate}}{H_0}$ ▷ Target-dependent data rate (bytes/s)
    return $R_{\text{dep}} \cdot \text{duration}$ ▷ Memory usage in bytes
end procedure

3.8 Visualization of the Initialization Output

To provide a clear and intuitive understanding of the preprocessing workflow, the full initialization sequence which is from raw orbital trajectory to the final bin representation is visualized across three figures. Rather than depicting only the spacecraft path, these figures illustrate the internal transformations applied to each target during the initialization phase. Together, they reveal how target geometry, discretization, and bin segmentation operate as a coherent pipeline that prepares data for the scheduling algorithm.

Figure 3.10 shows the first stage of the workflow, where the original target trajectories are plotted along the satellite ground track. At this stage, each target is represented as a continuous geospatial interval on the orbit, reflecting the raw input dataset.

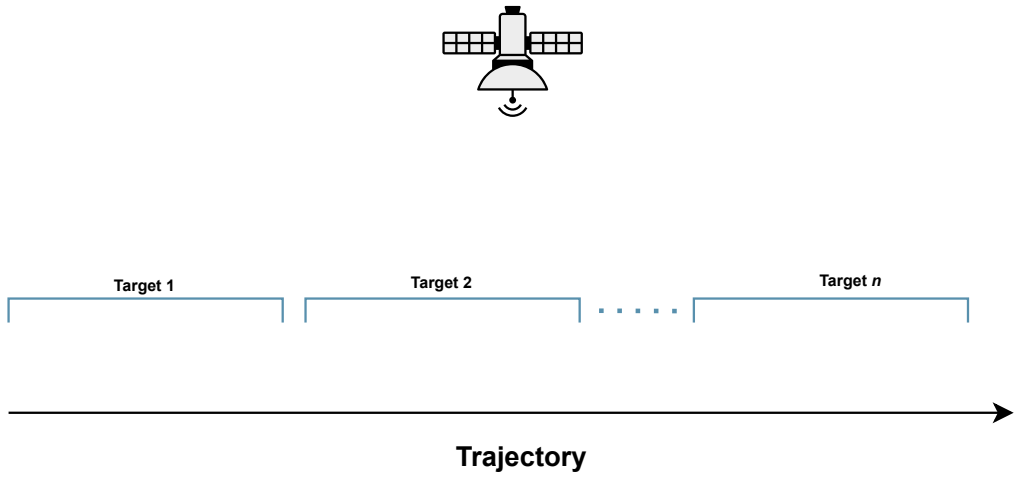


Figure 3.10: Example of trajectory initialization along the orbit.

Figure 3.11 presents the second stage, where each target is discretized into a series of uniformly distributed endpoints according to the predefined step size. These endpoints serve as the logical breakpoints that define the boundaries of the smallest units handled by our system. As shown in the visualization, the discretization step creates regularly spaced endpoints along each target segment.

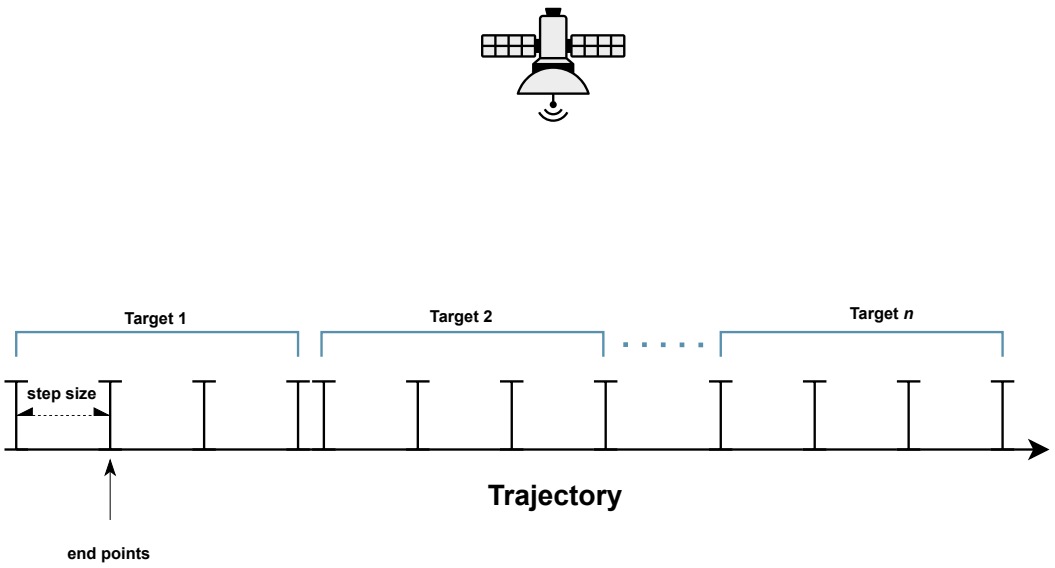


Figure 3.11: Example of endpoints initialization along the trajectory.

Finally, Figure 3.12 illustrates the third and most critical stage of the initialization process: the construction of bins. Each pair of adjacent endpoints is grouped into one bin, forming a structured sequence of fine-grained intervals for every target. The visualization shows how each target becomes a sequence of bins (Bin 1, Bin 2, Bin 3, ...), which together form a discretized representation of the raw trajectory.

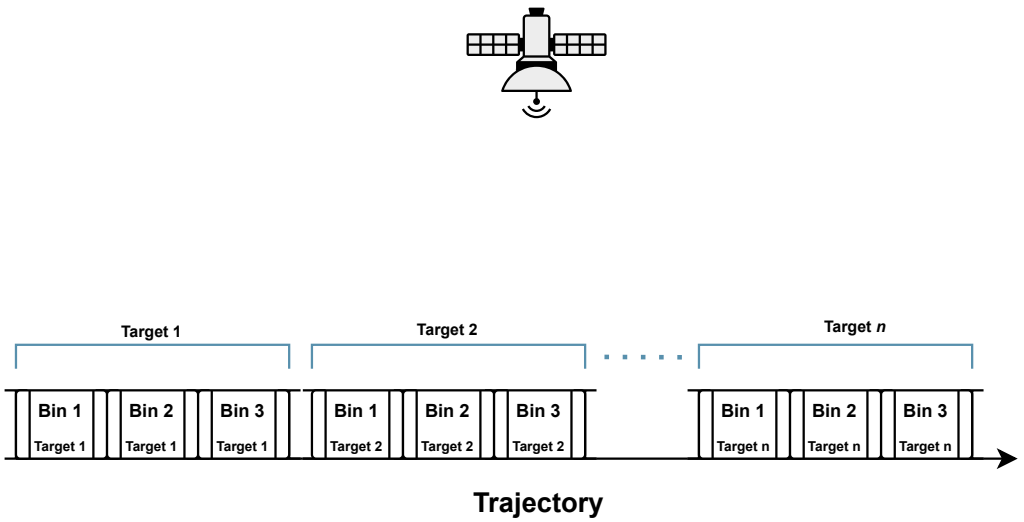


Figure 3.12: Example of bins along the trajectory.

This three-step workflow provides two important benefits. First, it guarantees that all targets—regardless of their geographical extent or shape—are represented in a uniform, fine-grained structure which provides flexibility of target implementation of memory and mode selection. Second, it allows the heuristic scheduler to reason at the bin level rather than at the original target level, resulting in more flexible decision-making, better memory control, and more precise prioritization across large-scale datasets. These visualizations therefore, not only validate the correctness of the initialization process but also highlight its foundational role in the overall optimization pipeline.

3.9 Summary

This chapter establishes the foundation for the upcoming heuristic optimization. By transforming raw target coordinates into structured bins enriched with operational metadata, the system achieves both transparency and adaptability. The modular design of the initialization code ensures that the algorithm remains easily interpretable for mission planners and can be

reconfigured for different orbital scenarios or satellite platforms.

CHAPTER 4

TARGET PRE-PROCESSING UNDER MEMORY CONSTRAINTS

Once the targets and the satellite trajectory have been discretized into bins, the next step of the heuristic is to decide which bins are worth acquiring under the onboard memory constraint. At this stage, we intentionally ignore the second operational constraint—the maximum number of commands—and focus exclusively on memory constraints, ensuring that the selected bins are memory-feasible on an orbit-by-orbit basis. This "targets pre-processing" phase produces a high-quality, memory-compliant set of bins that will later be refined when the number of command constraint is introduced.

Each bin is associated with a target identifier, an orbit index, a priority level, and a list of recording modes ordered from highest to lowest resolution. Every mode is characterized by a memory usage value, and initially, all bins are assumed to operate in the highest-resolution mode of user selection. The pre-processing algorithm proceeds in two main steps:

1. **Priority-driven target iteration.** All targets (standard and custom-interval targets) are extracted from the user-defined XML configuration together with their priorities. The list is sorted in descending order of priority. For each target in this order, the algorithm collects all bins belonging to that target and evaluates whether they can be safely acquired within the per-orbit memory limit.

2. **Intra-orbit mode adjustment.** If adding a target in its highest-resolution modes would exceed the memory budget in any given orbit, the algorithm selectively downgrades the modes of bins within that orbit (from HIGH to MEDIUM) while aiming to preserve data quality. This adjustment is performed using our proposed method, which requires no more than three commands for each target. Only bins belonging to

the current priority level are eligible for mode reduction in this step. The details of the proposed mode downgrade algorithm will be elaborated in the following chapter.

By iterating from the highest to the lowest priority and reducing modes only when strictly necessary, the pre-processing phase aims to maximize scientific value subject to the memory constraint, while preparing a feasible baseline for the following command-optimization phase.

4.1 Pessimistic Memory Check

Before introducing the full target pre-processing phase, we begin with several auxiliary procedures that substantially simplify and accelerate the main algorithm. The first of these is the *pessimistic memory check*, a lightweight feasibility test performed before attempting to select a candidate target. Notably, these checks never remove feasible assignments but only remove unfeasible assignments early.

Recall that a target is composed of multiple bins, each of which may be recorded in different user-specified modes. Since higher-resolution modes consume more onboard memory, it is necessary to determine, in advance, whether a target could ever be feasibly selected under the satellite memory constraints. Algorithm 5 carries out this assessment by calculating the *worst-case (lowest mode)* memory usage associated with the target and comparing it to the remaining memory per orbit.

For the orbit under consideration, the algorithm mainly focus the followings:

- All bins already selected from previously accepted targets along this orbit;
- All bins belonging to the candidate target.

Among these bins, it further distinguishes:

- *same-priority bins*: bins whose priority matches that of the candidate target;
- *different-priority bins*: bins whose priorities are strictly higher or lower.

The pessimistic check then applies the following assumptions:

- All same-priority bins (including those of the candidate target) will operate in their *lowest* recording mode available to that priority level;
- All different-priority bins are fixed in their current modes, i.e., no additional mode reductions are assumed to be possible for them.

Let M_{same} denote the total memory that would be consumed by all same-priority bins if recorded in their lowest modes. Let M_{diff} denote the memory consumed by all different-priority bins in their current modes. For each orbit, the pessimistic feasibility condition is expressed as:

$$M_{\text{same}} \leq M_{\text{max}} - M_{\text{diff}},$$

where M_{max} is the maximum allowable memory capacity per orbit.

If this inequality is violated for any orbit, the candidate target is rejected immediately, and the algorithm proceeds to the next target without modifying the current acquisition set. Conversely, if the inequality holds for all orbits intersected by the target, then the target passes the pessimistic check and is deemed eligible for inclusion in later stages, where more refined intra-orbit and inter-orbit mode-adjustment strategies may still be applied.

This pessimistic memory check serves two critical purposes. First, it dramatically increases computational efficiency (by over 20%, in our experiments) by filtering out targets that are guaranteed to be infeasible, thereby avoiding unnecessary mode-adjustment attempts and memory calculations depending on the targets' merging conditions.

Second, it acts as a conservative safety mechanism: by ensuring that even in the least favorable mode configuration the memory constraint cannot be violated, it prevents any target from being accepted that could later cause orbit-level memory overflow. This guarantees that the subsequent heuristic decisions operate only on targets that are feasible under strict worst-case assumptions, improving both robustness and transparency of the scheduling

process.

Algorithm 5 Pessimistic Memory Check

procedure PessimisticMemoryCheck($\mathcal{B}_{\text{cur}}, \mathcal{A}, o_{\text{cur}}, p_{\text{cur}}$)
  $\mathcal{A}' \leftarrow$ deep copy of $\mathcal{A}$
  $\mathcal{B}'_{\text{cur}} \leftarrow$ deep copy of $\mathcal{B}_{\text{cur}}$
  $\mathcal{B}_{\text{temp}} \leftarrow \mathcal{A}' \cup \mathcal{B}'_{\text{cur}}$
  $\mathcal{B}_{\text{diff}} \leftarrow \{\, b \in \mathcal{B}_{\text{temp}} \mid b.\text{orbit} = o_{\text{cur}},\ b.\text{priority} \neq p_{\text{cur}} \,\}$
  $\mathcal{B}_{\text{same}} \leftarrow \{\, b \in \mathcal{B}_{\text{temp}} \mid b.\text{orbit} = o_{\text{cur}},\ b.\text{priority} = p_{\text{cur}} \,\}$
  $M_{\text{same}} \leftarrow \sum_{b \in \mathcal{B}_{\text{same}}} b.\text{mode\_list}[\text{HIGH}].\text{memory}$
  $M_{\text{diff}} \leftarrow \sum_{b \in \mathcal{B}_{\text{diff}}} b.\text{mode\_list}[b.\text{current\_mode\_index}].\text{memory}$
  if $M_{\text{same}} \leq M_{\max} - M_{\text{diff}}$ **then**
    return True
  else
    return False
  end if
end procedure

4.2 Costs of Commands in Targets

When attempting to select a target, the algorithm may find that adding all of its bins in the highest recording mode would overflow the available memory in the current orbit. In this situation, we must reduce the modes of some bins from *High* to *Medium* in order to stay within the memory limit.

However, there is an important detail to consider. Because the smallest operational unit is the bin, we do not downgrade the entire target as a whole; instead, we downgrade individual bins. As a result, after mode reduction, a target may end up operating in a *hybrid* configuration, meaning that some bins remain in High mode while others are downgraded to Medium mode. If the user provides both mode options for the target, such hybrid configurations are possible.

Figure 4.1 provides a simplified illustration of several common hybrid cases:

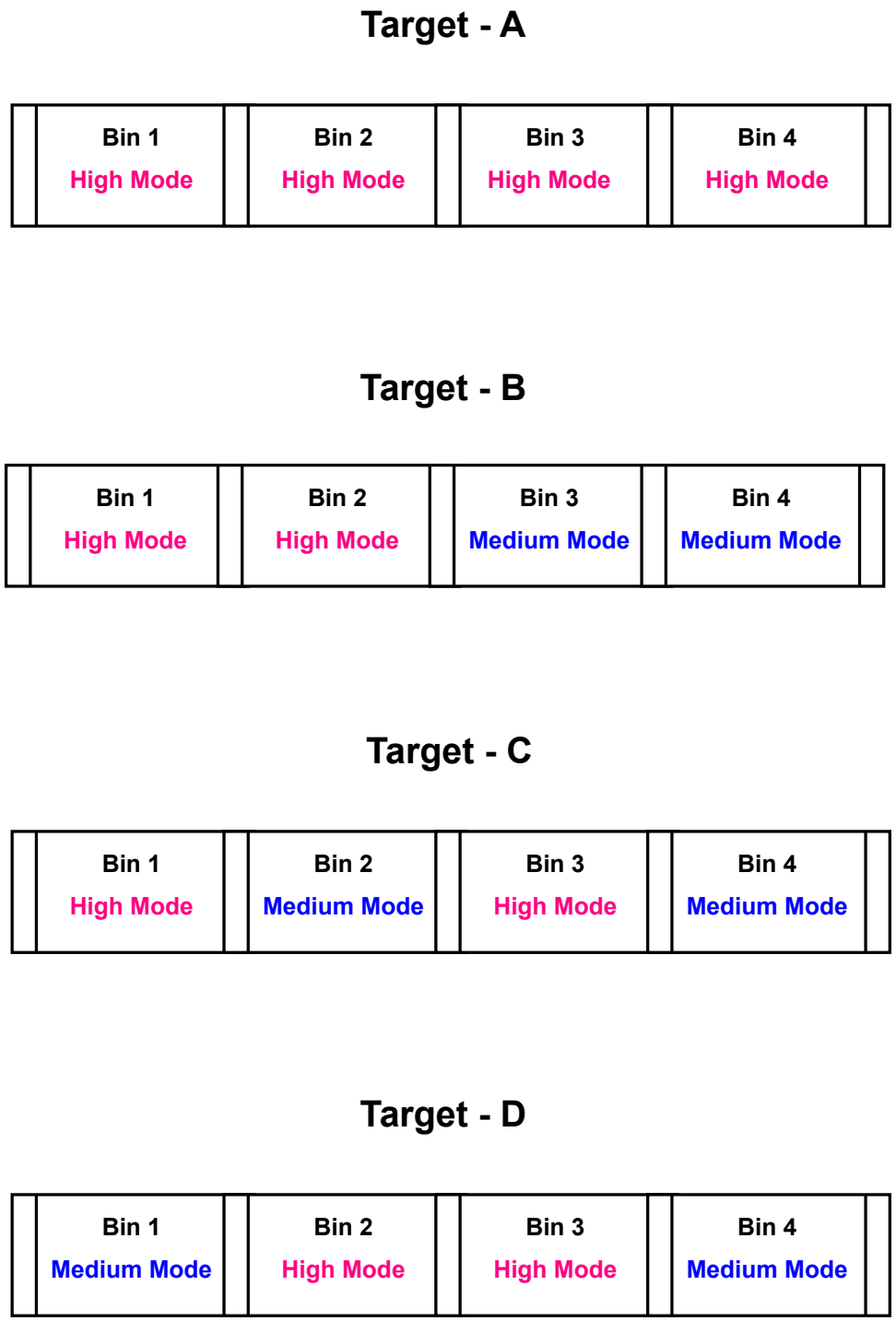


Figure 4.1: Example of Hybrid Bins in the Target

From the examples above, we can observe three typical patterns. Although this part of the algorithm focuses only on memory constraints, it is still useful to briefly mention the impact on the *number of commands* required to acquire a target. The command cost depends on how many times the satellite must switch between modes while recording the bins.

For instance:

- **Target A**: all bins are in High mode. Since the modes are uniform, the command cost is simply the basic cost of issuing the start and stop commands, which is 2.
- **Target B**: bins 1 and 2 are High mode, and bins 3 and 4 are Medium mode. There is one mode change (between bins 2 and 3), so the total command cost becomes 3.
- **Target C**: the modes alternate several times (High–Medium–High–Medium), resulting in three separate mode changes. This leads to a higher total command cost of 5.
- **Target D**: this case contains two mode shifts, giving a total command cost of 4.

The main idea here is that, even though we delay the official command-constraint processing to a later stage of the algorithm, it is still helpful to be aware that different hybrid configurations can lead to very different command costs. In general, we prefer configurations with fewer mode changes because they are more efficient and simpler for the satellite to execute. Among the examples above, Target C represents the least desirable case due to its multiple mode shifts, while Target B is more ideal: it requires only a single change of mode, yet it has the same number of High and Medium bins as Target C. In other words, targets B and C are the same target, then strategy of target B is much preferred since it costs only one extra command to select the target which has approximately same memory usage of the strategy used in target C.

4.3 Bin Mode Reduction

As discussed in the previous section, acquiring a target in its highest recording modes may lead to memory overflow on certain orbits. To handle this, we need an automated procedure that decides which bins should have their modes reduced from `HIGH` to `MEDIUM` while keeping the number of mode changes small. In real operational scenarios this becomes more complicated than simply reducing the current target's bins: in practice, we are also allowed to reduce the modes of *other* targets that share the same user-assigned priority and lie on the same orbit.

A second consideration is that, when multiple bins are eligible for downgrading, it is usually preferable to reduce the bin whose memory savings (from HIGH to MEDIUM) are the smallest. This helps preserve the overall high-resolution coverage for the target set.

With these ideas in mind, suppose that a target has passed the pessimistic memory check but still cannot be added in its full HIGH-mode configuration due to the actual per-orbit memory usage. In this case, the algorithm triggers a local mode-reduction routine. This routine reduces selected bins from HIGH to MEDIUM while following two simple guiding principles:

1. **Same-priority reduction scope.** Mode reductions are allowed on *any* bin within the same orbit whose priority matches the current target's priority—not only bins belonging to the target we are trying to add.

2. **Minimal loss per reduction.** Among all legally reducible bins, the algorithm always chooses the bin whose HIGH to MEDIUM downgrade produces the *smallest* drop in memory usage. This greedy step helps preserve as much high-resolution data as possible.

Operationally, Algorithm 6 groups all same-priority bins on the orbit, sorts them along the along-track direction, and then applies a simple edge-based rule to maintain modes

coherence:

- If all bins across the entire same-priority group (including bins from previously accepted targets and the current target) are in `HIGH` mode, only the first or last bin along the along-track sequence may be reduced. Between these two edge bins, the algorithm chooses the one whose HIGH to MEDIUM downgrade produces the smallest decrease in memory usage.

- If the leftmost bin in the target group is already in `MEDIUM` mode, then only the first `HIGH`-mode bin immediately to its right may be reduced.

- If the rightmost bin is in `MEDIUM` mode, then only the last `HIGH`-mode bin immediately to its left may be reduced.

After each reduction, the algorithm recomputes the total memory usage on that orbit. Mode reductions continue until one of the following conditions is reached:

- The total memory falls within the allowed per-orbit budget, or

- No further legal reductions exist.

If the second case occurs, the configuration is considered infeasible and the target is rejected for that orbit. Otherwise, the target is accepted together with the updated modes of all same-priority bins.

The outcome of the bin mode-reduction stage is a set of bins—across both the current target and previously select same-priority targets—whose modes have been adjusted just enough to satisfy the per-orbit memory limits while keeping high-resolution coverage whenever possible.

Algorithm 6 Mode Reduction in Bins

procedure DecreaseModesInBins($\mathcal{B}, M_{\text{threshold}}$)
  function TotalMemory($\mathcal{S}$)
    return $\sum_{b \in \mathcal{S}} b$.mode_list[$b$.current_mode_index].memory
  end function
  function FindReducibleInGroup($\mathcal{G}$)
    if $\mathcal{G}$ is empty **then**
      return $\emptyset$
    end if
    $\mathcal{H} \leftarrow \{\, b \in \mathcal{G} \mid b$.current_mode_index = HIGH $\}$
    if $\mathcal{H} = \emptyset$ **then**
      return $\emptyset$ ▷ all MEDIUM already
    else if $|\mathcal{H}| = |\mathcal{G}|$ **then**
      return $\{\mathcal{G}[1], \mathcal{G}[\text{end}]\}$ ▷ all HIGH: only edges
    else if $\mathcal{G}[1]$.current_mode_index $\neq$ HIGH **then**
      return first $b \in \mathcal{G}$ with HIGH mode
    else if $\mathcal{G}[\text{end}]$.current_mode_index $\neq$ HIGH **then**
      return last $b \in \mathcal{G}$ with HIGH mode
    else
      return $\emptyset$ ▷ no legal reduction
    end if
  end function
  while TotalMemory($\mathcal{B}$) $> M_{\text{threshold}}$ **do**
    $\mathcal{C} \leftarrow [\,]$ ▷ candidate bins
    for all bin types τ present in $\mathcal{B}$ **do**
      $\mathcal{G} \leftarrow \{\, b \in \mathcal{B} \mid b$.type $= \tau \,\}$
      Sort $\mathcal{G}$ by along-track index
      $\mathcal{R} \leftarrow$ FindReducibleInGroup($\mathcal{G}$)
      for all $b \in \mathcal{R}$ **do**
        $m_{\text{H}} \leftarrow b$.mode_list[HIGH].memory
        $m_{\text{M}} \leftarrow b$.mode_list[MEDIUM].memory
        $\Delta m \leftarrow m_{\text{H}} - m_{\text{M}}$
        Append $(\Delta m, b)$ to $\mathcal{C}$
      end for
    end for
    if $\mathcal{C}$ is empty **then**
      Print "No more legal reductions possible."
      exit with error (infeasible configuration)
    end if
    Sort $\mathcal{C}$ by Δm (ascending)
    $(\Delta m^*, b^*) \leftarrow \mathcal{C}[1]$
    b^*.current_mode_index $\leftarrow$ MEDIUM
  end while
end procedure

4.4 Bin Mode Reduction – Example

To illustrate how the bin-mode-reduction procedure works, we present a small example involving two targets, labeled Target A and Target B. Suppose that Target A is the newly encountered target we wish to select, while Target B has already been recorded in memory with all of its bins assigned in `HIGH` mode. Both targets share the same priority level and belong to the same orbit. For simplicity, we assume that each target contains four bins, and that both `HIGH` and `MEDIUM` modes are available. We also assume that only these two targets have been encountered so far, and the total orbital memory usage resulting from their current modes is shown in the figures below. The number displayed in the top-left corner represents the orbital memory constraint. If the total memory usage exceeds this value, the acquisition is considered to overflow the memory, which must be avoided.

Figure 4.2 illustrates the initial state (step 1). The orbital memory constraint is 50 MB, and Target A currently assigns `HIGH` mode to all four bins. Following the algorithm, we first run the pessimistic memory check. This means temporarily downgrading all bins to their lowest available mode and computing the total memory required. The pessimistic sum is 26 MB, which is below the 50 MB constraint. Therefore, we know that it is always possible to select the target eventually, even if we need to reduce the modes.

Next, we begin searching for bins whose modes may be downgraded while minimizing data loss. We call these *candidate bins*. At this point, the candidates are A-1, A-4, B-1, and B-4. Only the leading bins of each target are considered, as explained earlier, because the method enforces an upper limit on the number of commands per target.

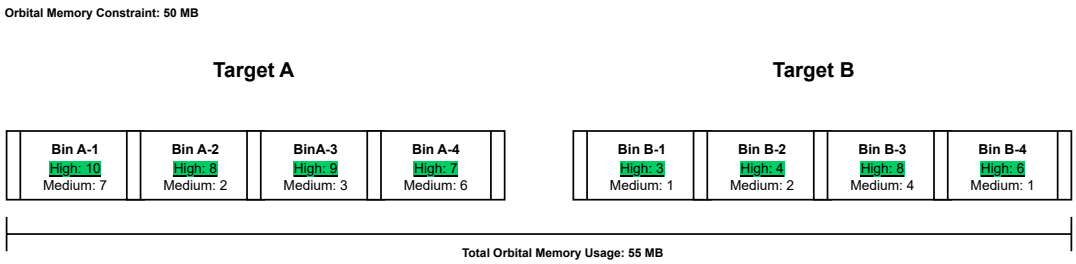


Figure 4.2: Initial configuration before mode reduction.

In step 2, shown in Figure 4.3, the algorithm identifies Bin A-4 as the first bin to downgrade because it has the smallest mode-shift drop (1 MB) among all candidate bins. After downgrading A-4 to `MEDIUM`, we recompute the total memory usage and obtain 54 MB, which still exceeds the 50 MB constraint.

Because one candidate (A-4) has been used, the candidate set updates to A-3, B-1, and B-4. Notice that A-1 is no longer a candidate. Downgrading A-1 at this point would cause the target to require four separate commands, violating the command-limit rule described earlier.

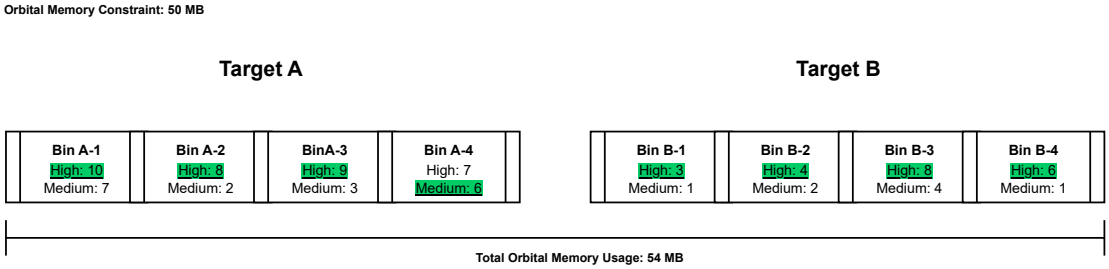


Figure 4.3: After downgrading the first candidate bin (A-4).

As illustrated in Figure 4.4, we continue selecting the bin with the smallest drop among the current candidates. Bin B-1 has the next smallest drop (2 MB), so we downgrade it to `MEDIUM`. Rechecking the memory usage gives 52 MB, which still exceeds the constraint. The candidates now become B-2 and A-3, so the process continues.

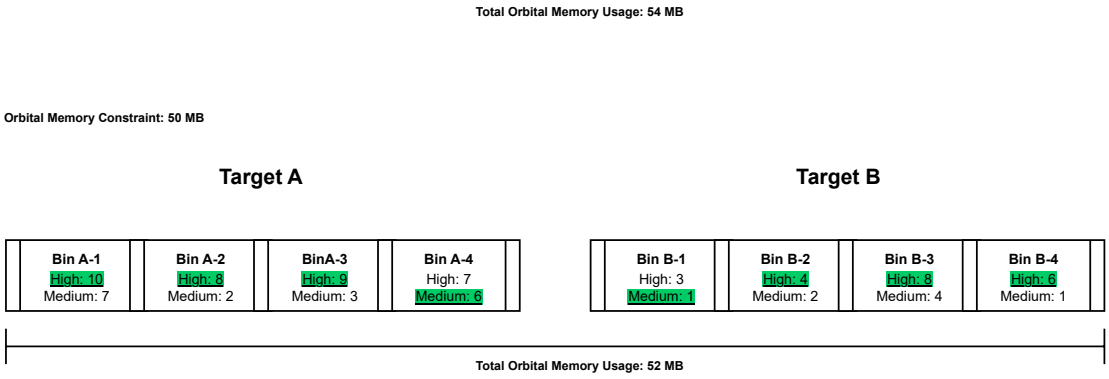


Figure 4.4: Continuing the reduction with the next smallest mode shift drop.

Finally, as shown in Figure 4.5, the next choice is between B-2 and A-3. Bin B-2 has a mode-shift drop of 2 MB, compared with the larger 6 MB drop required for A-3. Therefore, we downgrade B-2. The total memory usage now becomes exactly 50 MB, satisfying the constraint. At this point, the process stops. Only a small number of bins were downgraded, and we maintained as much data quality as possible without reducing all bins across both targets.

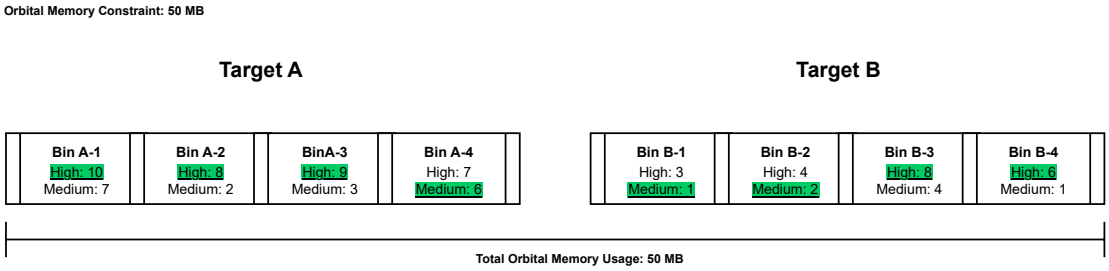


Figure 4.5: Final result after sufficient mode downgrades to satisfy the memory constraint.

4.5 Target Pre-Process Explanation

With the initialization phase completed—where targets are constructed, trajectories are discretized, and all bins are generated—we now proceed to the second major component of the proposed heuristic: *target pre-processing*. This phase focuses on handling the first of the

two mission constraints, namely the onboard memory limitation. As discussed earlier, we temporarily ignore the command-count constraint and prioritize ensuring that the memory limit is respected before any higher-level scheduling decisions are made.

The goal of this stage is to iterate through all targets, ordered by their priority, and determine whether each target can be selected under the current memory configuration. If the target cannot fit in memory in its initial configuration, the algorithm attempts to reduce the modes of relevant bins using the reduction strategy described in earlier sections. This ensures that we preserve as much data quality as possible while still keeping the overall memory usage below the orbital constraint.

Algorithm 7 below summarizes the entire workflow. At a high level, the target pre-processing procedure performs the following operations:

- It loads all bins associated with the targets and filters out bins of type `NORMAL`, since only target-related bins are relevant for acquisition decisions.
- It sorts all remaining bins by their priority so that the algorithm processes higher-value targets first.
- For each target, it performs a *pessimistic memory check*: all bins belonging to the target are temporarily downgraded to their lowest available recording mode. If even this conservative estimate exceeds the memory constraint, the target cannot be select under any circumstance and is skipped.
- If the pessimistic check passes, the algorithm evaluates whether the target—in its current mode configuration—fits in memory. If yes, the target is selected without modification.
- If not, the algorithm identifies all bins that share the same priority and orbit, and performs mode reduction until their combined memory usage falls below the threshold reserved for this priority level.

- After mode reduction (if required), the target is accepted and added to the list of selected bins.

This way, the pre-processing stage ensures that the system commits only to targets that can be feasibly stored within onboard memory, while also coordinating reductions so that data quality is preserved whenever possible. This stage sets up a stable and constraint-respecting foundation for the later phases of the heuristic algorithm.

The complete pseudo-code is provided below, detailing each step of this process.

Algorithm 7 Target Pre-Processing Under Memory Constraints

procedure PREPROCESSTARGETS($\mathcal{B}$)
  Write header to action log file
  $\mathcal{T} \leftarrow$ GETALLTARGETSWITHPRIORITY
  Sort $\mathcal{T}$ in descending order of priority
  $\mathcal{U} \leftarrow$ deep copy of $\mathcal{B}$
  Remove all bins from $\mathcal{U}$ whose type is NORMAL
  $\mathcal{U} \leftarrow$ SORTBINSBYPRIORITY($\mathcal{U}$)
  $\mathcal{A} \leftarrow [\,]$ ▷ selected (processed) bins
  for all $(t_{\text{name}}, p_{\text{cur}}) \in \mathcal{T}$ **do**
    $o_{\text{cur}} \leftarrow$ GETORBITBYNAME(t_{name})
    $\mathcal{B}_{\text{cur}} \leftarrow \{\, b \in \mathcal{U} \mid b.\text{type} = t_{\text{name}} \,\}$
    if not PESSIMISTICMEMORYCHECK($\mathcal{B}_{\text{cur}}, \mathcal{A}, o_{\text{cur}}, p_{\text{cur}}$) **then**
      Write to log: target t_{name} failed pessimistic check
      continue ▷ skip this target
    end if
    Write to log: target t_{name} passed pessimistic check
    $\mathcal{B}_{\text{orbit}} \leftarrow \{\, b \in \mathcal{B}_{\text{cur}} \cup \mathcal{A} \mid b.\text{orbit} = o_{\text{cur}} \,\}$
    $M_{\text{total}} \leftarrow \sum_{b \in \mathcal{B}_{\text{orbit}}} b.\text{mode\_list}[b.\text{current\_mode\_index}].\text{memory}$
    if $M_{\text{total}} \leq M_{\max}$ **then**
      Append $\mathcal{B}_{\text{cur}}$ to $\mathcal{A}$
      Write to log: target t_{name} fits without reduction
    else
      Write to log: target t_{name} requires mode reduction
      $\mathcal{B}_{\text{same}} \leftarrow \{\, b \in \mathcal{A} \cup \mathcal{B}_{\text{cur}} \mid b.\text{priority} = p_{\text{cur}},\ b.\text{orbit} = o_{\text{cur}} \,\}$
      $M_{\text{diff}} \leftarrow \sum_{\substack{b \in \mathcal{A} \cup \mathcal{B}_{\text{cur}} \\ b.\text{priority} \neq p_{\text{cur}},\ b.\text{orbit} = o_{\text{cur}}}} b.\text{mode\_list}[b.\text{current\_mode\_index}].\text{memory}$
      $M_{\text{threshold}} \leftarrow M_{\max} - M_{\text{diff}}$
      DECREASEMODESINBINS($\mathcal{B}_{\text{same}}, M_{\text{threshold}}$)
      Append $\mathcal{B}_{\text{cur}}$ to $\mathcal{A}$
      Write to log: bins selected after mode reduction
    end if
  end for
  return $\mathcal{A}$
end procedure

CHAPTER 5

TARGETS ACQUISITION OF COMMANDS CONSTRAINTS

Once the memory-constrained pre-processing phase has been completed, each candidate bin has already been assigned an operational mode that satisfies the onboard memory limitations while preserving as much scientific value as possible. In this final phase, we must additionally enforce the second mission constraint: the maximum number of recording commands that may be issued within a cycle. Intuitively, each contiguous recording session along the ground track is interpreted as a command. As the number of candidate targets grows, the total command usage required to acquire all of them may easily exceed the onboard limit. To address this, the proposed heuristic processes the targets in order of priority and progressively builds a feasible acquisition plan. For each target, the algorithm first checks whether it can be acquired without violating the remaining command budget. When this is not possible, the algorithm attempts to *merge* the current target with an already acquired, same-orbit neighbor, potentially reducing the total number of required commands.

Merging targets introduces additional memory considerations: the satellite must keep recording while slewing between the two target footprints, which unintentionally acquires intermediate land surfaces and any lower priority targets in between. These intermediate segments must share a common recording mode that is compatible with the boundary bins of both targets, and their cumulative memory usage must fit within the remaining orbit memory.

Whenever a merge is successful, the algorithm replaces the two original targets by a single merged target and records all unintentionally acquired targets as "intermediate acquisitions" that will be handled in a later clean-up step.

The remainder of this section formalizes this decision process. We first describe how individual bins are grouped into `Target` objects, then present the main command-constrained acquisition algorithm and its supporting procedures for nearest-target search, merge feasi-

bility checking, and target merging.

5.1 Selected Target Initialization

To enable command-level reasoning and merging operations, the planner first converts the flat list of selected bins into a structured list of `Target` objects. As shown in Algorithm 8, all bins are grouped by their globally unique `bin_type`, where each group corresponds to a single target footprint. The bins within each target are then sorted by their intra-target index to preserve the along-track acquisition order. Target-level attributes such as orbit number and priority are inherited from the first bin in the group, while the target endpoints are defined by the first bin's fore endpoint and the last bin's rear endpoint. The initialization also determines whether a target is mode-hybrid by checking if multiple mode names appear across its bins, and attaches orbit-specific metadata (e.g., the target orbital index and PSO timing descriptors) required by later stages such as nearest-neighbor search and merge feasibility assessment. The resulting `Target` list provides a compact representation for subsequent command-constrained acquisition decisions.

Algorithm 8 Target Initialization

procedure InitializeTargets($\mathcal{B}$)
  $\mathcal{G} \leftarrow$ GroupByType($\mathcal{B}$) ▷ `bin_type` $\mapsto$ bins
  $\mathcal{I} \leftarrow$ GetTargetOrbitalIndexDict
  $\mathcal{T} \leftarrow [\,]$
  for all `type` in sorted keys of $\mathcal{G}$ **do**
    $\mathcal{B}_t \leftarrow$ SortByIndex($\mathcal{G}$[`type`])
    $o \leftarrow \mathcal{B}_t[0]$.`orbit`, $p \leftarrow \mathcal{B}_t[0]$.`priority`
    $fore \leftarrow \mathcal{B}_t[0]$.`fore_endpoint`, $rear \leftarrow \mathcal{B}_t[-1]$.`rear_endpoint`
    $is\_hybrid \leftarrow$ HasMultipleModes($\mathcal{B}_t$)
    $idx \leftarrow \mathcal{I}$[`type`]
    $(pso_{start}, pso_{dur}) \leftarrow$ GetPSOMetadata(`type`)
    $T \leftarrow$ BuildTarget(`type`, $\mathcal{B}_t$, o, p, $fore$, $rear$, is_hybrid, idx, pso_{start}, pso_{dur})
    Append T to $\mathcal{T}$
  end for
  return $\mathcal{T}$
end procedure

5.2 How to Find the Nearest Target

When independent acquisition of a target is not feasible and a merging attempt is required, the algorithm must determine which previously acquired target provides the most suitable merging opportunity. Since merging is only allowed between targets located on the same orbit, the algorithm first identifies all acquired targets that share the same orbital track as the current target T_{cur}.

Here, the *current target* T_{cur} refers to the target that is currently under evaluation by the algorithm, for which the acquisition decision has not yet been finalized. In other words, it is the target being processed at this stage, which may ultimately be independently acquired, merged with another target, or discarded depending on feasibility.

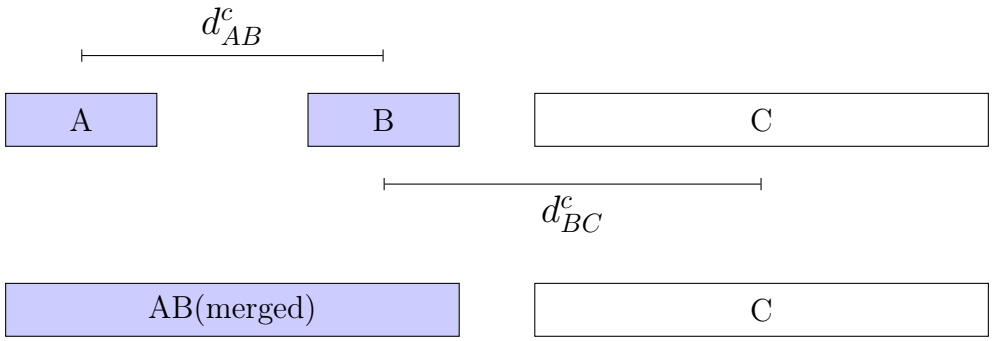
A key design choice in this step is how the distance between targets is defined. Two natural interval distance measures are illustrated in Figure 5.1. The first is a *center-based* distance, which computes the separation between the geometric centers of two targets. As shown in Figure 5.1(a), this approach can lead to suboptimal merging decisions when multiple targets are involved. In this example, the algorithm would merge target B with target A because the center-to-center distance d^c_{AB} is smaller than d^c_{BC}, even though target C is actually closer to B along the ground track.

To address this limitation, we instead adopt a *boundary-based* distance definition, illustrated in Figure 5.1(b). This approach measures the distance between the rear boundary endpoint of a preceding target and the fore boundary endpoint of a subsequent target. Under this definition, target B correctly merges with target C, as the boundary distance d^b_{BC} is smaller than d^b_{AB}. This choice better reflects the true recording cost of merging targets and avoids unnecessary acquisition of intermediate data, thereby reducing excess memory usage.

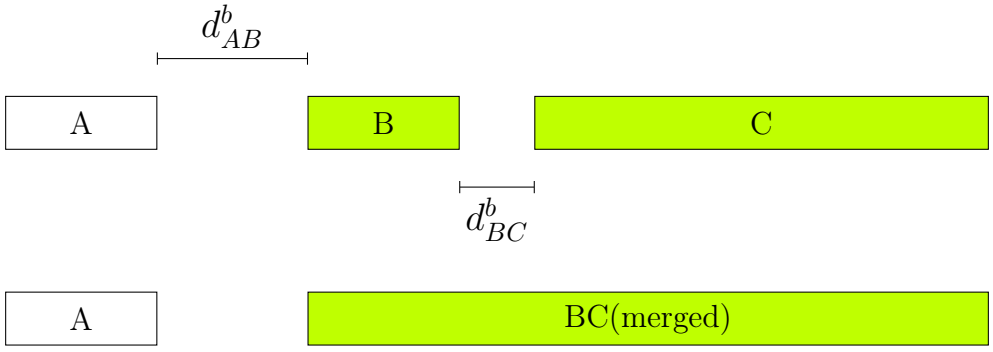
Based on this boundary-based distance, the nearest target is selected by examining the relative ordering of target footprints along the orbit. If the current target lies between two acquired targets, the algorithm compares the preceding and subsequent neighbors and selects

the one with the smaller boundary-to-boundary distance. If only one same-orbit neighbor exists, that target is selected by default. The identified nearest target then serves as the reference object for the merge feasibility assessment described in the following section.

Algorithm 9 summarizes the procedure used to identify the nearest previously acquired target on the same orbit.



(a) Center-based coordinates.



(b) Boundary-based coordinates.

Figure 5.1: Difference between center-based and boundary-based coordinates systems.

Algorithm 9 Find the Nearest Acquired Target on the Same Orbit

procedure FINDNEARESTTARGET($\mathcal{A}$, T_{cur})
    $\mathcal{S} \leftarrow \{\, T \in \mathcal{A} \mid T.\texttt{orbit\_number} = T_{\text{cur}}.\texttt{orbit\_number} \,\}$
    if $\mathcal{S} = \emptyset$ **then**
        return None
    end if
    $\mathcal{P} \leftarrow \{\, T \in \mathcal{S} \mid T.\texttt{target\_orbital\_index} < T_{\text{cur}}.\texttt{target\_orbital\_index} \,\}$ ▷ preceding targets
    $\mathcal{Q} \leftarrow \{\, T \in \mathcal{S} \mid T.\texttt{target\_orbital\_index} > T_{\text{cur}}.\texttt{target\_orbital\_index} \,\}$ ▷ subsequent targets
    if $\mathcal{P} = \emptyset$ **and** $\mathcal{Q} \neq \emptyset$ **then**
        return first element of $\mathcal{Q}$
    else if $\mathcal{P} \neq \emptyset$ **and** $\mathcal{Q} = \emptyset$ **then**
        return last element of $\mathcal{P}$
    else
        $T_{\text{pre}} \leftarrow$ last element of $\mathcal{P}$
        $T_{\text{sub}} \leftarrow$ first element of $\mathcal{Q}$
        $d_{\text{pre}} \leftarrow$ DISTANCE($T_{\text{cur}}.\texttt{fore\_coordinate}$, $T_{\text{pre}}.\texttt{rear\_coordinate}$)
        $d_{\text{sub}} \leftarrow$ DISTANCE($T_{\text{cur}}.\texttt{rear\_coordinate}$, $T_{\text{sub}}.\texttt{fore\_coordinate}$)
        if $d_{\text{pre}} < d_{\text{sub}}$ **then**
            return T_{pre}
        else
            return T_{sub}
        end if
    end if
end procedure

5.3 Merge Feasibility Assessment

Once the nearest same-orbit target has been identified, the algorithm must determine whether the current target, which is currently under evaluation and not yet acquired not like all other acquired targets, can be feasibly merged with it under the remaining mission constraints.

This decision is non-trivial, as merging alters both the spatial extent and the recording configuration of the acquisition, and may introduce additional memory consumption or command overhead.

The merge feasibility assessment therefore evaluates the candidate merge from two complementary perspectives. First, it verifies *mode compatibility* between the boundary bins of the two targets, ensuring that the intervening interval can be recorded continuously without requiring unsupported mode switching.

Figure 5.2 illustrates four representative cases that capture the most common situations encountered during the merging process between the current target (Target B) and the nearest previously acquired target (Target A). In Cases 1 and 2, Target B is a *singular-mode* target, meaning that all of its bins share the same acquisition mode (medium mode in this example). In contrast, Target A is hybrid in Case 1 and singular with all bins in high mode in Case 2. In both cases, command availability is not a limiting factor, since the satellite can continue recording without interruption or mode switching when transitioning from Target A to Target B. As a result, no additional commands are required to acquire Target B in these scenarios.

However, mode compatibility remains a necessary condition. In Case 2, the boundary bins (A–4 and B–1) do not share a common acquisition mode. Enforcing a continuous recording would therefore require one additional commands, violating the merging assumption and leading to rejection of this case if it does not have enough commands left.

Cases 3 and 4 represent scenarios in which the current target (Target B) follows a *hybrid-mode* setting. In such cases, at least one additional command is required to perform an

internal mode switch during the acquisition of Target B. For example, in Case 3, the satellite continues recording Target A in high mode and proceeds into Target B until bin B–3, where a mode switch from medium to high is required. This switch needs one additional command, after which recording continues in high mode until the end of Target B. A similar command overhead occurs in Case 4, further illustrating that hybrid targets necessarily consume one additional command during acquisition.

These cases highlight why command availability must be explicitly considered during the merge feasibility assessment and motivate the upper-bound constraints introduced during the pre-processing phase.

Figure 5.2: Merging scenarios illustrating mode compatibility and command consumption.

Second, the algorithm evaluates the *memory feasibility* of the merge by estimating the additional memory required to record the intermediate interval and verifying that the combined memory usage remains within the remaining orbital memory budget.

As illustrated in Figure 5.3, merging introduces an additional recording interval between the nearest previously acquired target (Target A) and the current target (Target B). Once the boundary-mode compatibility condition is satisfied, this intermediate interval is discretized into bins in the same manner as regular targets. This design allows the algorithm to estimate memory consumption at the finest operational granularity, thereby preserving the integrity of the smallest acquisition unit. More importantly, these interval bins are retained and incorporated into the merged target representation when the merge is feasible, which ensures that consistency between feasibility assessment and subsequent target construction.

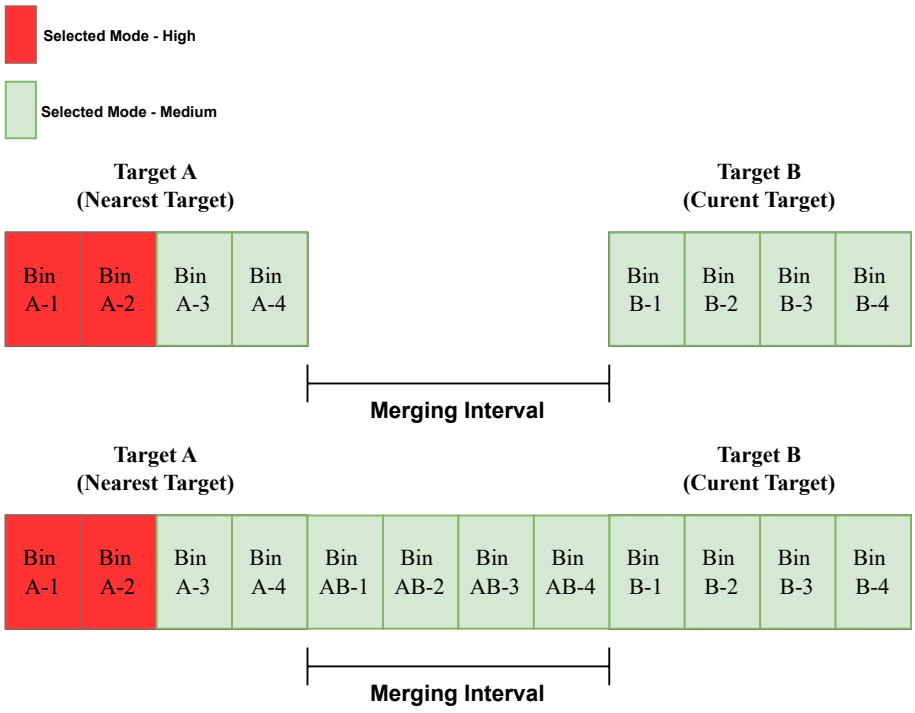


Figure 5.3: Merging Interval during the targets merging process.

If either condition is violated, the merge attempt is rejected and the current target is skipped to preserve overall constraint compliance. Otherwise, the two targets are merged into a single composite target with updated spatial and temporal attributes, which then replaces the original target in the acquired target list. This feasibility check enables the algorithm

to recover acquisition opportunities under command scarcity while avoiding unnecessary memory overconsumption.

Algorithm 10 Merge Feasibility Assessment (`IsMergeable`)

procedure IsMergeable(T_{cur}, T_{near}, C_{rem}, M_{rem})
  (has_common, $mode_{\text{bd}}$) ← CommonBoundaryMode(T_{cur}, T_{near})
  if not has_common **then**
    return (False, None)
  end if
  if T_{cur}.`is_mode_hybrid` = True **and** $C_{\text{rem}} < 1$ **then**
    return (False, None)
  end if
  $\mathcal{E}$ ← InitializeIntervalEndpoints(T_{cur}, T_{near})
  $\mathcal{B}_{\text{int}}$ ← InitializeIntervalBins($\mathcal{E}$, T_{cur}, T_{near})
  AssignIntervalMode($\mathcal{B}_{\text{int}}$, $mode_{\text{bd}}$)
  M_{int} ← IntervalMemoryUsage($\mathcal{B}_{\text{int}}$)
  if $M_{\text{int}} + T_{\text{cur}}$.`memory_usage` $> M_{\text{rem}}$ **then**
    return (False, None)
  end if
  return (True, $\mathcal{B}_{\text{int}}$)
end procedure

5.4 Merge Nearest Target

After a candidate merge passes the feasibility assessment (Algorithm 10), the algorithm proceeds to explicitly construct a new composite target by merging the current target T_{cur} with its nearest same-orbit neighbor T_{near}. The objective of this stage is to replace two separate acquisition segments with a single continuous recording segment, thereby reducing the total command usage while preserving the original target ordering along the orbit.

As illustrated in Figure 5.4, the merging operation concatenates three bin sequences in along-track order: the bin list of the leading target, which is Target A in the Figure 5.2, the discretized interval bins returned by the feasibility check, and the bin list of the trailing target, which is the target B in the Figure 5.4. This concatenation yields a unified bin list that represents the full continuous acquisition from the fore endpoint of the first target to the

rear endpoint of the second target. To maintain internal consistency, all bins in the merged list are re-indexed sequentially, and the merged target updates its endpoints to match the first bin's fore endpoint and the last bin's rear endpoint.

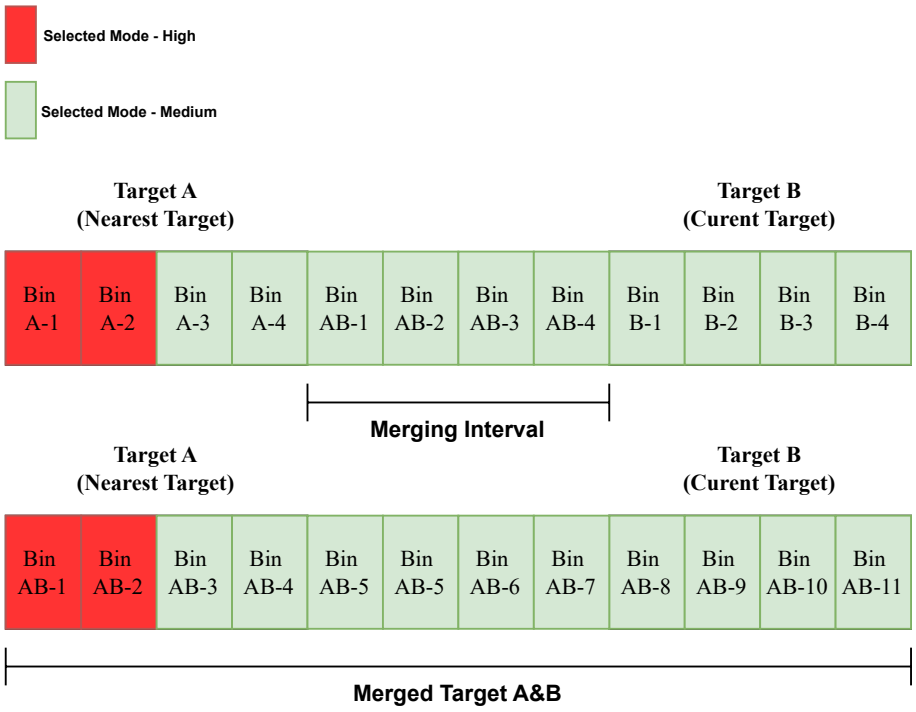


Figure 5.4: Merged Target A & B.

In addition, the merged target records information through a `merged_targets_list`, which stores the intermediate targets that become unintentionally acquired during continuous recording, along with the mode used for that interval. Finally, the merged target updates orbit-dependent temporal metadata, including `start_pso` and `pso_duration`, so that subsequent stages (e.g., further nearest-neighbor search and additional merging) can operate on a consistent target-level representation. Algorithm 11 summarizes this merge construction procedure.

Algorithm 11 Merge the Current Target with Its Nearest Same-Orbit Neighbor

procedure MERGETARGETS($T_{\text{cur}},\ T_{\text{near}},\ \mathcal{B}_{\text{int}}$)
  $(T_{\text{lead}}, T_{\text{trail}}) \leftarrow$ ORDERTARGETSBYINDEX($T_{\text{cur}}, T_{\text{near}}$)
  $\mathcal{B}_{\text{merge}} \leftarrow T_{\text{lead}}.\texttt{bin\_list} \parallel \mathcal{B}_{\text{int}} \parallel T_{\text{trail}}.\texttt{bin\_list}$
  for $i \leftarrow 0$ **to** $|\mathcal{B}_{\text{merge}}| - 1$ **do**
    $\mathcal{B}_{\text{merge}}[i].\texttt{index} \leftarrow i$
  end for
  $fore \leftarrow \mathcal{B}_{\text{merge}}[0].\texttt{fore\_endpoint}$
  $rear \leftarrow \mathcal{B}_{\text{merge}}[|\mathcal{B}_{\text{merge}}| - 1].\texttt{rear\_endpoint}$
  $p \leftarrow \max(T_{\text{cur}}.\texttt{priority},\ T_{\text{near}}.\texttt{priority})$
  $o \leftarrow T_{\text{near}}.\texttt{orbit\_number}$ ▷ same orbit by construction
  $(pso_{start}, pso_{dur}) \leftarrow$ UPDATEPSO($T_{\text{lead}}, T_{\text{trail}}$)
  $\mathcal{L} \leftarrow$ BUILDMERGEDTARGETSLIST($T_{\text{lead}}, T_{\text{trail}}, \mathcal{B}_{\text{int}}$)
  $is\_hybrid \leftarrow$ HASMULTIPLEMODES($\mathcal{B}_{\text{merge}}$)
  $name \leftarrow$ CONCATNAMES($T_{\text{lead}}.\texttt{target\_name},\ T_{\text{trail}}.\texttt{target\_name}$)
  $T_{\text{new}} \leftarrow$ CREATETARGET($name,\ \mathcal{B}_{\text{merge}},\ o,\ p,\ fore,\ rear,\ pso_{start},\ pso_{dur},\ is\_hybrid$)
  $T_{\text{new}}.\texttt{is\_merged} \leftarrow$ True
  $T_{\text{new}}.\texttt{merged\_targets\_list} \leftarrow \mathcal{L}$
  return T_{new}
end procedure

5.5 Command-Constrained Target Acquisition

This section presents the overall command-constrained target acquisition procedure, which integrates all previously introduced components into a single decision-making pipeline. Building upon the target initialization, nearest neighbor identification, merge feasibility assessment, and target merging operations, the algorithm constructs a final acquisition plan that strictly respects mission command constraints.

The core challenge addressed at this stage is that each uninterrupted recording interval consumes command resources. As a result, independently acquiring all remaining targets may exceed the allowable number of commands per orbit. To handle this issue, the algorithm processes targets in descending order of priority and, for each target, determines whether it should be acquired independently, implicitly accepted as a consequence of prior merging, merged with an acquired target, or simply skipped.

First of all, Algorithm 13 checks whether the current target has already been uninten-

tionally recorded during the merging of higher-priority targets as described in Algorithm 12. If such an intermediate acquisition exists and the associated recording mode is compatible with the target's allowed modes, the target is accepted without consuming additional commands. Otherwise, the target is marked as recorded but unusable and will be handled for further final evaluation.

If the target has not been implicitly recorded, the algorithm evaluates whether acquiring it independently would remain within the remaining command budget.

When this is not feasible, the algorithm attempts to merge the target with the nearest previously acquired target on the same orbit, as determined by Algorithm 9. The merge is only executed if it passes the feasibility checks defined in Algorithm 10, which jointly enforce boundary-mode compatibility and remaining orbital memory constraints. Successful merges replace the existing acquired target with a new composite target, constructed as described in the previous section.

Through this integrated decision process, the algorithm produces a prioritized and a command-feasible acquisition plan that accounts for both explicit recording decisions and implicit acquisitions arising from target merging. The resulting plan maximizes scientific value while strictly adhering to mission-level command and memory constraints.

Algorithm 12 Check for Intermediate Acquisition

```
1: procedure CheckIntermediateAcquisition(𝒜, T)
2:     for all T_acq ∈ 𝒜 do
3:         if T ∈ T_acq.merged_targets_list then
4:             return (True, GetMode(T_acq, T))
5:         end if
6:     end for
7:     return (False, None)
8: end procedure
```

Algorithm 13 Command-Constrained Target Acquisition

procedure AcquireTargets($\mathcal{B}_{\text{proc}}$)
  $\mathcal{A} \leftarrow [\,]$ ▷ acquired targets
  $\mathcal{R} \leftarrow [\,]$ ▷ intermediate recorded targets
  $\mathcal{B}_{\text{sel}} \leftarrow$ deep copy of $\mathcal{B}_{\text{proc}}$
  $\mathcal{T}_{\text{sel}} \leftarrow$ InitializeTargets($\mathcal{B}_{\text{sel}}$)
  Sort $\mathcal{T}_{\text{sel}}$ in descending order of priority
  for all $T_{\text{cur}} \in \mathcal{T}_{\text{sel}}$ **do**
    $(C_{\text{rem}}, M_{\text{rem}}) \leftarrow$ RemainingCommandsAndMemory($\mathcal{A}$)
    $(is\_intermediate,\ mode_{\text{inter}}) \leftarrow$ CheckIntermediateAcquisition($\mathcal{A},\ T_{\text{cur}}$)
    if $is\_intermediate$ = True **then**
      if $mode_{\text{inter}} \in$ GetTargetModes(T_{cur}) **then**
        SetAllBinsToMode($T_{\text{cur}},\ mode_{\text{inter}}$)
        Append T_{cur} to $\mathcal{A}$
      else
        Append T_{cur} to $\mathcal{R}$
      end if
      continue
    end if
    if $is\_intermediate$ = True **then**
      if $mode_{\text{inter}} \in$ GetTargetModes(T_{cur}) **then**
        SetAllBinsToMode($T_{\text{cur}}, mode_{\text{inter}}$)
        Append T_{cur} to $\mathcal{A}$
      else
        Append T_{cur} to $\mathcal{R}$
      end if
      continue
    end if
    if CheckCommandsWithCurrentTarget($\mathcal{A}, T_{\text{cur}}$) **then**
      Append T_{cur} to $\mathcal{A}$
      continue
    end if
    if not HasSameOrbitTarget($\mathcal{A}, T_{\text{cur}}$) **then**
      continue
    end if
    $T_{\text{near}} \leftarrow$ FindNearestTarget($\mathcal{A}, T_{\text{cur}}$)
    $(flag, \mathcal{B}_{\text{int}}) \leftarrow$ IsMergeable($T_{\text{cur}}, T_{\text{near}}, C_{\text{rem}}, M_{\text{rem}}$)
    if $flag$ = False **then**
      continue
    end if
    $T_{\text{new}} \leftarrow$ MergeTargets($T_{\text{cur}}, T_{\text{near}}, \mathcal{B}_{\text{int}}$)
    ReplaceInListByIdentity($\mathcal{A}, T_{\text{near}}, T_{\text{new}}$)
  end for
  return $\mathcal{A}$
end procedure

Part IV

Experiment Result and Conclusion

CHAPTER 6

EVALUATION METRICS

Since the considered mission planning problem is multi-objective and constrained, no single metric is sufficient to fully characterize solution quality. Therefore, a set of complementary metrics is adopted to evaluate the algorithm from operational, scientific, and computational perspectives.

6.1 Assignment Value

The primary performance metric is the *Assignment Value* (AV), which quantifies the overall quality of a generated acquisition plan by jointly considering target priorities and assigned acquisition modes. Unlike classical knapsack-based formulations that only count the number of scheduled tasks, this metric explicitly captures the trade-off between acquiring a target at its preferred resolution and recording it at a degraded mode due to resource constraints.

In practice, a target recorded at a degraded mode still retains partial scientific value, even though it does not fully satisfy the user's original request. Therefore, a binary success/failure metric would fail to capture this important trade-off between data quality and resource feasibility. To address this limitation, targets are divided into two categories: *acquired targets* and *recorded targets*. This distinction is central to the proposed evaluation framework, as it allows partial satisfaction of user requests to be explicitly quantified rather than ignored.

In this proposed framework, the smallest operational and decision-making unit is the *bin*, obtained by discretizing each target along the satellite trajectory. Consequently, evaluation metrics are defined at the bin level, while target-level performance can be recovered through aggregation.

Formally, the assignment value is defined as a normalized weighted sum of the contributions of acquired and recorded bins in targets:

$$\mathrm{AV} = \frac{\sum_{t \in AT} \sum_{b \in \mathcal{B}_t^A} w(m_b, p_t) \; + \; \epsilon \sum_{t \in RT} \sum_{b \in \mathcal{B}_t^R} w(m_b, p_t)}{\sum_{t \in T} \sum_{b \in \mathcal{B}_t} w(m_b^\star, p_t)}. \tag{6.1}$$

where:

- T denotes the set of all user-defined targets.
- $\mathcal{B}_t$ denotes the set of bins obtained by discretizing target $t \in T$ along the satellite trajectory.
- $\mathcal{B}_t^A \subseteq \mathcal{B}_t$ denotes the subset of bins of target t that are *acquired*.
- $\mathcal{B}_t^R \subseteq \mathcal{B}_t$ denotes the subset of bins of target t that are *recorded*.
- $AT := \{\, t \in T \mid \mathcal{B}_t^A \neq \emptyset \,\}$ denotes the set of targets that contain at least one acquired bin.
- $RT := \{\, t \in T \mid \mathcal{B}_t^R \neq \emptyset \,\}$ denotes the set of targets that contain at least one recorded bin.
- m_b denotes the recording mode assigned to bin b.
- $m_b^\star$ denotes the highest preferred recording mode associated with the parent target of bin b, used as the reference for normalization.
- p_t denotes the priority assigned to target t, which is inherited by all bins belonging to that target.
- $\epsilon \in [0, 1]$ is a customizable coefficient that controls the relative contribution of recorded bins with respect to acquired bins.
- $w(m_b, p_t)$ is a weight function that combines recording mode quality and target priority. In this work, it is defined as $w(m_b, p_t) = p_t \cdot q(m_b)$, where $q(m_b) \in [0, 1]$ is a mode-dependent quality factor. Note that this linear formulation is adopted for simplicity,

and the framework is not restricted to this choice; $w(\cdot)$ can be defined as any function that appropriately captures the trade-off between mode quality and target priority.

When in the case of $\epsilon = 0$, the assignment value becomes a classical Knapsack metric where we are just trying to maximize the number of requests with the highest mode:

$$\text{AV} = \frac{\sum_{t \in AT} \sum_{b \in \mathcal{B}_t^A} w(m_b, p_t)}{\sum_{t \in T} \sum_{b \in \mathcal{B}_t} w(m_b^\star, p_t)}. \tag{6.2}$$

The normalization by the maximum achievable weighted value ensures that the assignment value is bounded within $[0, 1]$, enabling consistent comparison across different scenarios, target sets, and mission configurations.

A value of AV $= 1$ corresponds to the "ideal" (infeasible, in general) case in which all targets are acquired at their highest preferred mode, while lower values indicate increasing levels of compromise due to operational constraints. Therefore, AV $= 1$ serves as a theoretical performance upper bound, since the denominator represents an idealized solution that cannot typically be achieved under real-world mission constraints. Nevertheless, our definition of AV is still valuable, since its primary purpose is not to provide an absolute measure of solution quality, but rather to serve as a normalized criterion for comparing the relative performance of different algorithms.

In the proposed algorithm, acquisition decisions are driven by target priority, which also demonstrates that onboard resources such as memory capacity and the number of executable commands is limited. By processing targets in descending order of priority, the algorithm systematically allocates high-quality acquisition opportunities to the most critical targets first, therefore, maximizing the overall scientific or operational value of the mission under resource constraints.

In the experimental evaluation presented in the next chapter, the assignment value is used as the primary metric to assess the trade-offs achieved by the proposed heuristic under varying memory limits, command budgets, and priority distributions.

CHAPTER 7

EXPERIMENT RESULT

In this thesis, the proposed constraint-aware heuristic, as described in Figure 7.1, is evaluated using both synthetic scenarios and realistic mission data derived from radar altimeter operations. The experimental assessment focuses primarily on computational efficiency and mission-critical constraints, including onboard memory capacity and the limited number of executable commands per orbit, as discussed in the previous chapters.

The set of candidate acquisition targets is derived from the Operational Land and Thematic Center OLTC portal, which provides representative hydrology-related targets used in current altimetry missions. These targets are evaluated against real altimetry mission conditions using publicly available Sentinel-6 data products obtained from the EUMETSAT data center [26].

To provide a meaningful performance assessment, the results obtained with the proposed method are compared against two reference approaches. The first baseline corresponds to our previously published work in the edited volume *Modeling and Optimization in Space Engineering: Challenges of the Near Future* [69]. While this earlier approach adopts a similar priority-driven scheduling philosophy, it operates at the target level and does not incorporate bin initialization, fine-grained memory estimation, or bin-level mode optimization. As a result, it provides a natural baseline for quantifying the benefits introduced by the bin-based formulation and enhanced constraint handling proposed in this thesis.

In addition to this target-level baseline, a greedy scheduling strategy is considered as a second point of comparison.

In this greedy approach, targets with uniform priorities are processed in random order and scheduled using the highest available acquisition mode whenever resource constraints allow, after exhausting the number of commands. No target merging, bin-level reorganization, or optimization of command usage is performed. This strategy reflects a simplified yet

realistic operational baseline, commonly adopted in mission planning workflows, and serves to highlight the limitations of purely local decision-making.

By comparing the proposed heuristic against both an established target-level baseline and a simplified greedy strategy, the experimental results aim to demonstrate not only absolute performance improvements, but also the specific contributions of bin-level discretization, mode adaptation, and command-aware optimization.

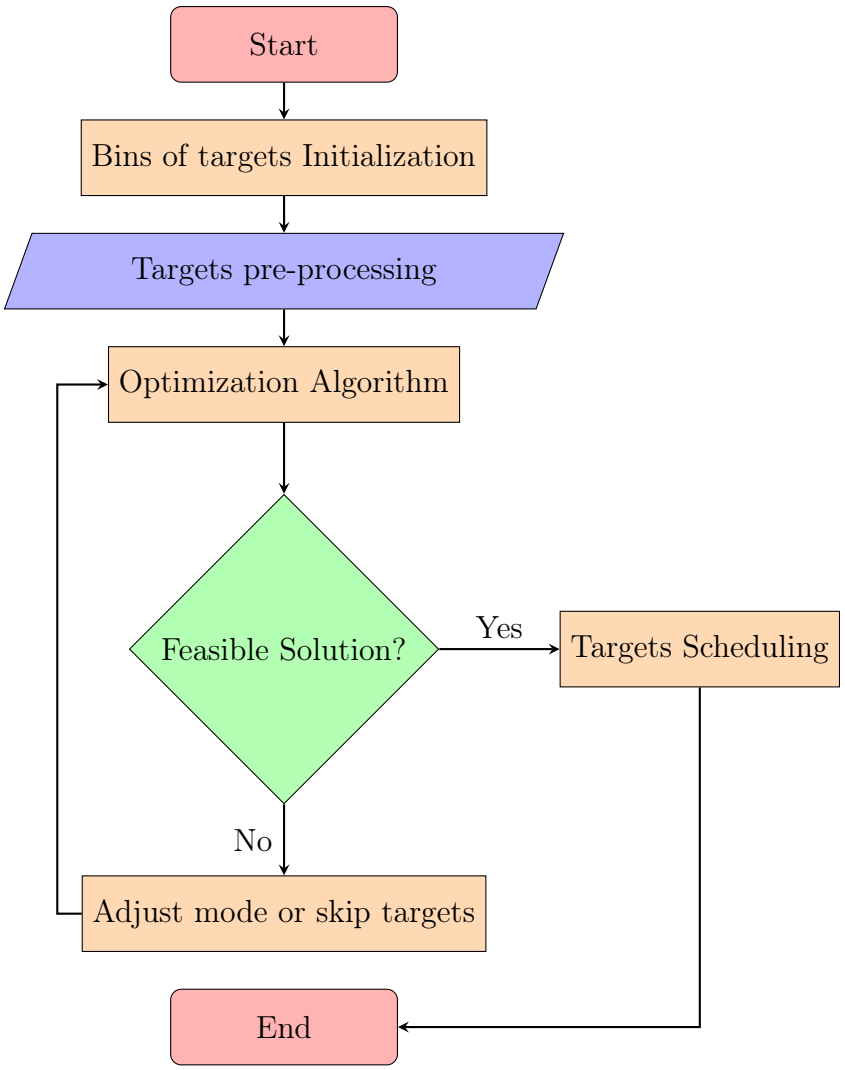


Figure 7.1: Flowchart of the Proposed Optimization Framework

Table 7.1 from our previously published work reports the performance of the greedy algorithm when applied to a representative set of altimetry acquisition targets across multiple

orbits [69]. For each orbit, the table summarizes the percentage of onboard memory allocated to high-resolution (LX) acquisitions, the number of targets successfully acquired, and the total number of candidate targets.

Orbit	Memory Usage [%]		Acquired Targets		Total Targets	AV
	LX	LRMC	LX	LRMC		
1	21.4	76.8	6	60	83	0.024
2	24.6	72.1	9	78	109	0.119
3	19.8	78.5	8	72	100	0.100
4	27.1	70.2	7	53	75	0.627
5	29.4	68.6	8	54	78	0.333
6	31.0	65.1	6	35	51	0.608
7	22.8	73.6	5	31	45	0.422
8	18.9	77.4	4	28	40	0.250
9	26.7	69.5	7	55	78	0.397
10	28.2	70.6	9	65	93	0.505
11	23.5	72.8	6	41	59	0.169
12	30.6	66.2	8	48	70	0.729
13	27.4	68.9	9	64	91	0.407
14	20.7	75.0	6	46	65	0.246
15	19.6	76.3	6	48	67	0.149

Table 7.1: Greedy baseline results with mode-level breakdown of memory usage and acquired targets.

Before introducing the proposed bin-level heuristic, we first evaluate a target-level baseline in which targets are treated as indivisible units. In this setting, mission planners explicitly specify, for each target, a priority level and a set of permitted acquisition modes. This configuration reflects a realistic operational scenario in which scientific importance and acceptable data quality are determined by the mission control center.

Table 7.2 illustrates an example of such a user-defined configurations for Orbit 1. Each target is associated with a priority value and a mode preference set, which may include the high-resolution mode (`LX`) only, or both `LX` and the lower-resolution fallback mode (`LRMC`). Targets allowing both modes can be flexibly scheduled depending on the available onboard

resources, whereas targets restricted to LX can only be acquired at the highest quality.

Table 7.2: Orbit 1 Status

Target	Mode	Priority
1	[LX]	10
2	[LX, LRMC]	2
3	[LX, LRMC]	6
4	[LX, LRMC]	5
5	[LX, LRMC]	1
6	[LX, LRMC]	1
7	[LX, LRMC]	1
8	[LX, LRMC]	1
9	[LX, LRMC]	8
10	[LX, LRMC]	6
	⋮	
74	[LX]	10
75	[LX, LRMC]	1
76	[LX, LRMC]	1
77	[LX]	10
78	[LX, LRMC]	1
79	[LX, LRMC]	1
80	[LX, LRMC]	1
81	[LX, LRMC]	1
82	[LX, LRMC]	9

Based on these user-defined inputs, the target-level baseline attempts to maximize mission value by prioritizing high-priority targets while adapting the acquisition mode whenever possible. Unlike the greedy baseline, this approach allows a target to be recorded in LRMC when LX acquisition is not feasible, thereby avoiding unnecessary target rejection under memory constraints. However, since targets are still treated as atomic entities, no bin-level optimization or partial acquisition is performed.

The resulting performance of the target-level baseline across all orbits is summarized in Table 7.3. For each orbit, the table reports the memory usage attributed to LX and LRMC, the number of acquired targets per mode, the number of recorded targets, and the corresponding assignment value (AV) computed using $\epsilon = 0.3$.

Orbit	Memory Usage [%]		Acquired Targets		Recorded Targets	Total Targets	AV ($\epsilon = 0.3$)
	LX	LRMC	LX	LRMC			
1	24.5	59.0	12	51	18	83	0.524
2	21.0	63.5	6	22	30	109	0.339
3	27.5	52.0	5	18	25	100	0.305
4	31.0	50.5	14	41	12	75	0.781
5	34.0	47.5	8	30	18	78	0.556
6	29.5	55.0	9	28	8	51	0.773
7	22.0	60.5	6	18	15	45	0.633
8	26.0	54.0	4	12	14	40	0.505
9	33.0	49.0	10	31	20	78	0.603
10	35.5	48.0	12	46	18	93	0.682
11	23.0	58.5	5	16	16	59	0.437
12	37.0	51.0	14	44	7	70	0.859
13	32.0	53.5	12	38	18	91	0.609
14	25.5	57.0	6	20	14	65	0.465
15	21.5	61.0	5	16	15	67	0.381

Table 7.3: Target-level baseline: mode-level breakdown of memory usage and acquired targets.

While the target-level baseline already improves over the greedy strategy by introducing mode flexibility, it still treats each target as an indivisible entity. As a consequence, once a target is selected, all its internal segments are acquired using the same recording mode, even though local variations in geometry or resource availability may allow a more efficient allocation.

To address this issue, we extend the target-level formulation by operating at the bin level. By discretizing each target into multiple bins, the proposed approach enables fine-grained mode adaptation within individual targets, allowing high-priority segments to be acquired in `LX` while assigning `LRMC` to less critical segments when necessary.

This additional degree of freedom does not increase the total number of targets that can be scheduled, but it improves how available resources are utilized within each target. The resulting performance of the bin-level method is presented in Table 7.4, where a consistent

yet moderate improvement over the target-level baseline can be observed across all orbits.

Orbit	Memory Usage [%]		Acquired Targets		Recorded Targets	Total Targets	AV ($\epsilon = 0.3$)
	LX	LRMC	LX	LRMC			
1	27.0	56.0	16	48	14	83	0.563
2	24.5	60.0	9	20	26	109	0.382
3	30.0	49.0	8	16	22	100	0.352
4	34.0	47.0	18	38	10	75	0.821
5	37.0	45.0	12	28	16	78	0.601
6	32.5	51.0	12	26	7	51	0.812
7	25.5	57.0	9	16	12	45	0.681
8	29.0	51.0	6	11	12	40	0.552
9	35.0	47.0	13	29	17	78	0.651
10	38.0	45.0	16	43	14	93	0.731
11	26.5	55.0	8	15	13	59	0.489
12	39.0	48.0	18	40	6	70	0.902
13	34.5	50.0	16	36	16	91	0.658
14	28.0	55.0	9	18	13	65	0.509
15	24.5	58.0	8	15	13	67	0.428

Table 7.4: Proposed bin-level method: mode-level breakdown of memory usage and acquired targets.

Figure 7.2 provides a consolidated comparison of the three acquisition strategies in terms of assignment value across all orbits. The results clearly illustrate a consistent performance pattern, with the greedy baseline yielding the lowest AV, followed by the target-level baseline, and finally the proposed bin-level method achieving the highest AV.

Introducing mode flexibility at the target level leads to a substantial increase in AV across all orbits. By allowing targets to be recorded in `LRMC` when `LX` acquisition is not feasible, the target-level baseline significantly reduces target rejection and improves overall coverage. However, since targets are still treated as indivisible units, opportunities for further optimization within individual targets remain unexploited.

The proposed bin-level method consistently outperforms the target-level baseline, although the improvement remains moderate. This behavior is expected, as the bin-level for-

mulation does not increase the number of schedulable targets but instead enhances resource utilization through fine-grained mode adaptation within targets. By selectively assigning `LX` to high-value bins to less critical bins, the algorithm achieves a higher AV without introducing excessive complexity or unrealistic performance gains.

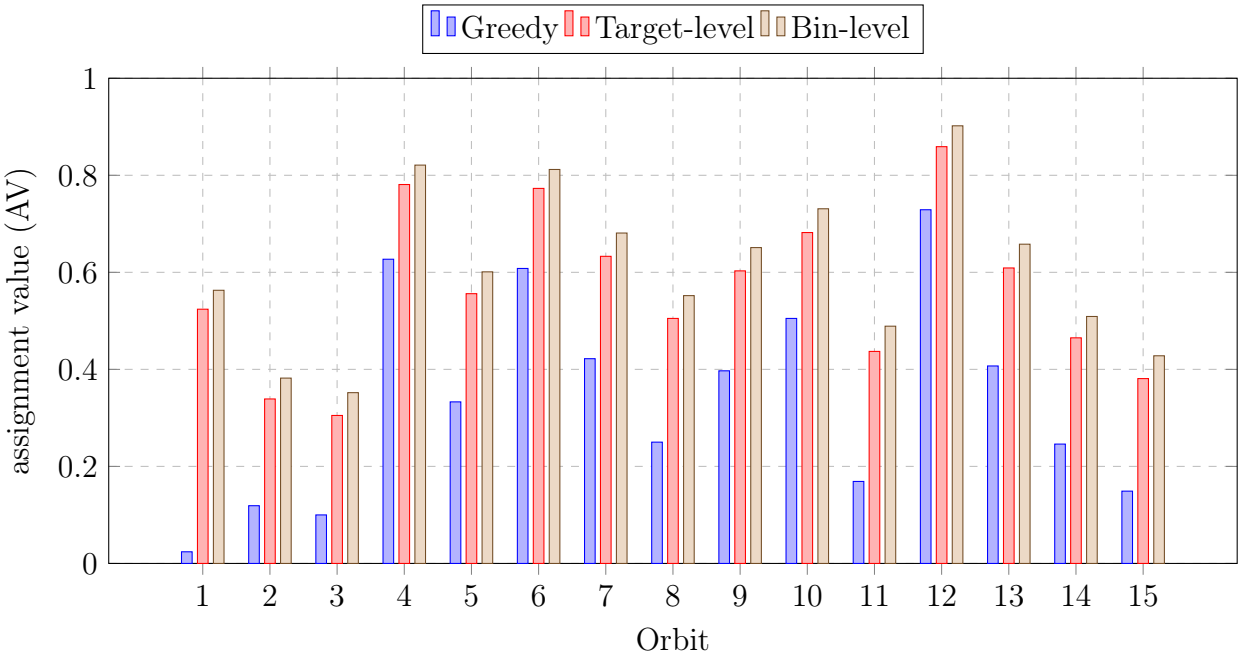


Figure 7.2: Comparison of assignment value (AV) across different acquisition strategies for each orbit.

Figure 7.3 reports the computational behavior of the proposed bin-level method under two complementary experimental settings.

The left subfigure evaluates the scalability of the algorithm with respect to the total number of targets. In this experiment, the number of targets is gradually increased while keeping other parameters unchanged. As expected, the computation time grows approximately linearly, confirming that the additional bin-level processing increases the constant factor of the runtime but does not change its overall complexity order.

The right subfigure investigates a different aspect of the algorithm by fixing the total number of targets to 100 and progressively increasing the number of high-priority targets. This experiment is designed to assess the sensitivity of the bin-level method to priority

distribution rather than problem size.

When the number of high-priority targets is zero, all targets are assigned the minimum priority level. In this configuration, the algorithm treats all targets uniformly and evaluates them independently at the bin level. Since no early decisions or preferential merges are triggered, the algorithm explores a larger set of feasible combinations, which results in a relatively higher and stable execution time. As the number of high-priority targets increases from zero, additional structure is introduced into the decision process. In particular, high-priority targets are processed earlier and create more opportunities for bin-level merging with subsequent same-priority targets. These early merges effectively reduce the number of independent bin evaluations required, leading to a temporary decrease in computation time in the intermediate regime.

When the proportion of high-priority targets becomes large, the algorithm performs more frequent priority-based checks, mode evaluations, and conflict resolution steps. These operations introduce additional overhead, causing the computation time to increase again. Nevertheless, the observed growth remains controlled and does not exhibit exponential behavior.

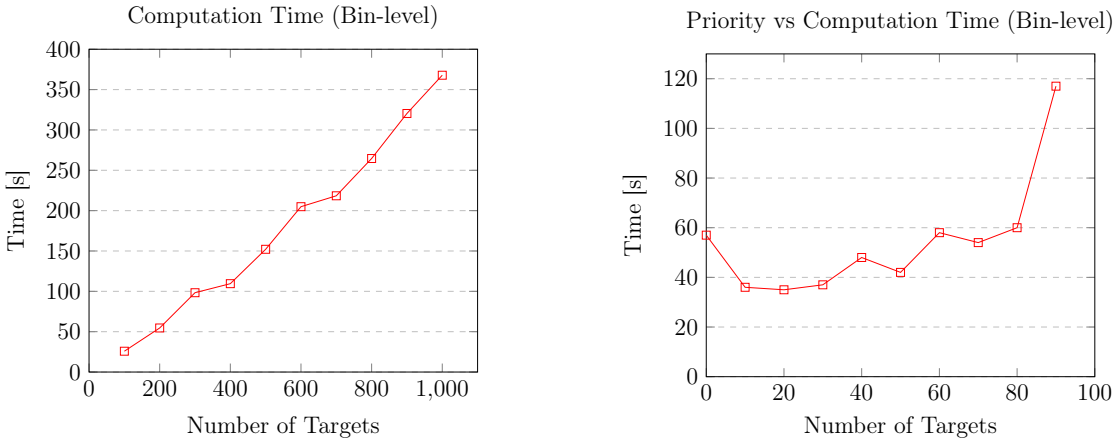


Figure 7.3: Computation time of the bin-level method as a function of the number of targets and target priority.

CHAPTER 8

DISCUSSIONS AND CONCLUSIONS

8.1 Thesis Summary

This thesis investigated the problem of altimetry data-acquisition planning under strict onboard constraints, with a particular focus on memory capacity and the limited number of recording commands. Motivated by the operational complexity of modern Earth-observation missions, a constraint-aware heuristic framework was proposed to improve acquisition efficiency while preserving computational tractability.

The core contribution of this work lies in the introduction of a bin-level optimization strategy built upon the heuristic framework proposed in our previously published work [69]. In this extended formulation, each target is discretized into smaller acquisition units and evaluated individually. Compared to traditional target-level planning, this finer-grained representation enables more accurate estimation of onboard memory usage, greater flexibility in recording mode selection, and more effective exploitation of merging opportunities across adjacent targets. By explicitly incorporating command constraints together with mode-dependent memory costs, the proposed approach more faithfully reflects real mission operations.

Overall, this thesis demonstrates that bin-level optimization represents a practical and effective extension of existing heuristic planning approaches for altimetry missions. By bridging the gap between high-level target selection and low-level operational constraints, the proposed framework provides a flexible tool for mission planners to tailor acquisition strategies according to evolving mission requirements.

8.2 Limitation and Insight

While the proposed bin-level optimization framework demonstrates clear advantages for the considered altimetry mission scenario, it is important to acknowledge its limitations. The design of the heuristic, including the target discretization strategy, memory modeling, and command constraints, is closely tailored to the operational characteristics of the considered satellite mission. As a result, direct application of the proposed method to other missions may require additional adaptations.

However, this mission-specific focus also provides an important insight. By embedding operational constraints directly into the optimization process, the proposed framework demonstrates how domain knowledge can be leveraged to improve planning quality without resorting to computationally expensive global optimization techniques. The bin-level abstraction, in particular, offers a flexible intermediate representation that can be redefined to accommodate different mission characteristics.

8.3 Future Work

Future work may explore adaptive bin discretization strategies, integration with dynamic downlink scheduling, and the extension of the proposed framework to multi-instrument or multi-satellite missions.

In this work, we did not exhaustively investigate how varying the hyper-parameters of our algorithm affects performance. For example, one interesting future direction is fine-tuning the bin discretization step size, which acts as a key meta-parameter of the proposed framework. A smaller step size leads to a finer bin representation, improving acquisition accuracy and potentially increasing the assignment value (AV), but at the cost of higher computational complexity. Conversely, a larger step size reduces computation time by coarsening the bin structure, but may degrade solution quality and reduce the resulting AV.

This trade-off suggests the formulation of a joint objective function that explicitly balances solution quality and computational cost. By defining an objective that jointly considers the achieved assignment value and the required computational time, future work could investigate optimization strategies that automatically select or adapt the step size to maximize overall mission utility under given operational constraints.

BIBLIOGRAPHY

[1] C. Ahara and J. Rossbach. The scheduling problem in satellite communications systems. *IEEE Transactions on Communication Technology*, 15(3):364–371, 1967.

[2] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256, 2002. doi:10.1023/A:1013689704352.

[3] L. Barbulescu, J. Watson, L. Whitley, and A. Howe. Scheduling space–ground communications for the air force satellite control network. *Journal of Scheduling*, 7:7–34, 2004.

[4] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[5] T. Benoist and B. Rottembourg. Upper bounds for revenue maximization in a satellite scheduling problem. *4OR*, 2(3):235–249, 2004.

[6] Éric Bensana, Michel Lemaître, and Gérard Verfaillie. Earth observation satellite management. *Constraints*, 4(3):293–299, 1999.

[7] Dimitri P. Bertsekas. Dynamic programming and optimal control. *Athena Scientific*, 2012.

[8] N. Bianchessi and G. Righini. Planning and scheduling algorithms for the cosmoskymed constellation. *Aerospace Science and Technology*, 12(7):535–544, 2008.

[9] Nicola Bianchessi, Jean-François Cordeau, Jacques Desrosiers, Gilbert Laporte, and Véronique Raymond. A heuristic for the multi-satellite, multi-orbit and multi-user scheduling problem. *European Journal of Operational Research*, 177(2):750–762, 2007.

[10] J. Bonnet, M. Gleizes, E. Kaddoum, S. Rainjonneau, and G. Flandin. Multi-satellite mission planning using a self-adaptive multi-agent system. In *2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems*, pages 11–20. IEEE, 2015.

[11] N. Brown, B. Arguello, L. Nozick, and N. Xu. A heuristic approach to satellite range scheduling with bounds using lagrangian relaxation. *IEEE Systems Journal*, 12(4): 3828–3836, 2018.

[12] Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012. doi:10.1561/2200000024.

[13] Edmund K. Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Ozcan, and John Woodward. Hyper-heuristics: A survey of the state of the art. *European Journal of Operational Research*, 207(1):1–14, 2010. doi:10.1016/j.ejor.2010.02.032.

[14] Raul Castano and Andrew Lenzen. Onboard autonomy for space missions. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, pages 1–8. ESA, 2001.

[15] Amedeo Cesta, Simone Fratini, and Andrea Orlandini. Automated planning and scheduling for space mission control. *Annual Reviews in Control*, 38(1):84–94, 2014. doi:10.1016/j.arcontrol.2014.03.004.

[16] Z. Chang, A. Punnen, Z. Zhou, and S. Cheng. Solving dynamic satellite image data downlink scheduling problem via an adaptive bi-objective optimization algorithm. *Computers & Operations Research*, 160:106388, 2023.

[17] X. Chen, G. Reinelt, G. Dai, and M. Wang. Priority-based and conflict-avoidance heuristics for multi-satellite scheduling. *Applied Soft Computing*, 69:177–191, 2018.

[18] X. Chen, G. Reinelt, G. Dai, and A. Spitz. A mixed integer linear programming model for multi-satellite scheduling. *European Journal of Operational Research*, 275 (2):694–707, 2019.

[19] Steve Chien, Gregg Rabideau, Russell Knight, and Rob Sherwood. Automated planning and scheduling for space mission operations. *Proceedings of the IEEE Aerospace Conference*, 7:375–390, 2000. doi:10.1109/AERO.2000.878426.

[20] Steve Chien, Rob Sherwood, Daniel Tran, Benjamin Cichy, and Raul Castano. Autonomous science on the eo-1 mission. *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, pages 1–8, 2005.

[21] Steve A. Chien, Gregg Rabideau, Russell L. Knight, R. Sherwood, Belinda Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, and D. Tran. Aspen – automated planning and scheduling for space mission operations. In *Proceedings of SpaceOps 2000*, Toulouse, France, 2000. SpaceOps. Describes the ASPEN system for automation of planning and scheduling in spacecraft mission operations.

[22] 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. doi:10.2514/1.I010620.

[23] J. Chun et al. Deep reinforcement learning for the agile earth observation satellite scheduling problem. *Mathematics*, 11(19):4059, 2023. doi:10.3390/math11194059.

[24] J. Cordeau and G. Laporte. Maximizing the value of an earth observation satellite orbit. *Journal of the Operational Research Society*, 56(8):962–968, 2004.

[25] Amin Dastanpour, Suhaimi Ibrahim, and Reza Mashinchi. Using genetic algorithm to supporting artificial neural network for intrusion detection system. In *Proceedings of the International Conference on Computer Security and Digital Investigation (ComSec)*, pages 1–13. The Society of Digital Information and Wireless Communication, 2014. Available via SDIWC / academic repositories.

[26] Salvatore Dinardo, Claire Maraldi, Emeline Cadier, Pierre Rieu, Jeremie Aublanc, Adrien Guerou, Francois Boy, Thomas Moreau, Nicolas Picot, and Remko Scharroo. Sentinel-6 mf poseidon-4 radar altimeter: Main scientific results from s6pp lrm and ufsar chains in the first year of the mission. *Advances in Space Research*, 73(1):337–375, 2024.

[27] B. Du and S. Li. A new multi-satellite autonomous mission allocation and planning method. *Acta Astronautica*, 163:287–298, 2019.

[28] Y. Du, L. Xing, J. Zhang, Y. Chen, and Y. He. Moea based memetic algorithms for multi-objective satellite range scheduling problem. *Swarm and Evolutionary Computation*, 50:100576, 2019.

[29] Y. H. Du, L. N. Xing, and Z. Q. Cai. Survey on intelligent scheduling technologies for unmanned flying craft clusters. *Acta Automatica Sinica*, 46(2):222–241, 2020.

[30] D. Eddy and M. Kochenderfer. A maximum independent set method for scheduling earth-observing satellite constellations. *Journal of Spacecraft and Rockets*, 58(5):1416–1429, 2021.

[31] Bruno Ferrari, Matteo Rossi, and Luca Galli. Satellite scheduling problems: A survey of applications in earth and outer space observation. *Computers & Operations Research*, 164:106875, 2025. doi:10.1016/j.cor.2024.106875.

[32] Y. Gao, J. Sun, and H. Zhang. Deep reinforcement learning for satellite downlink scheduling. *Aerospace Science and Technology*, 112:106651, 2021.

[33] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Francisco, 1979. ISBN 0716710447.

[34] Fred Glover and Manuel Laguna. *Tabu Search*. Springer, Boston, MA, 1997. ISBN 978-1-4613-4941-0. doi:10.1007/978-1-4615-6089-0.

[35] Tuomas Haarnoja et al. Soft actor–critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International Conference on Machine Learning*, pages 1861–1870, 2018.

[36] Soon-mi Han, Seung-woo Beak, Kyuem-rae Cho, Dae-woo Lee, and Hae-dong Kim. Satellite mission scheduling using genetic algorithm. In *Proceedings of the SICE-ICASE International Joint Conference*, pages 5861–5866, Busan, Korea, 2006. IEEE. doi:10.1109/SICE.2006.315619.

[37] L. He, X. Liu, G. Laporte, Y. Chen, and Y. Chen. An improved adaptive large neighborhood search algorithm for multiple agile satellites scheduling. *Computers & Operations Research*, 100:12–25, 2018.

[38] Y. He, L. Xing, Y. Chen, W. Pedrycz, L. Wang, and G. Wu. A generic markov decision process model and reinforcement learning method for scheduling agile earth observation satellites. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(3):1463–1474, 2020.

[39] Adam Herrmann, Mark Stephenson, and Hanspeter Schaub. Reinforcement learning for multi-satellite agile earth observing scheduling under various communication assumptions. In *AAS Space Flight Mechanics Meeting*, 2023. AAS 23-146.

[40] Adam Herrmann, Mark Stephenson, and Hanspeter Schaub. A comparative analysis of reinforcement learning methods for spacecraft scheduling. *Frontiers in Space Technologies*, 4:1263489, 2023. doi:10.3389/frspt.2023.1263489.

[41] Y. Huang, Z. Mu, S. Wu, B. Cui, and Y. Duan. Revising the observation satellite scheduling problem based on deep reinforcement learning. *Remote Sensing*, 13(12): 2377, 2021. doi:10.3390/rs13122377.

[42] D. Joslin and D. Clements. Squeaky wheel optimization. *Journal of Artificial Intelligence Research*, 10:353–373, 1999.

[43] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[44] D. Karapetyan, S. Minic, K. Malladi, and A. Punnen. Satellite downlink scheduling problem: A case study. *Omega*, 53:115–123, 2015.

[45] Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. Optimization by simulated annealing. In *Science*, volume 220, pages 671–680, 1983. doi:10.1126/science.220.4598.671.

[46] Ailsa H. Land and Alison G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[47] Wiley J. Larson and James R. Wertz. *Applied Space Systems Engineering*. Microcosm Press, 2014.

[48] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020. ISBN 9781108486828.

[49] Michel Lemaître, Gérard Verfaillie, Frank Jouhaud, Jean-Michel Lachiver, and Nicolas Bataille. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology*, 6:367–381, 2002.

[50] S. Li, Q. Yu, and H. Ding. Reviews and prospects in satellite range scheduling problem. *Autonomous Intelligent Systems*, 3(1):9, 2023.

[51] X. Li, Z. Li, and X. Wang. Automatic scheduling for earth observation satellite with temporal specifications. *IEEE Transactions on Aerospace and Electronic Systems*, 56 (4):3350–3365, 2020.

[52] Xiaoming Li, Cheng Zhao, and Jianjun Xu. A review of frameworks, models, and algorithms for large-scale imaging satellite mission planning. *Expert Systems with Applications*, 245:125123, 2025. doi:10.1016/j.eswa.2025.125123.

[53] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[54] Timothy Lillicrap et al. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

[55] Q. Liu, Y. Chen, and H. Wang. Learning-based priority estimation for agile earth observation satellite scheduling. *Expert Systems with Applications*, 183:115401, 2021.

[56] X. Liu, G. Laporte, Y. Chen, and R. He. An adaptive large neighborhood search metaheuristic for agile satellite scheduling with time-dependent transition time. *Computers & Operations Research*, 86:41–53, 2017.

[57] Z. Liu, L. Gao, and Y. Chai. Research on task scheduling mechanism of distributed satellite system. *Journal of Computational Information Systems*, 10(24):10703–10713, 2014.

[58] A. Maillard, G. Verfaillie, C. Pralet, et al. Adaptable data download schedules for agile earth-observing satellites. *Journal of Aerospace Information Systems*, 13(3):280–300, 2016.

[59] Volodymyr Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015. doi:10.1038/nature14236.

[60] Volodymyr Mnih et al. Asynchronous methods for deep reinforcement learning. *International Conference on Machine Learning*, pages 1928–1937, 2016.

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

[62] David R. Morrison, Sheldon H. Jacobson, Joshua J. Sauppe, and Edward C. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2016. doi:10.1016/j.disopt.2016.01.005.

[63] Francesco Orabona. *A Modern Introduction to Online Learning*. arXiv:1912.13213, 2019. URL `https://arxiv.org/abs/1912.13213`.

[64] J. Ou et al. Deep reinforcement learning method for satellite range scheduling problem. *Engineering Applications of Artificial Intelligence*, 2023. doi:10.1016/j.engappai.2023.105445.

[65] Bruno Machado Pacheco, Laio Oriel Seman, Cezar Antonio Rigo, Eduardo Camponogara, Eduardo Augusto Bezerra, and Leandro dos Santos Coelho. Graph neural networks for the offline nanosatellite task scheduling problem. *Applied Soft Computing*, 186:114139, 2025. doi:10.1016/j.asoc.2025.114139. URL `https://www.sciencedirect.com/science/article/pii/S1568494625014528`.

[66] V. Pallagani, K. Roy, B. Muppasani, F. Fabiano, A. Loreggia, K. Murugesan, B. Srivastava, F. Rossi, L. Horesh, and A. Sheth. On the prospects of incorporating large language models (llms) in automated planning and scheduling (aps). *arXiv preprint arXiv:2401.02500*, 2024.

[67] Donald A. Parish. A genetic algorithm approach to automating satellite range scheduling. Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, USA, 1994. URL `https://scholar.afit.edu/etd/6770`. AFIT-GOR-ENS-94M-10.

[68] Guansheng Peng, Guopeng Song, Lining Xing, Aldy Gunawan, and Pieter Vansteenwegen. An exact algorithm for agile earth observation satellite scheduling with time-dependent profits. *Computers & Operations Research*, 120:104946, 2020. doi:10.1016/j.cor.2020.104946.

[69] Jonathan Pergoli, Tommaso Cesari, Zhihao Jin, Tancredi Busatta, Michele Maestrini, and Pierluigi Di Lizia. A constraint-aware heuristic for altimetry data-acquisition optimization. In Giorgio Fasano and János D. Pintér, editors, *Modeling and Optimization in Space Engineering: Challenges of the Near Future*, Lecture Notes in Computer Science. Springer, Cham, 2025.

[70] David Pisinger and Stefan Ropke. Large neighborhood search. *Handbook of Metaheuristics*, pages 399–419, 2010.

[71] Warren B. Powell. Approximate dynamic programming: Solving the curses of dimensionality. *Wiley*, 2011.

[72] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994. ISBN 9780471619772.

[73] D. S. Qiu, C. He, J. Liu, et al. A dynamic scheduling method of earth-observing satellites by employing rolling horizon strategy. *The Scientific World Journal*, page 304047, 2013.

[74] R. V. Rao. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *International Journal of Industrial Engineering Computations*, 7(1):19–34, 2016.

[75] Giovanni Righini and Matteo Salani. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers & Operations Research*, 36(4):1191–1203, 2009.

[76] A. A. Salman, I. Ahmad, and M. G. Omran. A metaheuristic algorithm to solve satellite broadcast scheduling problem. *Information Sciences*, 322:72–91, 2015. doi:10.1016/j.ins.2015.06.015.

[77] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.

[78] John Schulman et al. Proximal policy optimization algorithms. In *arXiv preprint arXiv:1707.06347*, 2017.

[79] Shai Shalev-Shwartz. *Online Learning and Online Convex Optimization*, volume 4. Now Publishers, 2012. doi:10.1561/2200000018.

[80] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. doi:10.1002/j.1538-7305.1948.tb01338.x.

[81] Y. C. She, S. Li, and Y. B. Zhao. Onboard mission planning for agile satellite using modified mixed-integer linear programming. *Aerospace Science and Technology*, 72: 204–216, 2018.

[82] David Silver et al. Mastering the game of go without human knowledge. *Nature*, 550: 354–359, 2017.

[83] Lorenzo Bruzzone Stefano Paterna, Massimo Santoni. An approach based on multiobjective genetic algorithms to schedule observations in planetary remote sensing missions. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, PP(99):1–1, 2020. doi:10.1109/JSTARS.2020.3015284. License: CC BY 4.0.

[84] Kai Sun, Li-ning Xing, and Ying-wu Chen. Agile earth observing satellites mission scheduling based on decomposition optimization algorithm. *Computer Integrated Manufacturing Systems*, 19(1):127–136, 2013. In Chinese.

[85] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.

[86] El-Ghazali Talbi. Metaheuristics: From design to implementation. *Wiley*, 2009.

[87] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *AAAI Conference on Artificial Intelligence*, 2016.

[88] Michel Vasquez and Jin-Kao Hao. A "logic-constrained" knapsack formulation and a tabu algorithm for the daily photograph scheduling of an earth observation satellite. *Computational Optimization and Applications*, 20:137–157, 2001. doi:10.1023/A:1011203002719.

[89] Haibo Wang, Minqiang Xu, Rixin Wang, and Yuqing Li. Scheduling earth observing satellites with hybrid ant colony optimization algorithm. In *Proceedings of the 2009 International Conference on Artificial Intelligence and Computational Intelligence*, pages 1–5, Shanghai, China, November 2009. IEEE. doi:10.1109/AICI.2009.87.

[90] X. Wang, G. Wu, L. Xing, and W. Pedrycz. Agile earth observation satellite scheduling over 20 years: Formulations, methods, and future directions. *IEEE Systems Journal*, 15(3):3881–3892, 2020.

[91] Z. Wang, X. Hu, H. Ma, and W. Xia. Learning multi-satellite scheduling policy with heterogeneous graph neural network. *Advances in Space Research*, 73(6):2921–2938, 2024.

[92] Ziyu Wang, Tom Schaul, Matteo Hessel, et al. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1995–2003, 2016.

[93] Dennis Wei and Alan V. Oppenheim. A branch-and-bound algorithm for quadratically-constrained sparse filter design. *IEEE Transactions on Signal Processing*, 61(4):1006–1018, 2013. doi:10.1109/TSP.2012.2226450.

[94] L. Wei, Y. Chen, M. Chen, and Y. Chen. Deep reinforcement learning and parameter transfer based approach for the multi-objective agile earth observation satellite scheduling problem. *Applied Soft Computing*, 110:107607, 2021.

[95] W.J. Wolfe and S.E. Sorensen. Three scheduling algorithms applied to the earth observing systems domain. *Management Science*, 46(1):148–168, 2000.

[96] G. H. Wu, M. H. Ma, J. H. Zhu, et al. Multi-satellite observation integrated scheduling method oriented to emergency tasks and common tasks. *Journal of Systems Engineering and Electronics*, 23(5):723–733, 2012.

[97] J. Wu, Y. Chen, Y. He, L. Xing, and Y. Hu. Survey on autonomous task scheduling technology for earth observation satellites. *Journal of Systems Engineering and Electronics*, 33(6):1176–1189, 2022.

[98] J. Wu, B. Y. Song, G. T. Zhang, et al. A data-driven improved genetic algorithm for agile earth observation satellite scheduling with time-dependent transition time. *Computers & Industrial Engineering*, 174:108823, 2022.

[99] Jian Wu, Jiawei Zhang, Jinghui Yang, and Lining Xing. Research on task priority model and algorithm for satellite scheduling problem. *IEEE Access*, 7:103031–103046, 2019. doi:10.1109/ACCESS.2019.2928992.

[100] F. Xhafa and A. Ip. Optimisation problems and resolution methods in satellite scheduling and spacecraft operation: a survey. *Enterprise Information Systems*, 15(8):1022–1045, 2019.

[101] J. Zhang et al. A multiobjective satellite data transmission scheduling method based NSGA-II. *2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*, 50:100560, 2019.

[102] Z. X. Zheng, J. Guo, and E. Gill. Onboard autonomous mission re-planning for multi-satellite system. *Acta Astronautica*, 145:28–43, 2018.