A queue-based approach for fine-tuning the efficiency-fairness tradeoff in distributed satellite scheduling

Shai Krigman
Ariel University
Ariel, Israel
shai.krigman@msmail.ariel.ac.il

Tal Grinshpoun
Ariel University
Ariel, Israel
talgr@ariel.ac.il

Lihi Dery
Ariel University
Ariel, Israel
lihid@ariel.ac.il

ABSTRACT

As the demand for Earth observation satellite (EOS) services grows, allocating limited orbital windows among diverse stakeholders requires balancing overall system utility with equitable access. While distributed constraint optimization (DCOP) has been successfully applied to address user privacy in these settings, existing methods often prioritize efficiency over fairness. A recent hybrid approach offers a compromise between efficiency and fairness. However, it lacks the granularity needed for specific operational contexts. This paper proposes a queue-based distributed algorithm that utilizes multi-level priority queues to resolve resource conflicts. By adjusting the number of queues, the mechanism provides a tunable tradeoff between efficiency and fairness. We provide a formal description of the algorithm and demonstrate its effectiveness in highly-conflicted satellite scheduling scenarios.

KEYWORDS

Earth observation satellites, Distributed Scheduling, Multi-agent systems, Efficiency-fairness tradeoff, DCOP

ACM Reference Format:
Shai Krigman, Tal Grinshpoun, and Lihi Dery. 2026. A queue-based approach for fine-tuning the efficiency-fairness tradeoff in distributed satellite scheduling. In *Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., Paphos, Cyprus, May 2026*, IFAAMAS, 4 pages.

1 INTRODUCTION

Systems of observation satellites (EOS) [12] are frequently co-financed by multiple stakeholders (e.g., countries, companies, or research institutes) due to their high cost, requiring allocation of shared satellite resources once operational. In multi-satellite settings, scheduling must accommodate requests from multiple users, balancing *efficiency* — maximizing utilization for high-priority tasks — and *fairness*, as perceived by stakeholders [1]. Most existing approaches are centralized, assuming all requests are submitted to a single optimizing authority; however, stakeholders may be reluctant to share sensitive or proprietary information.

Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., S. Chien, G. Picard, I. Zilberstein (Chairs), May 2026, Paphos, Cyprus. © 2026 Copyright held by the owner/author(s).

To mitigate these concerns, the distributed constraint optimization problem (DCOP) modeling [6] has been proposed for EOS scheduling [11], followed by several DCOP-based methods [5, 9, 10, 14]. These approaches enable users to coordinate via message exchange and jointly construct a schedule without a central authority.

While DCOP-based algorithms typically emphasize global utility maximization, fairness — namely, equitable resource allocation among stakeholders — remains essential in shared EOS systems.

Prior work introduced a hybrid approach that uses a probabilistic rule to mitigate the "efficiency-first" bias [8]. However, that method provides a static compromise. In this paper, we focus on a queue-based approach that replaces randomization with a structured selection process across multiple priority queues, allowing system operators to explicitly control the efficiency-fairness tradeoff by adjusting the number of queues.

2 PROBLEM DEFINITION

The EOS scheduling problem consists of a set $\mathcal{S}$ of satellites and a set $\mathcal{U}$ of independent users. Each satellite $s \in \mathcal{S}$ follows an orbit plan OP_s and has a capacity limit κ_s, defined as the maximum number of observations during OP_s. Each user $u \in \mathcal{U}$ submits a set $\mathcal{R}$ of observation requests. Each request $r \in \mathcal{R}$ specifies a geographic area to be observed for a duration $\Delta_r \in \mathcal{T}$ within the validity window $[t_r^{start}, t_r^{end}]$, where $t_r^{start}, t_r^{end} \in \mathcal{T}$ and $\mathcal{T}$ denotes the system timeline. The location to be observed is given by p_r (latitude–longitude–altitude).

Given the high cost of Earth observation satellites, they are typically funded by multiple stakeholders with unequal contributions. To reflect these disparities, users are allocated "tokens" representing their entitlement to system usage. Each request r is associated with a reward ρ_r (in tokens), reflecting either the stakeholder's relative contribution or the user's willingness to pay; thus, identical requests may carry different ρ_r values across users.

Each request may induce multiple observation opportunities O. Each opportunity $o \in O$ is characterized by its satellite s_o, start time t_o^{start}, duration Δ_o, and observation location, and yields a reward ρ_o derived from ρ_r and adjusted according to factors such as observation angle and weather conditions. Users generate O based on OP_s, t_r^{start}, t_r^{end}, Δ_r, and p_r. A feasible solution assigns at most one $o \in O$ to each $r \in \mathcal{R}$ such that no two assigned opportunities on the same satellite overlap and the capacity constraint κ_s is satisfied for every $s \in \mathcal{S}$. A desirable solution is both efficient and fair.

3 DCOP MODELING OF THE PROBLEM

A *DCOP* is a tuple $< \mathcal{A}, \mathcal{X}, \alpha, \mathcal{D}, C >$, where $\mathcal{A} = \{A_1, \ldots, A_n\}$ is a finite set of agents; $\mathcal{X} = \{X_1, \ldots, X_m\}$ is a finite set of variables; α :

$\mathcal{X} \rightarrow \mathcal{A}$ assigns each variable to a single agent; $\mathcal{D} = \{D_1, \ldots, D_m\}$ is a set of finite domains, where D_i contains the values of X_i; and C is a set of constraints, each $c \in C$ being a function $c : D_{i_1} \times \cdots \times D_{i_k} \rightarrow \mathbb{R}_+$ that assigns a non-negative cost to every value combination of its scope. A *complete assignment* assigns values to all variables in $\mathcal{X}$, and an *optimal solution* is a complete assignment of minimal total cost.

Krigman et al. [9] model the EOS scheduling problem as a DCOP in which agents represent users ($\mathcal{A} \coloneqq \{u \in \mathcal{U}\}$), variables represent requests ($\mathcal{X} \coloneqq \{r \in \mathcal{R}\}$), and α assigns each r to its requesting user according to $\mathcal{R}_u$. The domain of X_r is $D_r \coloneqq O_r \cup \{\perp\}$, where O_r is the set of feasible observation opportunities for r and $\perp$ denotes rejection.

Three constraint types are defined. **Unary** constraints assign a cost to each value based on the opportunity reward,[1] **binary** constraints impose ∞ cost on overlapping opportunities assigned to the same satellite and zero otherwise; and **global** constraints over opportunities of each satellite s enforce that the number of assigned opportunities does not exceed κ_s.

4 THE DSARC_xQ ALGORITHM

The *DSA_RC* algorithm [3, 7] is an extension of the classical Distributed Stochastic Algorithm (*DSA*) [13] that can explicitly handle resource capacity constraints. In the EOS scheduling context, each agent in *DSA_RC* computes, for each of its requests, the total number of observation opportunities requested by all agents from the satellite associated with its selected opportunity. If this number exceeds the satellite's capacity, the agent independently decides, for each request, whether to withdraw the corresponding opportunity.

Three approaches were proposed in the literature for making the above decision — an efficiency-aimed approach *DSARC_Eff* [9], a fairness-aimed approach *DSARC_Fair* [3], and a hybrid approach *DSARC_Hyb* [8]. In the next subsection, we propose a novel queue-based approach that enables fine-tuning of the efficiency-fairness tradeoff.

Algorithm 1 presents the *DSA_RC* algorithm in a general manner that encapsulates all the existing approaches together with the new queue-based approach. Each agent distributively executes the algorithm for each request. We focus on the call to the function DecideToRemoveOpp (Line 10), which differntiates between the above approaches. The new queue-based approach is described next.

4.1 Queue-based decision

Our proposed approach is inspired by the biased front-queue selector for web crawling [2].

In this approach, presented in Function 2, the agent partitions all observations associated with the same satellite into x reward-based queues, where x is an input parameter (Line 1). The reward range is divided into x intervals, each mapped to a queue, and each observation is assigned accordingly. The first queue contains the highest-reward observations and the last contains the lowest;

[1]Since DCOPs are formulated as minimization problems with non-negative costs, the cost of opportunity o is defined as $cost_o = maxCost - \rho_o$, where $maxCost > \max_{o \in O} \rho_o$. The cost of $\perp$ is $maxCost$.

Algorithm 1: *DSA_RC* run by agent u for request r

Input: p, K

1: $counter \leftarrow 0$
2: $o_r \leftarrow$ Random(O_r)
3: send s_{o_r} and ρ_{o_r} to all agents
4: **while** $counter < K$ **do**
5: $\quad Rew \leftarrow \{\rho_{o_r}\}$
6: $\quad$ collect all agents' s_o and ρ_o
7: $\quad$ **if** $s_{o_r} = s_o$ **then**
8: $\quad\quad Rew \leftarrow Rew \cup \{\rho_o\}$
9: $\quad$ **if** $|Rew| > \kappa_{s_{o_r}}$ **then**
10: $\quad\quad$ **if** DecideToRemoveOpp($Rew, \kappa_{s_{o_r}}$) **then**
11: $\quad\quad\quad o_r \leftarrow \perp$
12: $\quad\quad\quad$ send o_r to all agents
13: $\quad$ **else if** Random([0..1)) $< p$ **then**
14: $\quad\quad o_r \leftarrow$ valid value with minimal cost in D_r
15: $\quad\quad$ send s_{o_r} and ρ_{o_r} to all agents
16: $\quad counter \leftarrow counter + 1$
17: $X_r \leftarrow o_r$

the partitioning may be skewed so that, for example, most high-reward observations reside in the first queue. Each queue is sorted in descending reward order (Lines 2–3). A list arr is then constructed (Lines 5–14) as follows: in iteration i, one observation is selected sequentially from each of the first i queues (when non-empty). Thus, the first iteration draws from the first queue only; the second from the first and second queues; the third from the first, second, and third queues; and so on. The process continues until all queues are empty or arr reaches the capacity limit κ_s. Finally, if the agent's opportunity is not included in arr, it is dropped (Lines 15–18).

The construction of arr enables the inclusion of lower-priority opportunities, promoting fairness while maintaining efficiency. Higher-reward opportunities retain a greater likelihood of selection. The parameter x governs the efficiency-fairness tradeoff: $x = 1$ yields the purely efficiency-oriented variant, whereas larger x increases fairness at the expense of efficiency. This algorithm is denoted *DSARC_xQ*, where x is the number of queues.

5 EXPERIMENTAL EVALUATION

We evaluated four decision types: efficiency-oriented (*DSARC_Eff*), fairness-oriented (*DSARC_Fair*), hybrid (*DSARC_Hyb*), and queue-based (*DSARC_xQ*). For *DSARC_xQ*, we tested *DSARC_2Q*, *DSARC_4Q*, and *DSARC_8Q*. Results were compared to centralized post-processing (*DSA_PP*) [9]. Parameters were set to $p = 0.7$ and $K = 10$ in all algorithms, following [9].

Problem instances were based on the "highly-conflicted problems" of [10]: a 10-minute horizon, three satellites (capacity 20 each), ten users submitting $|\mathcal{R}_u| = \{2, 4, \ldots, 20\}$ requests, and ten opportunities per request. Request validity windows were $[100 : 200]$, with $\Delta_o = [10 : 20]$ and $\rho_o = [10 : 50]$. We also generated a second set of high-density problems.

To capture diverse settings, we considered four reward distributions per instance: uniform, exponential, normal, and a bimodal split (two user groups with high vs. low rewards).

Function 2: Queue-based DecideToRemoveOpp

Input: Rew, κ_s, x

```
Divide Rew into x queues according to ρ_o
for all queues do
    sort queue from highest ρ_o to lowest
cyc ← 0, arr ← ∅
while queues are not empty do
    for i ← 0 to cyc − 1 do
        if queue[i] is not empty then
            move next o from queue[i] to arr
            if |arr| = κ_s then
                break while
    if cyc = x then
        cyc ← 0
    else
        cyc ← cyc + 1
if o_r ∈ arr then
    return false
else
    return true
```

Efficiency was measured by the total number of scheduled requests and total rewards. Fairness was evaluated via the Gini coefficient [4], computed over total rewards and over scheduled requests (with $\rho_o = 1$).

Results for the highly-conflicted set appear in Figure 1. For small workloads, all methods schedule all requests; beyond capacity, scheduling decreases accordingly. Results are therefore reported from 48 requests onward.

Efficiency and fairness exhibit a clear tradeoff: gains in one reduce the other. *DSARC_Hyb* provides a balanced but fixed compromise. When no prior knowledge is available, *DSARC_Hyb* is a robust choice. With distributional insight, *DSARC_xQ* can perform better. Under uniform rewards, *DSARC_2Q* exceeds *DSARC_Hyb* in total rewards while matching its fairness (Gini). Under exponential rewards, *DSARC_2Q* matches *DSARC_Eff* in efficiency and approaches *DSARC_Hyb* in fairness. For bimodal rewards, *DSARC_2Q* and *DSARC_4Q* achieve the highest fairness in scheduled requests.

6 CONCLUSION

We have presented the *DSARC_xQ* algorithm for managing the efficiency-fairness tradeoff in distributed satellite scheduling. By implementing a structured priority queue system, this approach moves beyond the static compromise of the previous hybrid coordination model, offering a tunable solution that can be tailored to specific reward distributions. Our evaluation demonstrates that while the foundation of distributed scheduling relies on efficient DCOP modeling, the addition of multi-level queues effectively prevents the systemic exclusion of users with lower-priority tasks without requiring centralized coordination.

Possible future research directions include investigating the application of this mechanism in dynamic environments and adopting advanced privacy-preserving protocols to further secure stakeholder information.

ACKNOWLEDGEMENTS

This research was supported by the Ministry of Innovation, Science & Technology, Israel.

REFERENCES

[1] Nicolas Bataille, Michel Lemaitre, and Gerard Verfaillie. 1999. Efficiency and Fairness when Sharing the Use of a Satellite. In *Artificial Intelligence, Robotics and Automation in Space*, Vol. 440. 465.

[2] Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. 1998. Efficient crawling through URL ordering. *Computer networks and ISDN systems* 30, 1-7 (1998), 161–172.

[3] Lihi Dery, Tal Grinshpoun, and Ilya Khakhiashvili. 2025. Distributed Course Allocation with Asymmetric Friendships. *Autonomous Agents and Multi-Agent Systems* 39, 1 (2025), 26.

[4] Corrado Gini. 1936. On the measure of concentration with special reference to income and statistics. *Colorado College Publication, General Series* 208, 1 (1936), 73–79.

[5] Ryan Harrod, Shreya Parjan, and Steve Chien. 2022. Distributed observation allocation for a large-scale constellation. (2022).

[6] Katsutoshi Hirayama and Makoto Yokoo. 1997. Distributed partial constraint satisfaction problem. In *International conference on principles and practice of constraint programming*. Springer, 222–236.

[7] Ilya Khakhiashvili, Tal Grinshpoun, and Lihi Dery. 2021. Course Allocation with Friendships as an Asymmetric Distributed Constraint Optimization Problem. In *2021 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. IEEE, 688–693.

[8] Shai Krigman, Lihi Dery, and Grinshpoun. 2025. A Fair Share: Fair Allocation of Satellite Observation Windows According to User Preferences in a Distributed Setting. In *Adjunct Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization (UMAP Adjunct '25)*. 245–248.

[9] Shai Krigman, Tal Grinshpoun, and Lihi Dery. 2024. Scheduling of Earth observing satellites using distributed constraint optimization. *Journal of Scheduling* 27, 5 (2024), 507–524.

[10] Gauthier Picard. 2021. Auction-based and Distributed Optimization Approaches for Scheduling Observations in Satellite Constellations with Exclusive Orbit Portions. *arXiv preprint arXiv:2106.03548* (2021).

[11] Gauthier Picard, Clément Caron, Jean-Loup Farges, Jonathan Guerra, Cédric Pralet, and Stéphanie Roussel. 2021. Autonomous agents and multiagent systems challenges in Earth observation satellite constellations. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021)*. 39–44.

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

[13] Weixiong Zhang, Guandong Wang, Zhao Xing, and Lars Wittenburg. 2005. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence* 161, 1-2 (2005), 55–87.

[14] Itai Zilberstein, Ananya Rao, Matthew Salis, and Steve Chien. 2025. Decentralized, decomposition-based observation scheduling for a large-scale satellite constellation. *Journal of Artificial Intelligence Research* 82 (2025), 169–208.

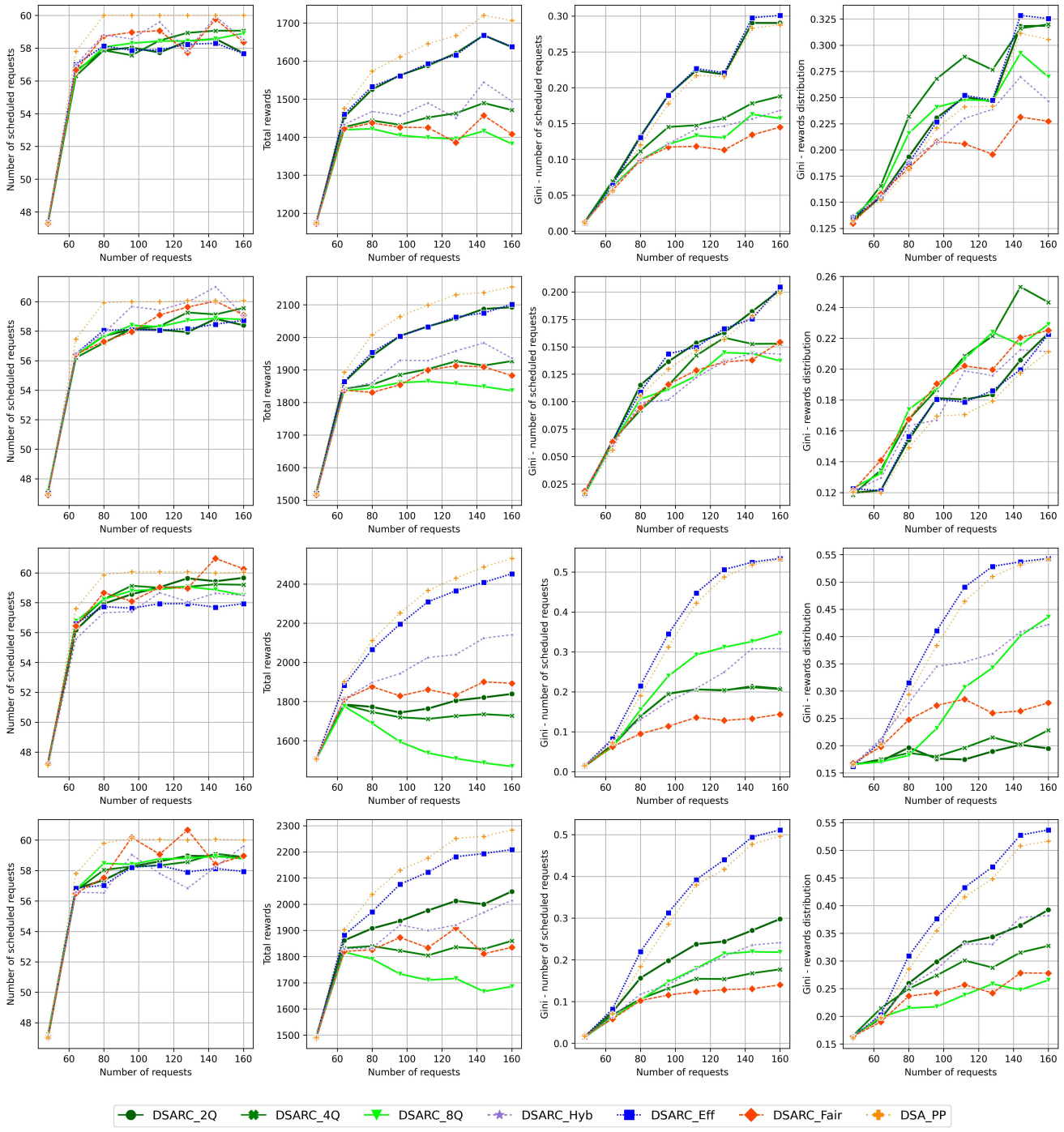

Figure 1: Comparison of scheduling algorithms. Each row corresponds to a different reward distribution: Uniform (first row), Exponential (second row), Normal (third row), and Two equal groups (fourth row). The columns represent the following evaluation metrics: (1) Number of Scheduled Requests, (2) Total Rewards, (3) Gini Coefficient for Scheduled Requests, and (4) Gini Coefficient for Total Rewards.