Quantum algorithms applied to satellite mission planning for Earth observation

Optimization

By Terra Quantum AG
terraquantum.swiss

2023

Quantum Algorithms applied to Satellite Mission Planning for Earth Observation

Serge Rainjonneau[†], Igor Tokarev[*], Sergei Iudin[*], Saaketh Rayaprolu[*], Karan Pinto[*], Daria Lemtiuzhnikova[*], Miras Koblan[*], Egor Barashov[*], Mohammad Kordzanganeh[*], Markus Pflitsch[*], and Alexey Melnikov[*§]
[†] Thales Alenia Space, 31100 Toulouse, France
[*] Terra Quantum AG, 9000 St. Gallen, Switzerland

***Abstract*—Earth imaging satellites are a crucial part of our everyday lives that enable global tracking of industrial activities. Use cases span many applications, from weather forecasting to digital maps, carbon footprint tracking, and vegetation monitoring. However, there are also limitations; satellites are difficult to manufacture, expensive to maintain, and tricky to launch into orbit. Therefore, it is critical that satellites are employed efficiently. This poses a challenge known as the satellite mission planning problem, which could be computationally prohibitive to solve on large scales. However, close-to-optimal algorithms can often provide satisfactory resolutions, such as greedy reinforcement learning, and optimization algorithms. This paper introduces a set of quantum algorithms to solve the mission planning problem and demonstrate an advantage over the classical algorithms implemented thus far. The problem is formulated as maximizing the number of high-priority tasks completed on real datasets containing thousands of tasks and multiple satellites. This work demonstrates that through solution-chaining and clustering, optimization and machine learning algorithms offer the greatest potential for optimal solutions. Most notably, this paper illustrates that a hybridized quantum-enhanced reinforcement learning agent can achieve a completion percentage of 98.5% over high-priority tasks, which is a significant improvement over the baseline greedy methods with a completion rate of 63.6%. The results presented in this work pave the way to quantum-enabled solutions in the space industry and, more generally, future mission planning problems across industries.**

***Index Terms*—Quantum algorithms, Earth observation, quantum reinforcement learning, satellite mission planning, quantum optimization**

I. Introduction

The reliable functioning of Earth-orbiting satellites crucially affects our everyday services such as connectivity [1], navigation [2], and media [3]. Most satellites receive dynamic instructions on how to execute their mission in orbit, and planning the exact sequence of tasks is critical to the efficiency and sustainability of the project [4]. Algorithmic optimization solutions have been suggested [5]–[7] as a remedy to the planning problem, and with the rise of quantum technologies, there is a need for exploring how quantum computing can improve the time complexity or the quality of these solutions. This work focuses on planning the mission of Earth-orbiting imaging satellites using quantum machine learning and optimization. Specifically, it explores the use of near-term quantum technologies to improve the effectiveness of the solution, today. Similar approaches such as the contribution of Ref. [8] suggested an algorithm to overcome the scheduling task using quantum annealers. Ref. [9] reviewed the literature and found that although there existed innovative developments, none revealed any practical advantage over classical approaches. The current work explores the interplay between classical and gate-based quantum computing algorithms. The practical advantage of employing this hybrid approach was shown in our earlier contributions [10]–[12].

In general, space mission planning can be a computationally hard problem [4] to solve; in large-scale missions, the size of the problem requires a prohibitive amount of computational resources. Finding an efficient plan requires the optimization of movement and the re-ordering of tasks to maximize the number of total completed tasks. In this work, each task is a request made to the satellite to capture an image of the surface of the Earth, and the aim is to maximize the number of images taken, given a list of all requests and a total available time. In this work, the imaging satellites orbit the Earth exactly on the Earth's terminator, which is the line separating Earth's sun-lit areas from the dark ones. In 24 hours satellite orbits approximately 15 times around the Earth, which leads to 15 orbits shown schematically in Fig. 4(a). To capture the image, the satellite must continuously aim the camera at the target area for a period of time, known as the acquisition slot. Each requested image has an allocated data take opportunity (DTO) window. To accomplish a request, the satellite must point in the appropriate direction within this window for the entire period of acquisition. To aim at an area on Earth, the satellite needs to rotate its camera to point in the desired direction. The latter movement introduces a time delay that when added to the acquisition time could limit the overall agility of the satellite in covering multiple areas. The efficient satellite mission plan will be able to choose the order of the acquisition requests to maximize the number of completed requests.

Discussed in this paper are the benefits of the optimization and reinforcement learning algorithms over a greedy baseline algorithm; the unique advantage offered by the potential of quantum computing; and a demonstration of how quantum methods can be applied to improve machine learning and optimization models. Specifically, the novel quantum rein-

[§]Corresponding author, e-mail: ame@terraquantum.swiss forcement learning approach shown in Sec. III-D4 offers a completion rate of 98.5% on highest-priority requests in a multi-satellite system. The model hybridizes the AlphaZero approach in Ref. [13] and is trained on the QMware quantum cloud [14]. Sec. II introduces the practical problem setup in more detail, including the details on the data formatting in Sec. II-A, re-ordering the requests in Sec. II-B, relaying algorithm in Sec. II-C, and on solution chaining in Sec. II-D. Sec. III offers three algorithms and their respective results: Sec. III-B establishes a baseline greedy algorithm, and Sec. III-C and III-D discuss the optimization and reinforcement learning algorithms. Finally, Sec. IV provides a summary of the results.

II. Satellite mission planning

Each satellite's orbit is constrained to the Earth's terminator, and each satellite can only rotate at a maximum of one degree per second to orient itself appropriately to capture images within the acquisition window. Furthermore, the DTO duration for each area to be captured is defined by the arc ranging within 45° (referred to as the depointing angle) from the apex point, which is the point directly above the center of the request on Earth. As a result of the width of these arcs, multiple requests can have substantial overlap in their acquisition windows. Furthermore, while coordinates for image requests are provided in latitude and longitude, the satellite coordinates support an Earth-centric inertial (ECI) format, which provides more support for spatial location. Finally, three datasets are tested in this work, including two single-satellite sets of 50 and 462 requests and a two-satellite system with 2000 requests.

A. Data Formatting

Two information sets were used in the preparation of this paper:

- information about the satellite motion, including:
  - the orbit number,
  - time stamp, and
  - satellite position and velocity in Cartesian ECI coordinates; and
- information about acquisition request, including:
  - request ID,
  - request priority ranging from 1 to 4, where 4 denotes the lowest priority,
  - start and end times of the DTO window of the request,
  - the coordinates of the start and end of the median line,
  - satellite ID, and
  - Boolean values indicating the progress of the acquisition.

Fig. 1 demonstrates the satellite movements during acquisition, as well as the data that has to be tracked during the capture, such as the angles of acquisition, the points at which the DTO begins and ends, and the median coordinates.

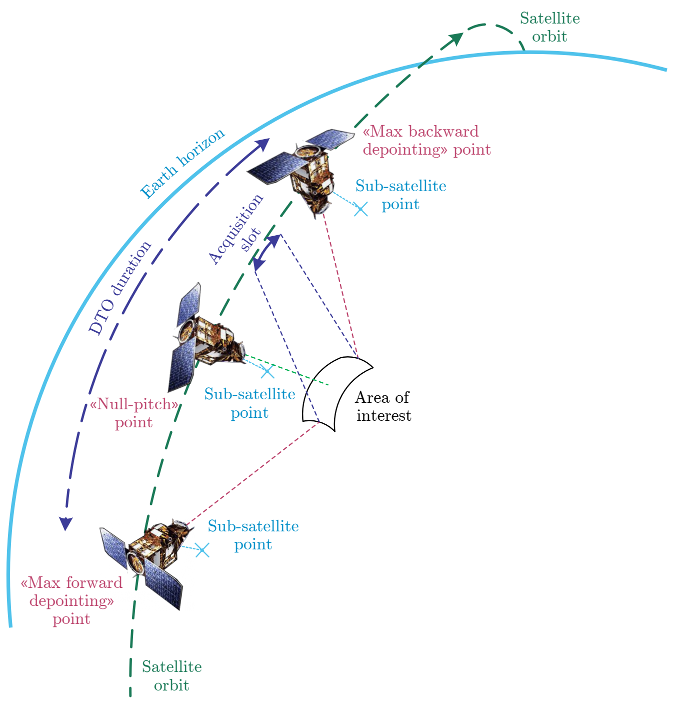


Fig. 1. The trajectory of a satellite orbiting on the Earth's terminator and across a DTO window. At the first and third positions on the orbital line, the satellite is at the ends of the request's DTO window, as its depointing angle is maximized at 45°. At its second position, it is at the apex point, directly above the request location.

B. Request Priority Ordering

By considering the priority associated with each request, it is possible to express this objective as first maximizing the completed requests in the order of priority; the highest priority requests are considered first, and only upon their completion are the lower priority requests considered. The exact quantification compares the number of higher priority requests accomplished and moves on to the next priority in case of equality, but this work judges the algorithms by their completion rate performance purely on the high-priority π_1 request.

C. Relaying Algorithm

To move from one request to another the satellite must rotate according to the relaying algorithm. To compute this rotation, there are two points of interest known as the final median points of the first request and the initial median point of the second request. These points, respectively, signify the time at which the first request was completed and the time at which the next request will begin. For a given transition, the relaying algorithm operates in two steps: 1) it computes the relative positions of each median point with respect to the satellite and 2) calculates in degrees the angle between these vectors. Assuming that the satellite rotates at a constant speed of one degree every second and that the Earth is a perfect sphere, the resultant angle can be approximated as the total relaying time in seconds.

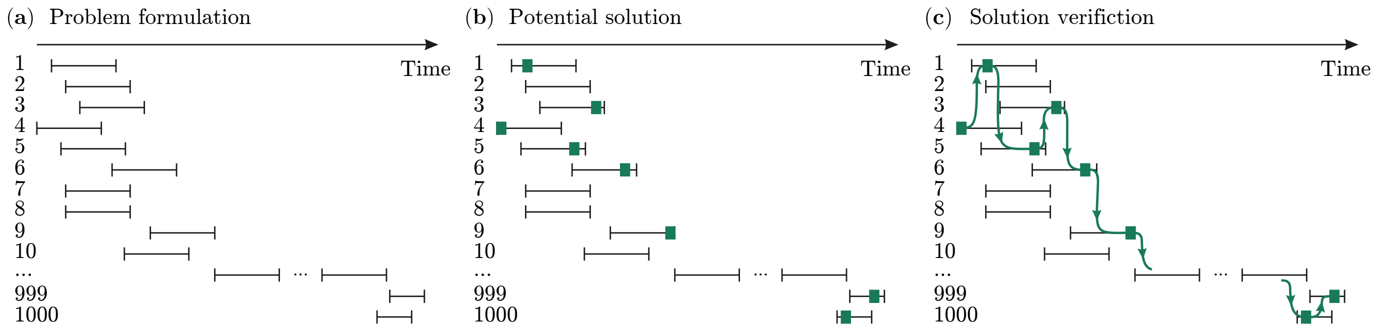


Fig. 2. (a) Visualization of DTO windows (horizontal bars) over a horizontal axis of time and a vertical axis indicating request index. (b) Visualization of solutions (green rectangles). (c) The chaining of requests based on their DTO windows and completion windows. As is evident, chaining provides a roadmap to connect all completed requests and map out the order of requests.

D. *Solution Chaining*

Solution chaining considers the DTO windows, overlaps, and the length of the windows compared to an estimate of an acquisition time. DTO windows and possible solutions within those windows are then visualized over an axis of time - see Fig. 2b. The most efficient preliminary method for this setting is to sort the requests by the start of their DTO windows and subsequently map the acquisitions.

Next, the individual requests `chained`: the intermittent time between captures is calculated using the relaying algorithm, after which the algorithm ensures sufficient time for the relaying movement between the completion of each pair of consecutive requests, ensuring that a sequence of requests can be executed in the provided time. Chaining enables the suggested solutions within the DTO windows of each request, which are otherwise independently scattered, to be connected together into one comprehensive solution for satellite movement.

Shown in Fig. 2(c) is a visualization of the chaining process between multiple requests over time.

III. Results: Quantum Algorithms for Satellite Mission Planning

Fig. 3 shows a summary of the algorithms presented in the paper.

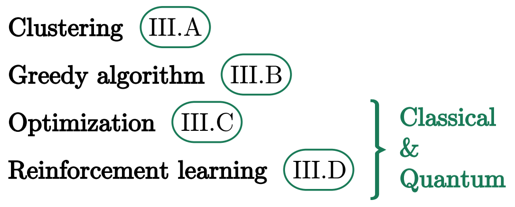


Fig. 3. Algorithms that are presented in this paper.

A. *Clustering for Pre-processing*

To deal with increasing complexity in problems is to reduce the size of the dataset and deal with fewer parameters or data points at once. Therefore, clustering is an extremely useful method to stratify the data by similarities and subsequently shorten the number of calculations necessary in the overall program. The simplest method to cluster the dataset is by sorting the samples based on a single feature of the data, a method known as bunching. The bunching algorithms are explained in App. A, but the focus of this section is on unsupervised-learning clustering methods such as the K-means algorithm [15], [16].

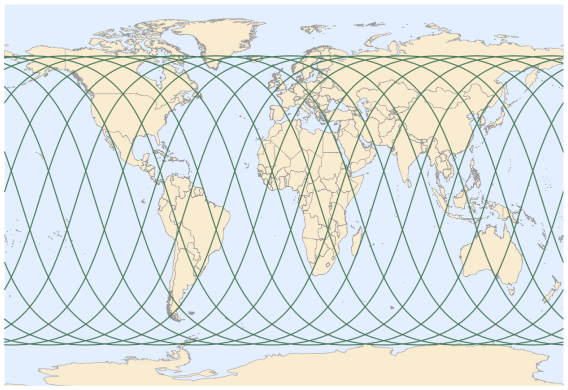
This algorithm is a well-known clustering algorithm and its usage of physical distances in creating clusters makes it a natural fit for the Space Mission Planning problem. This algorithm is an iterative algorithm, meaning that it runs multiple iterations of the same steps and finally converges at a solution. The first step of this algorithm is the assignment stage; once k random points are initialized to represent the k cluster `centroids`, each point in the dataset is assigned to the closest centroid. Once this is complete, the values of all points assigned to each centroid k_n are averaged, and the value of k_n is updated to the newly obtained mean value. Once these two steps are completed, they are repeated until the centroids no longer shift between two iterations (generally to an error threshold), which is when the final clustering for the dataset is attained.

B. *Greedy algorithm*

The classical greedy algorithm is used to provide a reference solution. First, the orbital data and request data are separated by satellite, and then the requests are clustered. In each cluster, the algorithm begins at the satellite's given start time, and increments by 1 second until it enters the DTO window of the first request. Once the time stamp enters a DTO window the algorithm checks if there is enough time left in the window to complete a request. If so, it completes the request, increments the time by the sum of the relaying time and the acquisition time, adds the request ID to a list of completed IDs, and moves on to the next request. In the event of multiple requests being available at a timestamp, the algorithm chooses the highest priority request. If there is not enough time to complete a request, it simply moves on to the next request and repeats the process. Once the final request is completed, or the timestamp exceeds the final DTO window, the algorithm outputs the percentage of completed requests of each priority.

An important limitation of the greedy algorithm is its bandwidth for anticipation, but it provides a baseline for

(a)



(b)

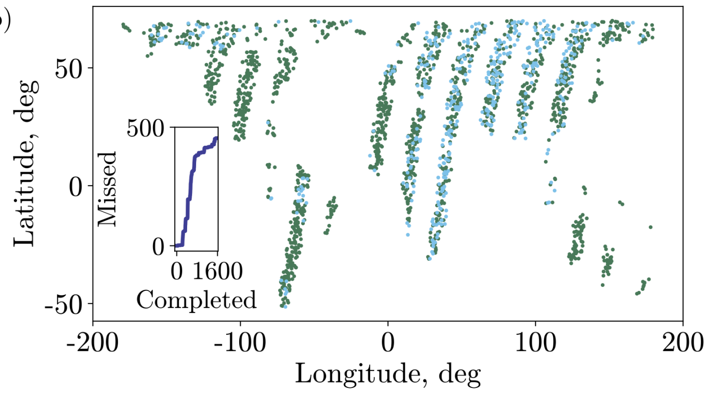


Fig. 4. (a) Schematic view of 15 satellite orbits performed within approximately 24 hours. (b) The results of the greedy algorithm are shown on the map. The task contains 2000 requests that are planned for acquisition using 2 satellites. Green points correspond to completed requests, whereas blue points are missed requests. (Inset) Progression of completed compared to missed requests.

TABLE I
RESULTS FOR THE GREEDY ALGORITHM BY PRIORITY, RUN OVER TWO SEPARATE DATASETS.

Priority	**1sat/462req**	**2sat/2000req**
π_1	99.2%	75.8%
π_2	88.6%	36.1%
π_3	80.7%	19.2%
π_4	74.0%	18.2%

TABLE II
RESULTS OF THE GREEDY ALGORITHM USING CLASSICAL K-MEANS CLUSTERING.

	KMeans + Greedy	
Priority	**1sat/462**	**2sat/2000req**
π_1	99.1%	63.6%

comparison. Its simplicity allows it to run faster than other models, making it a good benchmark for data preprocessing. The results in Fig. 4(b) and Tab. I show the greedy algorithm struggles as the complexity of the input data increases. Therefore, in order to solve this problem, algorithms that can handle high levels of complexity without compromising accuracy are required.

C. Optimization Methods

As discussed earlier, one viable way to improve the results of an algorithm is to break the data into smaller clusters to reduce the amount of necessary processing power. However, it is not out of the question to attempt to build a fundamentally more powerful algorithm, such as an optimization model. Optimization problems are problems that involve information and formulations including graphs, movement patterns and permutations, and multiple viable solutions, out of which one ideal solution must be determined by the metrics and constraints of the problem. As such, optimization methods are well suited for mission planning problems akin to the one at hand, which aims to map out the best possible course of action for a system of satellites seeking to maximize the number of completed requests. Optimization, however, is expensive in terms of both time and computation; consequently, the potential of harnessing the quantum advantage for optimization problems could be especially valuable in discovering and delivering a large speedup for mission planning problems.

Some research for optimization for space mission planning currently exists. For instance, in Ref. [17], the problem of Earth observation from a satellite (EOS) is investigated, focusing on obtaining images of certain areas of the Earth's surface related to customer requests. An optimization approach for daily photo selection (DPSP) for EOS is proposed. DPSP is related to operational management and planning processes, where each photo ordered by the client brings profit, but not all requests can be satisfied due to some physical and technological limitations. The objective is therefore to select a subset of queries for which the profit is maximized, and the proposed algorithm is based on the metaheuristic ant colony optimization algorithm (ACO). Examples based on real data are used as reference problems. The results of the calculations show that the proposed algorithm is able to generate competitive and promising solutions.

The most well-known quantum-friendly optimization method is the quadratic unconstrained binary optimization (QUBO) model, which encapsulates the set of all optimization problems with the following attributes:

- all variables are treated as binary objects
- all variables are either linear or quadratic products of linear variables
- constraints while not necessarily absolute, are enforced accordingly by use of the reward function

For the space mission planning problem, a few different formulations derived from QUBO are researched as possible solutions. Despite these formulations being NP-hard problems, the algorithms can be potentially accelerated quadratically with the use of quantum computation. These solutions each hold their own benefits and drawbacks but can yet serve as viable solutions to the satellite optimization problem.

1) Integer Optimization Model: The integer optimization model is conceptualized by encoding the desirable outcome of the solution in the cost function, which should be maximized or minimized on the lattice of integer points of the special feasible subset in the multidimensional space. This feasible subspace is defined by adding the constraints of the

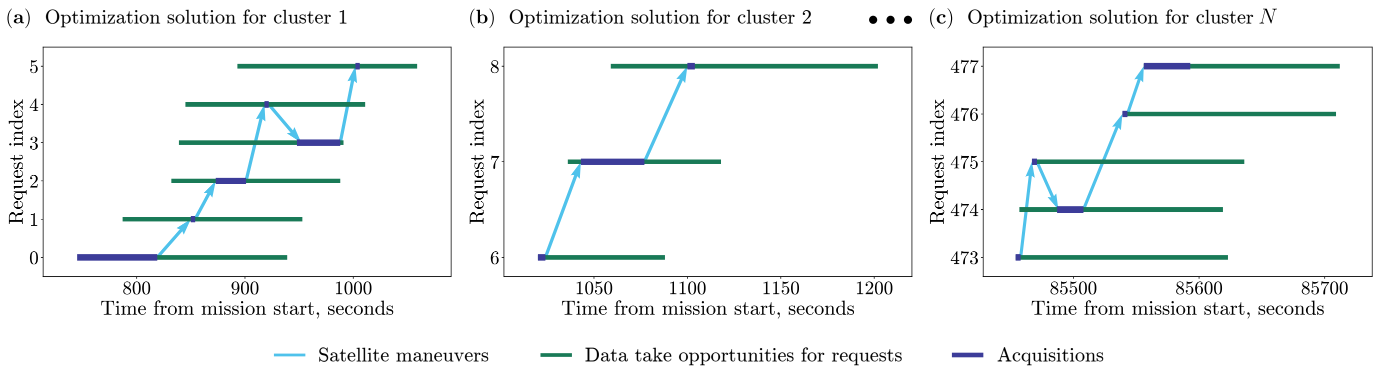


Fig. 5. Chaining results of classical integer optimization algorithm within clusters of data points.

satellite mission planning problem. One of the most powerful approaches to that kind of problem is the `branch and cut` algorithm [18], used in solvers. In addition, the following functions are incorporated to streamline the necessary calculations for the algorithm to stay within the constraints. Incorporation of these functions is crucial to developing viable solutions as these functions, built over the Orekit space flight dynamics library [19], are the key to accurate space mission planning.

- *Attitude Pointing:* given the satellite, timestamp, and a pair of longitude and latitude coordinates, this function returns the attitude (roll, pitch, and yaw) angles of the satellite when it is pointed towards the provided coordinates.
- *Acquisition Duration:* given the starting and ending median coordinates of any request, this function returns the amount of time, in milliseconds, that the satellite would take to acquire the image.
- *Maneuver Duration:* given the beginning and ending attitude angles (roll, pitch, and yaw), this function returns the amount of time necessary for the satellite to complete the maneuver from one attitude to the other.
- *Read Ephemeris:* given a JSON file and a satellite ID, this function returns the satellite orbital information (directional speed, directional position, timestamps, etc.) in the form of an Orekit Ephemeris object, which is then used to perform calculations for other functions using the Orekit library.

Upon incorporation of the above functions, revision of the constraints on this integer-based model, and modification of the clustering algorithm to generate a greater number of smaller clusters, the runtime of the algorithm was maintained despite the significant increase in complexity. Given a set of requests, the variables central to this model include the priority of each request, the acquisition duration of each request, the start and the end of the DTO windows, and also the location of the median points of the requests. From this, the algorithm calculates for each reasonable pair of the requests inside the small cluster all possible start times for the 1st request in a way that allows the relaying maneuver to the 2d request. This is achieved through simple iteration, using the Orekit functions described above: an initial value for the relaying maneuver duration is set to 1 second. Then, the maneuver start and end angles are calculated with the attitude pointing function. Afterwards, if the rotation time between the obtained angles via the maneuver duration function is not equal to or greater than 1 second, and the DTO limitations are not violated, meaning the relaying time is equal to 1 second. Otherwise, the duration is increased by 1 second, and this procedure is repeated until either the duration is appropriate or the DTO is violated. The result of these computations, t_{min}, is treated as the minimum relaying maneuver time.

Given a set of requests $f \in \mathbf{F}$, the priority of each request is denoted as π_f, and the acquisition duration as τ_f. The start of the DTO window (release time) for request f is represented as r_f, and the deadline of the DTO window is represented as d_f. In addition, $\mathbf{Q} \subset \mathbb{N}$ indicates the order of accepted requests. As a result, index $q \in \mathbf{Q} = \{0, \ldots, Q-1\}$ demonstrates the position in the queue of requests in which request f is accepted. The discretized rotation of the satellite at the inception of the acquisition of request f is indicated by $\alpha_f^{start} \in \mathfrak{A}_f^{start}$. Similarly $\alpha_f^{end} \in \mathfrak{A}_f^{end}$ is the discretized rotation of the satellite at the conclusion of the acquisition of the request. However, all possible inceptions $b_{f\alpha}$ for requests can be streamlined in order of increasing time. Thus all the possible angles will correspond to these moments of time. The set of possible pairs is defined as $\mathbf{L}$, such that the maneuver $f_1 \to f_2$ is possible; i.e.:

$$\exists b_{f_1\alpha_{f_1}}, b_{f_2\alpha_{f_2}} : b_{f_1\alpha_{f_1}} + t_{min}(b_{f_1\alpha}, f_1, f_2) + \tau_{f_1} = b_{f_2\alpha_{f_2}}, \tag{1}$$

where $t_{min}(t, f_1, f_2)$ is the minimum amount of time required for a satellite to rotate from its position at the end of f_1 at the moment t to the starting point of f_2. For all $(f_1, f_2) \in \mathbf{L}$, the sets $\mathfrak{B}_{f_1f_2}^{start} \subseteq \mathfrak{A}_{f_1}^{end}$ and $\mathfrak{B}_{f_1f_2}^{end} \subseteq \mathfrak{A}_{f_2}^{start}$ and mapping $M_{f_1,f_2} : \mathfrak{B}_{f_1f_2}^{start} \to \mathfrak{B}_{f_1f_2}^{end}$ are defined such that $M_{f_1,f_2}(\alpha_1) = \alpha_2$, if $b_{f_1\alpha_1} + t_{min}(b_{f_1\alpha_1}, f_1, f_2) + \tau_{f_1} = b_{f_2\alpha_2}$.

The binary variable x_{fq} is introduced such that:

$$x_{fq} = \begin{cases} 1, & \text{if } f\text{-th request is started in the } q\text{-th slot,} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

In addition, the variable $y_{f\alpha}$ is also introduced, where:

$$y_{f\alpha} = \begin{cases} 1, & \text{if angle } \alpha \text{ is the start angle for request } f, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Finally, the indicator variable $\kappa_{f_1 f_2}$ shows if requests were completed successively one after another:

$$\kappa_{f_1 f_2} = \begin{cases} 1, & \text{if } f_2\text{-th follows right after the } f_1\text{-th}, \\ 0, & \text{else.} \end{cases} \quad (4)$$

The cost function is defined as:

$$\sum_{f \in \mathbf{F}} \left(\sum_{q \in \mathbf{Q}} J_f x_{fq} - \sum_{\alpha \in \mathfrak{A}_f^{start}} \gamma_{f\alpha} y_{f\alpha} \right) \to \max, \quad (5)$$

where J_f is the weight of request f, and $\gamma_{f\alpha}$ is a coefficient needed for penalizing time consuming solutions. $J_f = 1$ was selected for the lowest priority requests in each cluster, and for the higher priority requests, the weight is greater than the sum of all lower priority requests' weights by one, since each higher priority request is valued more than any amount of lower priority requests. The coefficients $\gamma_{f\alpha}$ range from 0 to $\frac{1}{Q}$ for the following definition:

$$\gamma_{f\alpha} = \frac{b_{f\alpha}}{Q(d_f - r_f - \tau_f)}. \quad (6)$$

It guarantees that $\sum_{\alpha \in \mathfrak{A}_f^{start}} \gamma_{f\alpha} y_{f\alpha} < 1$, and consequently, the completion of the lowest priority request will be more important than the particular order of requests, but the earliest possible completion of each request is preferable.

As was the case with the other optimization model, more than one request cannot be executed in the same order, and each request should be completed not more than once:

$$\forall f \in \mathbf{F}, \sum_{q \in \mathbf{Q}} x_{fq} \leq 1 \quad (7)$$

$$\forall q \in \mathbf{Q}, \sum_{f \in \mathbf{F}} x_{fq} \leq 1 \quad (8)$$

Any completed request is started with one particular possible angle:

$$\sum_{\alpha \in \mathfrak{A}_f^{\text{start}}} y_{f\alpha} = \sum_{q \in \mathbf{Q}} x_{fq} \quad \forall f \in \mathbf{F}. \quad (9)$$

All requests are completed one after another without empty slots in line, until the satellite stops:

$$\sum_{f_1 \in \mathbf{F}} x_{f_1(q-1)} \geq \sum_{f_2 \in \mathbf{F}} x_{f_2 q} \quad \forall q > 0. \quad (10)$$

In order to evaluate the variable $\kappa_{f1 f_2}$ appropriately, the following system of linear equations is introduced, and each excludes the impossible values of $\kappa_{f_1 f_2}$. Firstly, if requests f_1 and f_2 were completed straight one after another, then $\kappa_{f_1, f_2} = 1$:

$$\kappa_{f_1 f_2} + 1 \geq x_{f_1, q-1} + x_{f_2, q} \quad \forall f_1, f_2 \in \mathbf{F}, q > 0. \quad (11)$$

Each request is followed by not more than one request and each request follows after not more than one request:

$$\sum_{f_1 \in \mathbf{F}} \kappa_{f_1, f_2} \leq 1 \quad \forall f_2 \in \mathbf{F}, \quad (12)$$

$$\sum_{f_2 \in \mathbf{F}} \kappa_{f_1, f_2} \leq 1 \quad \forall f_1 \in \mathbf{F}. \quad (13)$$

If one request follows another one, then each of these requests was completed in some order:

$$\kappa_{f_1 f_2} \leq \sum_{q > 0} x_{f_2, q} \quad \forall f_1, f_2 \in \mathbf{F}, \quad (14)$$

$$\kappa_{f_1 f_2} \leq \sum_{q < Q-1} x_{f_1, q} \quad \forall f_1, f_2 \in \mathbf{F}. \quad (15)$$

With the use of predefined mapping M_{f_1, f_2}, if the request f_1 starts with angle α_1, and the request f_2 is the next one, then the next acquisition start angle is fixed:

$$y_{f_2 \alpha_2} + 1 \geq \kappa_{f_1, f_2} + y_{f_1 \alpha_1} \quad \forall \alpha_1 \in \mathfrak{B}_{f_1 f_2}^{start}, \quad (16)$$

where $\alpha_2 = M_{f_1, f_2}(\alpha_1)$.

Additionally, maneuver $f_1 \to f_2$ is possible only with particular initial angles for f_1, otherwise, the satellite will not have enough time to finish the acquisition of the request f_1 and move to the f_2:

$$\sum_{\alpha_1 \in \mathfrak{B}_{fg}^{start}} y_{f_1 \alpha_1} \geq \kappa_{f_1 f_2} \quad \forall (f_1, f_2) \in \mathbf{L}. \quad (17)$$

On the other hand, if $f_1 \to f_2$ is an impossible transaction, then

$$\kappa_{f_1, f_2} = 0 \quad \forall (f_1, f_2) \notin \mathbf{L}. \quad (18)$$

The final constraint fixes the acquisition start angle for the 1^{st} request in a queue as the earliest possible angle:

$$y_{f\alpha_0} \geq x_{f0} \quad \forall f \in \mathbf{F}, \quad (19)$$

where α_0 is an angle, corresponding to the beginning of the DTO for request f.

In contrast to the greedy algorithm, which considered the requests in the order of open DTO windows, this model is more intelligent in finding the best path to fit the maximal number of requests. In the end, the π_1 requests reached 98.1% completion by use of the Gurobi solver [20]. The solution obtained through optimization is partly depicted in Fig. 5. Furthermore, the clusters are connected to each other, calculating the minimum relaying maneuver time from the last request of the previous cluster to that of the next. This is illustrated in Fig. 5 as cutting the beginning of every DTO from the start if it proves impossible to rotate towards the request in this period. After this procedure, the next cluster can be treated as independent.

One of the greatest benefits of this model is its compatibility with both near-term and long-term quantum technology. As a linear optimization model that uses a grid of binary parameters, it can be transformed into QUBO, as it is shown in App. C, and fit rather quickly to the quantum Ising model, a model containing arrays of qubits in a grid where their spin states depend on their neighbors. Furthermore, as this model functions akin to a minimization problem, it would be extremely efficient to run on a quantum annealing machine [21], which could solve optimization problems by slightly changing the Hamiltonian from a given initial state with a known minimum to a new state, representing the optimal solution. However, it is complicated to use the satellite mission planning problem in the QUBO form via currently available classical or quantum devices. For example, both D-Wave's Leap Hybrid solver [22] and Gurobi have difficulties in solving even a small cluster with 4 requests over a time limit of 1 minute, but the whole solution for the 2000 requests and 2 satellites dataset can be obtained in less than 3 minutes using linear programming.

D. Reinforcement Learning

Reinforcement learning (RL) is a paradigm of machine learning where an agent interacts with some environment and trains by informed trial and error. The RL training algorithm uses reward functions to assign value to the actions of the agent in any state of the environment. Generally, a state can have constraints and features. The RL agent can use a policy model to decide on an action given a state, which subsequently affects the environment and transforms the state. If the resultant state of the environment is engineered by the data scientist to contain a positive reward, the policy model is trained to take the appropriate action to maximize the probability of its achieving that reward. The Environment is a function of a triplet of variables (S, A, P), where S is a state space, A is an action space, and P is a transition function. When the reward function, r, is factored in, a Markov Decision Process (MDP) is generated with the property (S, A, P, r), $r : S \times A \to \mathbb{R}$. The Agent starts from state s_0 and takes action a_0, for which the reward r_0 is obtained in each step of training and subsequently trains by producing trajectories $T := (s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \cdots)$.

1) RL Environments: Two environments are developed for this AEOS task: satellite- and request-centred environments. For each environment, requests are sorted by their DTO open time.

The satellite-centred environment: the agent views the problem from the perspective of the satellite and considers the 100 closest (by DTO open time) data points for each satellite, each of which utilizes 10 data parameters. Moreover, three additional features were also made available to the agent: the current time for the satellite and the latitude and longitude of the starting point of the last completed request, creating a total observation space of size 1003. The number of nearest requests is a variable that depends on task size. A Boolean flag is one of the features kept for each request, marking each data point as either complete or incomplete to avoid redundancy, and the observation space is made to only consider requests that are marked as incomplete in order to expedite the computations required by the agent, which is built as a neural network.

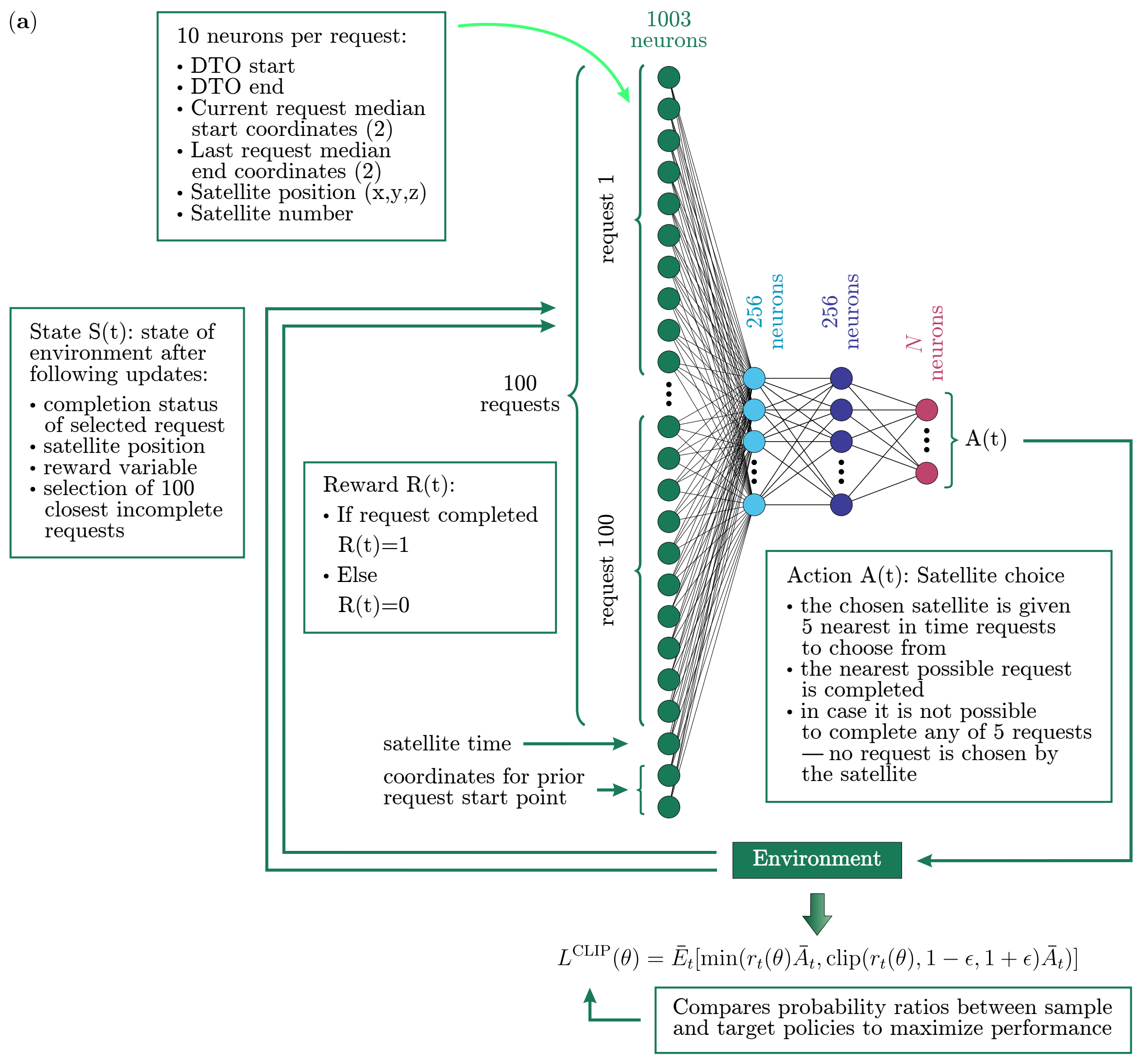
Once the request is selected, the agent is given a reward of 1 if that request is completed and 0, otherwise. With this process recurring, the agent attempts to complete as many requests as possible until the time of the satellite is greater than the DTO windows of all remaining requests, meaning they can no longer be completed. The agent works with each satellite in turn and predicts which request must be done. The data used in this environment was artificially generated. *The request-centred environment:* at each step, the agent views the problem from the perspective of a request, deciding which satellite is best suited to complete it. The 5 nearest request options are determined by DTO open time for each satellite and the request to execute is chosen by the minimum request execution time. This minimum time is calculated as the sum of the satellite timestamp, maneuver duration and the acquisition time due to solution chaining and must be less than the request DTO end time for it to be completed. This procedure is then iterated with the next batch of 5 nearest satellites.

2) Proximal Policy Optimization: The Proximal Policy Optimization (PPO) algorithm, shown in Fig. 6 and first introduced in Ref. [23], was implemented to provide a mission planning policy. In RL, a policy is an operation that maps an action space to a state space. The agent learns the best course of action for each situation by calculating the policy gradients. In other words, the agent uses gradient descent to calculate the expected value of each action at a certain state space and determine which action has the likelihood of the highest reward. The equation for the PPO algorithm is as follows:

$$L(\theta) = \bar{E}_t[\min(r_t(\theta)\bar{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\bar{A}_t)] \quad (20)$$

where $\bar{E}_t$ is the current expected value of the policy, θ is the policy parameter, ϵ is the hyperparameter, $\bar{A}_t$ is the estimated advantage provided by the PPO at time t, and $r_t(\theta)$ is the importance sampling ratio. This ratio, derived from the Monte Carlo sampling methods [23], denotes the ratio of probabilities under both the old and the new policies. By keeping the value of ϵ small, the model ensures that the update on the policy at each increment is not too large; as a consequence, the learning done by the model stays relevant.

3) Hybrid-quantum PPO: Inspired by the quantum advantage shown in Refs. [10]–[12], [24], [25], this section investigates the utility of a hybrid-quantum neural network as a policy model for RL. Fig. 6(b) illustrates the specific policy network within a hybrid MLP model. In the quantum-classical network, the output of the classical neural network is used as inputs to a parametrized quantum circuit (PQC). PQCs are quantum circuits that make use of parametrized quantum gates [26], [27] such as Pauli rotations to encode data, x, and trainable parameters, θ. The four-qubit PQC used in this work consists of three significant components: variational, data encoding, and measurement. The variational layer is composed of a layer of Pauli-X rotations encoding four trainable parameters followed by a layer of ring-shaped CNOTS for entanglement, the data encoding layer embedded four data features in parallel using Pauli-Z rotations, and the measurement layer comprised of two Z-basis measurements on the first two qubits. In sequence, four qubits were initialized in the ground state, and then a variational layer with randomly initialized parameters was appended followed by eight repetitions of encoding and variational layers. Each of the eight encoding layers encoded four features of the dataset, which created a lattice of 32 features across the four qubits. Finally, the measurement layer was included to produce two classical real-valued outputs. Fig. 9(a) shows this advantage in practice: the hybrid quantum neural network achieves a higher reward in


(a)
10 neurons per request:
• DTO start
• DTO end
• Current request median start coordinates (2)
• Last request median end coordinates (2)
• Satellite position (x,y,z)
• Satellite number
1003 neurons
request 1
100 requests
request 100
256 neurons
256 neurons
N neurons
A(t)
State S(t): state of environment after following updates:
• completion status of selected request
• satellite position
• reward variable
• selection of 100 closest incomplete requests
Reward R(t):
• If request completed R(t)=1
• Else R(t)=0
Action A(t): Satellite choice
• the chosen satellite is given 5 nearest in time requests to choose from
• the nearest possible request is completed
• in case it is not possible to complete any of 5 requests — no request is chosen by the satellite
satellite time
coordinates for prior request start point
Environment
$L^{\mathrm{CLIP}}(\theta) = \bar{E}_t[\min(r_t(\theta)\bar{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\bar{A}_t)]$
Compares probability ratios between sample and target policies to maximize performance

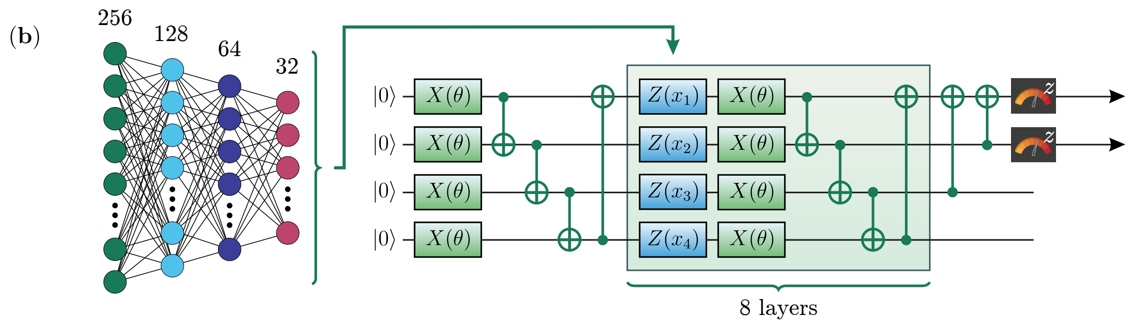


Fig. 6. (a) The reinforcement learning PPO model used to solve the mission planning problem. The state of the model encompasses the data of 100 requests, which are fed into the MLP agent to output an action (one request that is selected to complete), which then feeds into the environment (clip equation), generates an appropriate reward, and updates the state. (b) Quantum-hybrid reinforcement learning model. A quantum circuit (left) is added to the beginning of the MLP Agent in the RL model (right) to incorporate quantum computation into the neural network.

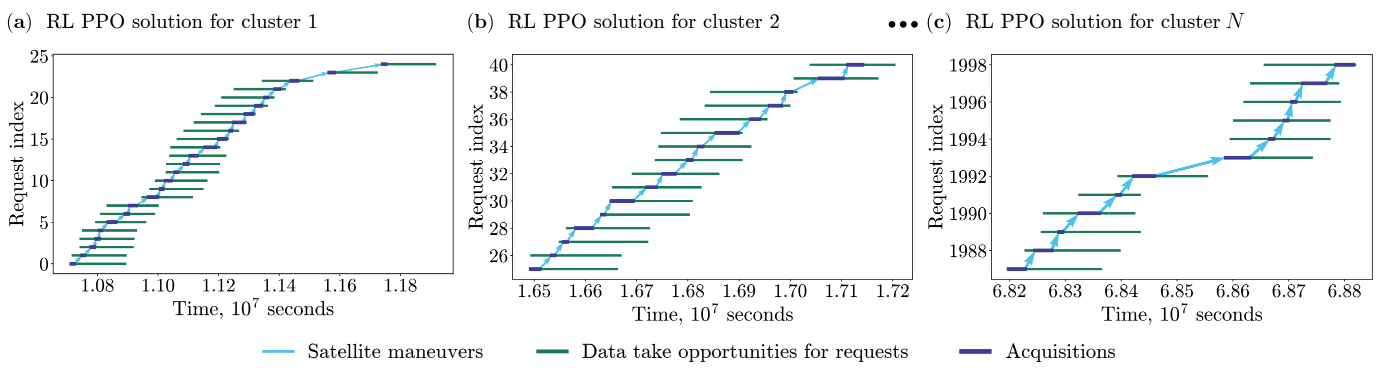


Fig. 7. Visualization of the chaining process used to complete requests within each cluster of the PPO algorithm.

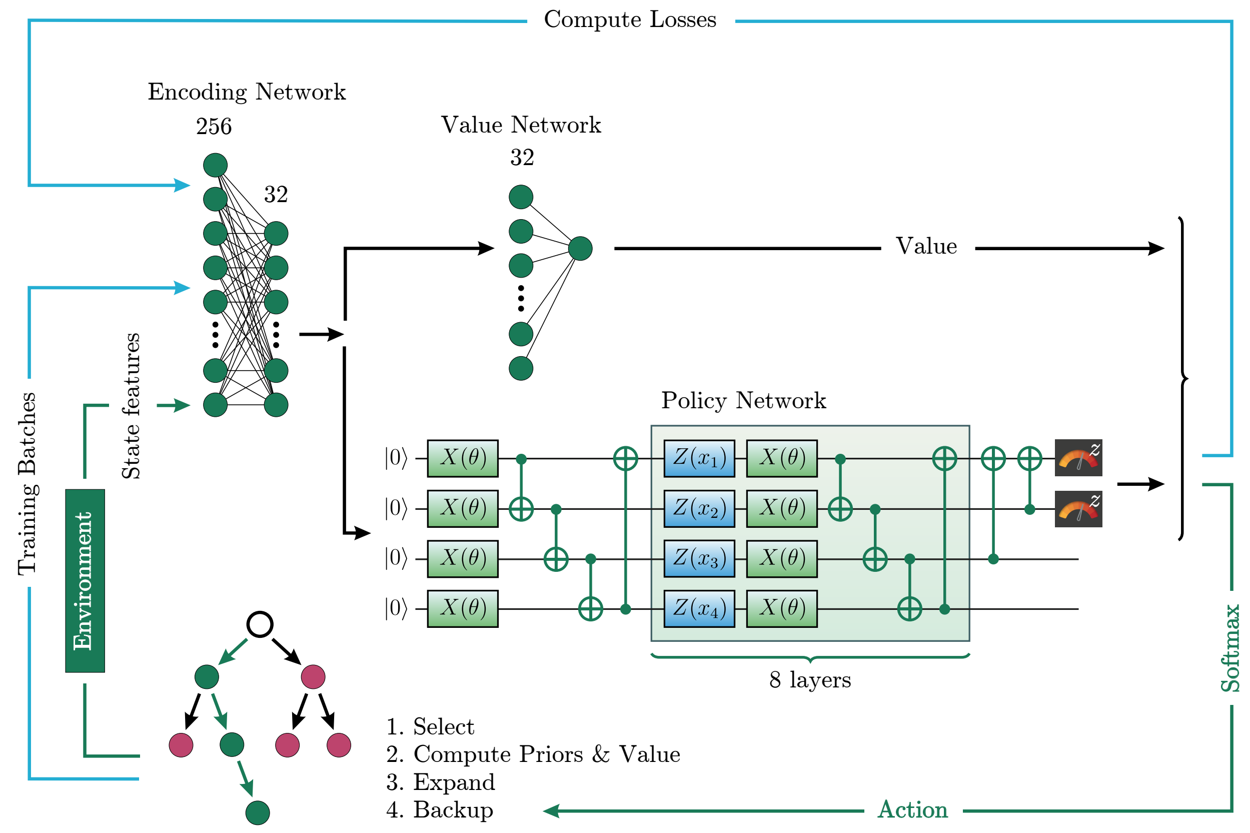


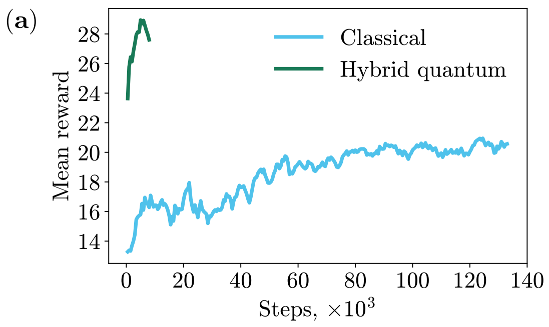
Fig. 8. The hybrid AlphaZero architecture used to achieve a near-optimal solution to the satellite mission planning.

only 8k steps which remains inaccessible to a classical network of the same complexity even after 110k environment steps.

4) Hybrid AlphaZero: The optimal results are expected from picking the best candidate from part of the solution pipeline. Specifically, in Sec. III-D, it was shown that reinforcement learning is a powerful algorithm that can be boosted in training and solution optimality through hybridization with quantum models. This section shows hybridized AlphaZero [13].

The classical AlphaZero was a reinforcement learning (RL) algorithm developed by DeepMind that showed promise in solving difficult RL problems, such as chess, shogi, and go [13]. AlphaZero uses a computational tree of environment states whose values and probabilities are determined from the outputs of two networks: value and policy networks. The AlphaZero implementation in this work is composed of four parts: a Monte Carlo tree search (MCTS), an encoding network, a policy network, and a value network.

The foundation of this model is akin to a Monte Carlo Tree Search (MCTS) model, which is used primarily for path prediction problems and board games based on strategy [28], [29]. The four major components of MCTS are selection, expansion, simulation, and backpropagation. At the start, the model begins at the root of the tree and selects optimal child nodes until it reaches a leaf. Once it reaches the leaf, it expands the tree, creating a new child node. Then, the model simulates the remainder of the path-finding process from the newly created node, and finally backpropagates with the newfound



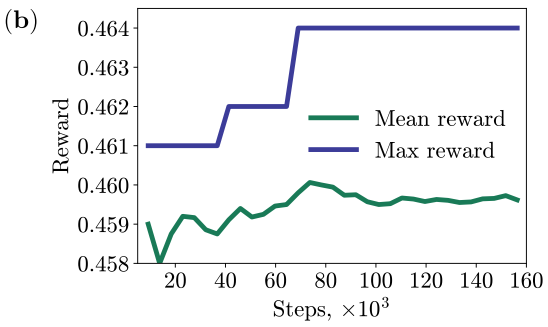


Fig. 9. (a) The rewards achieved by the classical and hybrid PPO RL agents. The hybrid model reaches a higher reward at a significantly improved learning speed. (b) The agent reaches a maximum normalized reward of 0.464 after 70,000 steps of the simulation. This is in comparison to a total possible reward of 0.471, translating to an accuracy of 98.5%.

TABLE III
A SUMMARY OF THE PERFORMANCE OF THE ALGORITHMS ON THE 2000 REQUESTS TASK PERFORMED WITH 2 SATELLITES. THE COMPARISON DEMONSTRATES THE GREEDY ALGORITHM AS A BASELINE, THE HYBRID ALPHAZERO, AND INTEGER OPTIMIZATION MODELS.

Model	Greedy + KMeans	**RL (Hybrid AlphaZero)**	Optimization
π_1 Completion %	63.6%	**98.5%**	98.1%

information to update the hyperparameters of the tree. The hybrid AlphaZero model uses MCTS in conjunction with a parametrized quantum circuit used as its policy network. The MLP agent of the model first recommends a state action. Then the action is applied to the environment which generates a reward and an updated state. However, in this case, the environment is the MCTS model; the recommended action is simulated in the model, and then the backpropagation step of the MCTS is used to update the weights within the tree itself to refine its recommendations.

As each iteration of this loop occurs, there is also an outer loop in this algorithm. That outer loop is primarily concerned with the loss values being optimized; as a result, once the agent recommends an action, the outer loop takes training examples of a certain size and simulates the results over these batches to calculate the policy loss, value loss, and total loss. Once these values are calculated, the weights of the neural network are adjusted and fed back into the original loop, where the cycle repeats until the loss values are optimized.

Fig. 8 illustrates the inner workings of the hybrid AlphaZero model employed in this paper. The state variables are passed into an initial encoding network with 2 hidden layers of sizes 256 and 32 neurons. The information is then passed in parallel to 1) a single neuron using a fully-connected layer, and 2) a policy network in the form of a PQC. The PQC resembles the one explained and implemented in the PPO model in Sec. III-D3. Fig 9(b) shows the training performance of this model, which achieves a completion rate of π_1 = 98.5%. Presenting the highest π_1 completion rate on the 2000 requests and 2 satellites dataset of any other model explored in this paper. The completion rates of the highest-performing algorithms from each section on this dataset are displayed in Fig. III.

IV. Discussion

This work provided two classes of solutions to the scheduling problem of satellite mission planning: optimization and hybridized reinforcement learning. From each class, the best performing candidates were the integer optimization model and the hybrid AlphaZero, respectively. The dataset with 2 satellites and 2000 was used as a performance benchmark for the algorithms presented in this work. The optimization and hybrid AlphaZero algorithms achieved π_1 completion rates of 98.1% and 98.5%, respectively, while the greedy algorithm only exhibited a 63.6% completion rate. In the single satellite model, the optimization algorithm reached 100% completion on π_1, π_2, and π_3 requests and 96.2% completion on π_4 requests in 6 minutes. This work showed that by using reinforcement learning and optimization models, it is possible to improve the results of mission planning that are otherwise obtained through simple greedy models. This work presents a step towards creating quantum-enhanced solutions in the space industry.

References

[1] P. Chini, G. Giambene, and S. Kota, "A survey on mobile satellite systems," *Int. J. Satellite Communications Networking*, vol. 28, pp. 29–57, 2009.

[2] E. Dziadczyk, W. Zabierowski, and A. Napieralski, "Satellite navigation system gps," in *2007 9th International Conference - The Experience of Designing and Applications of CAD Systems in Microelectronics*, 2007, pp. 504–506.

[3] B. Jaksic, D. Miljkovic, V. Maksimovic, M. Petrovic, and B. Gvozdic, "Satellite television transmission in the world - broadcasting systems and standards," *Acta Scientiarum. Technology*, vol. 42, no. 1, p. e44957, 2020.

[4] R. Grasset-Bourdel, G. Verfaillie, and A. Flipo, "Planning and replanning for a constellation of agile earth observation satellites," in *ICAPS-11 Workshop on Scheduling and Planning Applications (SPARK-11)*, Freiburg, Germany, 2011.

[5] Y. Huang, Z. Mu, S. Wu, B. Cui, and Y. Duan, "Revising the observation satellite scheduling problem based on deep reinforcement learning," *Remote Sensing*, vol. 13, no. 12, 2021.

[6] M. Lemaitre, G. Verfaillie, F. Jouhaud, J.-M. Lachiver, and N. Bataille, "Selecting and scheduling observations of agile satellites," *Aerospace Science and Technology*, vol. 6, pp. 367–381, 2002.

[7] H. A. Khojah and M. A. Mosa, "Multi-objective optimization for multi-satellite scheduling task," *Journal of Soft Computing Exploration*, vol. 3, pp. 19–30, 2022.

[8] G. Wu, H. Wang, W. Pedrycz, H. Li, and L. Wang, "Satellite observation scheduling with a novel adaptive simulated annealing algorithm and a dynamic task clustering strategy," *Computers & Industrial Engineering*, vol. 113, pp. 576–588, 2017.

[9] T. Stollenwerk, V. Michaud, E. Lobe, M. Picard, A. Basermann, and T. Botter, "Image acquisition planning for earth observation satellites with a quantum annealer," *arXiv preprint arXiv:2006.09724*, 2020.

[10] A. Sagingalieva, A. Kurkin, A. Melnikov, D. Kuhmistrov, M. Perelshtein, A. Melnikov, A. Skolik, and D. Von Dollen, "Hyperparameter optimization of hybrid quantum neural networks for car classification," *arXiv preprint arXiv:2205.04878*, 2022.

[11] A. Sagingalieva, M. Kordzanganeh, N. Kenbayev, D. Kosichkina, T. Tomashuk, and A. Melnikov, "Hybrid quantum neural network for drug response prediction," *arXiv preprint arXiv:2211.05777*, 2022.

[12] M. Perelshtein, A. Sagingalieva, K. Pinto, V. Shete, A. Pakhomchik, A. Melnikov, F. Neukart, G. Gesek, A. Melnikov, and V. Vinokur, "Practical application-specific advantage through hybrid quantum computing," *arXiv preprint arXiv:2205.04858*, 2022.

[13] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[14] M. Kordzanganeh, M. Buchberger, M. Povolotskii, W. Fischer, A. Kurkin, W. Somogyi, A. Sagingalieva, M. Pflitsch, and A. Melnikov, "Benchmarking simulated and physical quantum processing units using quantum and hybrid algorithms," *arXiv preprint arXiv:2211.15631*, 2022.

[15] X. Jin and J. Han, *K-Means Clustering*. Boston, MA: Springer US, 2010, pp. 563–564.

[16] J. MacQueen, "Classification and analysis of multivariate observations," in *5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.

[17] S. Kilic, "An optimization approach for the daily photograph selection of earth observation satellites," *Havacılık ve Uzay Teknolojileri Dergisi*, vol. 12, 2019.

[18] A. Land and A. Doig, "An automatic method of solving discrete programming problems," *Econometrica: Journal of the Econometric Society*, pp. 497–520, 1960.

[19] L. Maisonobe, V. Pommier, and P. Parraud, "Orekit: An open source library for operational flight dynamics applications," 2010.

[20] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2022.

[21] E. Tosatti, "Optimization using quantum mechanics: quantum annealing through adiabatic evolution," *Journal of Physics A: Mathematical and Theoretical*, vol. 41, p. 209801, 2008.

[22] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky *et al.*, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, pp. 194–198, 2011.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[24] S. Jerbi, C. Gyurik, S. C. Marshall, H. J. Briegel, and V. Dunjko, "Parametrized quantum policies for reinforcement learning," *arXiv preprint arXiv:2103.05577*, 2021.

[25] A. Skolik, S. Jerbi, and V. Dunjko, "Quantum agents in the Gym: a variational quantum algorithm for deep Q-learning," *Quantum*, vol. 6, p. 720, 2022.

[26] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Science and Technology*, vol. 4, no. 4, p. 043001, 2019.

[27] M. Schuld and N. Killoran, "Quantum machine learning in feature hilbert spaces," *Physical Review Letters*, vol. 122, no. 4, p. 040504, 2019.

[28] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck, "Monte-carlo tree search: A new framework for game ai," *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 4, p. 216–217, 2008.

[29] B.-D. Lee, "Monte-carlo tree search applied to the game of tic-tac-toe," *Journal of Korea Game Society*, vol. 14, pp. 47–54, 2014.

[30] A. Pakhomchik, S. Yudin, M. Perelshtein, A. Alekseyenko, and S. Yarkoni, "Solving workflow scheduling problems with qubo modeling," *arXiv preprint arXiv:2205.04844*, 2022.

Appendix

Fig. 10 showcases the logic of the greedy algorithm from Sec. III-B.

Below are a few bunching algorithms used to cluster the data purely by sorting with respect to certain features of the dataset.

A. DTO Bunching

In this algorithm, the data is first sorted by the DTO start times. After that, the data points are clustered by DTO overlap. In other words, all entries in any given cluster share at least one portion of their DTO windows with all other requests in the same cluster. As DTO windows are defined by the time when the satellite is in a position to capture the request, this method also functions as a form of geographical clustering due to the correlation of the location and DTO times.

B. Priority Bunching

In this algorithm, the requests are simply clustered by their priority ranks (1-4) and then ordered by the DTO start times. The utility of this lies greatly in the importance of the priority rankings, as the algorithm operates on the premise of each priority rank request being worth n times a request of the next highest priority rank, where n is the order of magnitude between two priority sets (for instance, a priority 1 request would be n^2 times as valuable as a priority 3 request).

C. Bunch Sort

This algorithm puts together the two clustering algorithms above, in a sense. First, the data is sorted by DTO windows, after which it is clustered by the DTO overlap. However, after that, the data is sorted within each cluster, this time by the end of their DTO windows and then by priority index. As a consequence, data points that have the same priority in the same cluster are ranked by the end of their DTO windows, maintaining the DTO structure within the priority indexing structure.

Adiabatic quantum computers can approximately solve NP-hard problems, such as quadratic unconstrained binary optimization, faster than classical computers. Since many machine learning problems are also NP-hard, adiabatic quantum computers might be instrumental in training machine learning models efficiently in the post-Moore's law era. In order to solve problems on adiabatic quantum computers, they must be formulated as QUBO problems, which is possible to do by several techniques. In this paper, the problem is formulated as a QUBO problem, making it conducive to being trained on adiabatic quantum computers. Since all the constraints will go into the cost function with some coefficients, they must be algebraically reformulated. Below there will be examples of how these constraints change according to the QUBO formulation.

$$\forall f \in \mathbf{F}, \sum_{q\in\mathbf{Q}} x_{fq} \leq 1 \Rightarrow \left(\sum_{q\in\mathbf{Q}} x_{fq} - \frac{1}{2}\right)^2, \tag{21}$$

$$\forall q \in \mathbf{Q}, \sum_{f\in\mathbf{F}} x_{fq} \leq 1 \Rightarrow \left(\sum_{f\in\mathbf{F}} x_{fq} - \frac{1}{2}\right)^2, \tag{22}$$

$$\forall f \in \mathbf{F}, \sum_{\alpha\in\mathfrak{A}_f^{start}} y_{f\alpha} = \sum_{q\in\mathbf{Q}} x_{fq} \Rightarrow \tag{23}$$

$$\left(\sum_{\alpha\in\mathfrak{A}_f^{start}} y_{f\alpha} - \sum_{q\in\mathbf{Q}} x_{fq}\right)^2, \tag{24}$$

$$\begin{gathered}\forall q > 0, \sum_{f_1\in\mathbf{F}} x_{f_1 q-1} \geq \sum_{f_2\in\mathbf{F}} x_{f_2 q} \Rightarrow \\ \sum_{f\in\mathbf{F}} (x_{f_2 q} - x_{f_1 q-1}) + \sum_i 2^i z_i = 0,\end{gathered} \tag{25}$$

where z_i is a slack binary variable, and the number of slack variables depends on the problem. Thus, the idea of the QUBO formulation is to reduce all constraints to linear equalities, square them and then pose as penalties in the cost function. The more rigorous analysis of the slack variable implementation is provided in [30]. QUBO formulation is written in general terms as:

$$Q = Q_0 + \beta C, \; C \geq 0, \tag{26}$$

where Q is a cost function, Q_0 is the objective function of the initial problem, β are some coefficients that must be picked up, C are non-negative definite quadratic constraints. The solution then takes the following form:

$$Q[q] = \min_\kappa Q[\kappa]. \tag{27}$$

In the case where $C[q] > 0$, it is possible to choose such β, that if q is unconstrained optimum, then it must be feasible. Beta is chosen such that if q is not feasible, then:

$$Q[q] > Q[feas] = Q[0] = 0, \tag{28}$$

where $Q[feas]$ means the initial zero solution.

The initial objective function that was used for the classical solution will take the following form for the quantum solution:

$$Q_0 = -\sum_{f\in\mathbf{F}} (\sum_{q\in\mathbf{Q}} J_f x_{fq}^2 - \sum_{\alpha\in\mathfrak{A}_f^{start}} \gamma_{f\alpha} y_{f\alpha}^2) \rightarrow \min. \tag{29}$$

Also, the odds β are chosen so that there is a better solution when the constraints are violated:

$$C[q] > 0 \Rightarrow Q[q] \geq Q_0[q] + \beta > 0. \tag{30}$$

Accordingly, in this case it is necessary to take $\beta > -Q_0[q]$. The boundaries of q must be estimated, as the precise values are undetermined. The worst estimate was chosen as the following:

$$\beta > -Q[q] = -(\min_\kappa Q_0[\kappa]) = -(-\sum_f J_f) = \sum_f J_f. \tag{31}$$

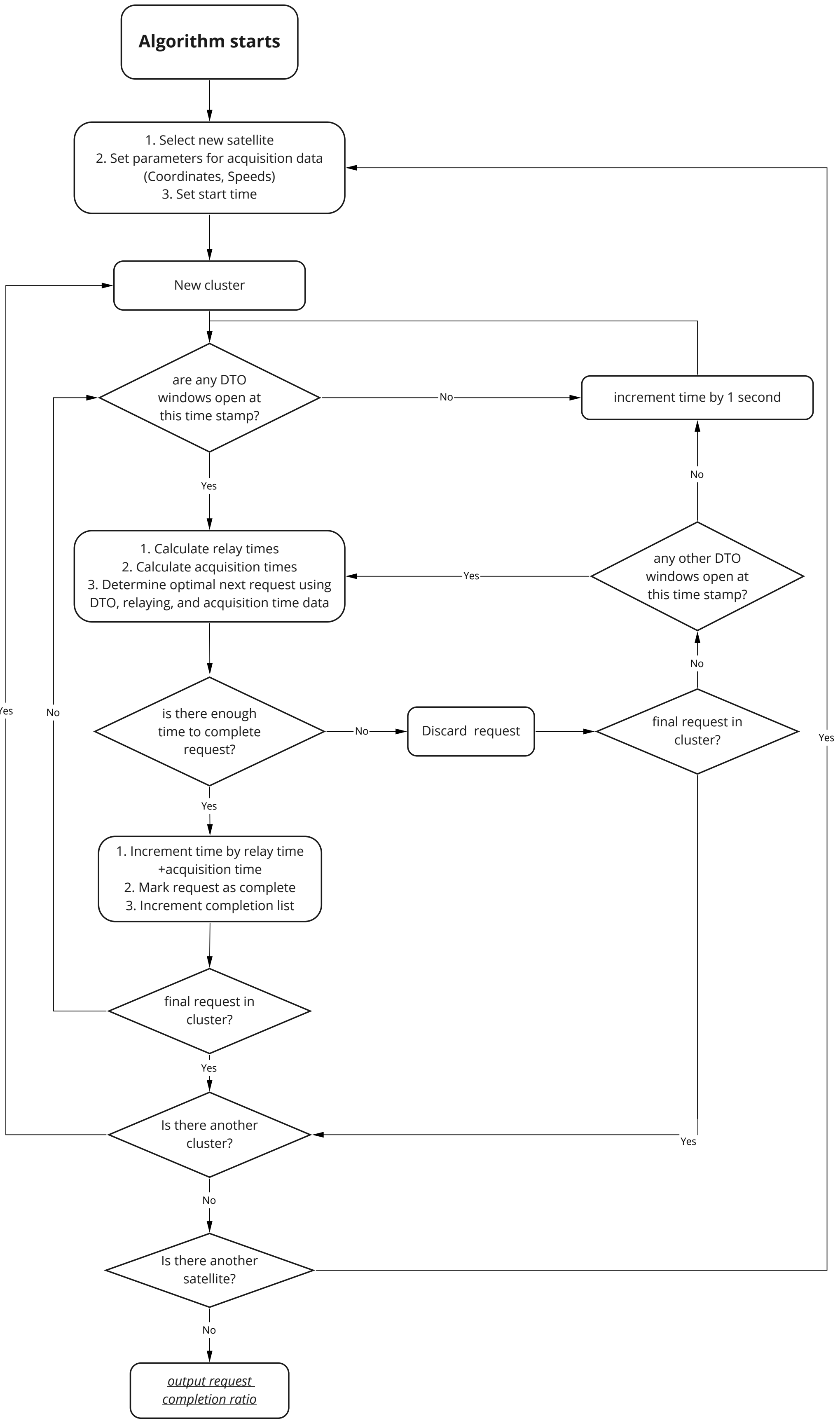

Algorithm starts
1. Select new satellite
2. Set parameters for acquisition data (Coordinates, Speeds)
3. Set start time
New cluster
are any DTO windows open at this time stamp?
No
increment time by 1 second
Yes
No
1. Calculate relay times
2. Calculate acquisition times
3. Determine optimal next request using DTO, relaying, and acquisition time data
Yes
any other DTO windows open at this time stamp?
No
is there enough time to complete request?
No
Discard request
final request in cluster?
Yes
No
Yes
Yes
1. Increment time by relay time +acquisition time
2. Mark request as complete
3. Increment completion list
final request in cluster?
Yes
Is there another cluster?
Yes
No
Is there another satellite?
No
output request completion ratio

Fig. 10. The flowchart used to implement the greedy algorithm in Sec. III-B.