# DISTRIBUTED CONSTRAINT OPTIMIZATION VIA ONLINE LEARNING AND ITERATIVE PRICING WITH APPLICATION TO LARGE-SCALE SATELLITE SCHEDULING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

*Distributed constraint optimization problems (DCOPs)* provide a popular framework for distributed decision making under limited communication, but many real-world instances are too large to solve monolithically. We address this challenge from two complementary directions. We revisit the connection between DCOPs and potential games, and adapt modern online learning algorithms for equilibrium finding to DCOPs. We show that these algorithms are competitive with representative incomplete DCOP algorithms. We then turn to decomposition frameworks for large-scale DCOPs, motivated by large-scale decentralized satellite scheduling. We propose a new framework that separates a DCOP into two interacting subproblems: a high-level meta-DCOP for task allocation, and independent local optimization problems for scheduling. To couple the two levels, we develop a novel iterative pricing method that updates the meta-level utilities using feedback from the local optimizers. Combining our online learning methods with our iterative pricing framework, we obtain near-optimal performance on real-world decentralized satellite scheduling problem instances, fulfilling over $99\%$ of observation requests compared with $87\%$ for state-of-the-art baselines.

## 1 INTRODUCTION

*Distributed constraint optimization problems (DCOPs)* model collaborative multiagent systems in which agents coordinate local variable assignments to optimize a shared global objective (Fioretto et al., 2018). DCOPs provide a popular framework for distributed decision making under limited communication, with applications ranging from mobile sensor teams (Pertzovsky et al., 2024) to satellite scheduling (Zilberstein et al., 2025b). However, solving DCOPs optimally is NP-hard (Modi et al., 2005). Complete algorithms typically incur exponential time or communication complexity (Modi et al., 2005; Hirayama & Yokoo, 1997; Gershman et al., 2009). Practical DCOP deployments often rely on incomplete local-search, sampling, or inference-based methods, including DSA (Zhang et al., 2005), MGM (Maheswaran et al., 2004), GDBA (Okamoto et al., 2016), D-Gibbs (Nguyen et al., 2019), and Maxsum variants (Zivan et al., 2017). While these methods are lightweight, many real-world domains remain too large or structurally complex to model and solve directly as monolithic DCOPs.

An important application is the coordination of large constellations of Earth-observing satellites. Satellite operations have historically followed centralized paradigms, in which a ground controller issues command sequences for each spacecraft well in advance of execution. Centralized planning is less suitable for emerging time-sensitive use-cases such as disaster response, transient science measurements, and opportunistic observations (Chien et al., 2024; Zilberstein et al., 2025a; Chien et al., 2025b). These use cases, together with rapid improvements in onboard computing, autonomy, and inter-satellite communication (Rijlaarsdam et al., 2024; Zilberstein et al., 2024), have motivated recent work on decentralized satellite scheduling (Phillips & Parra, 2021; Picard, 2022; Parjan & Chien, 2023; Zilberstein et al., 2025b; Zilberstein & Chien, 2026).

DCOPs are a successful model for distributed task allocation among spacecraft (Zilberstein et al., 2025b). However, the complete satellite scheduling problem includes complex local constraints such



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

as orbital visibility, sensor slewing, geometric tiling, onboard memory management, and downlink opportunities (Farges et al., 2024). Encoding such structure in a DCOP requires precomputed discretizations and can yield extremely large, dense constraint graphs, especially for modern constellations with hundreds of satellites servicing tens of thousands of observation requests over week-long planning horizons.

Prior work addressed the scalability challenge through the *neighborhood stochastic search algorithm (NSS)*, a domain-specific geometric decomposition method for the *multi-satellite constellation observation scheduling problem (COSP)* (Zilberstein et al., 2025b). NSS heuristically decomposes the global scheduling problem into smaller subproblems. The decomposition is tightly coupled to the structure of COSP and does not provide a general mechanism for coupling high-level allocation decisions with complex local constraints. This challenge extends well beyond satellite scheduling. Many large-scale DCOP applications require agents to coordinate a high-level allocation while independently solving rich local planning or scheduling problems.

**Our contributions** We introduce a new decomposition method for large DCOPs, which we refer to as *iterative pricing*. Our approach uses a high-level meta-DCOP as a discrete allocator. Given an assignment produced by the DCOP, each agent invokes an independent local solver to determine which assigned tasks can actually be scheduled. The outcome of the local solver is then fed back into the DCOP through assignment-dependent prices, which discourage future allocations that local agents cannot realize. Iterative pricing couples global coordination with local reasoning without requiring the local problem to be encoded directly in the DCOP.

Our method is inspired by distributed Lagrangian and pricing methods (Hirayama et al., 2009), but differs in its decomposition and update structure. Iterative pricing retains a high-level assignment DCOP, restricts each local solver to the assignments of that DCOP, and updates assignment-specific penalties based on the solution of the local optimization problem. The local solver can be an exact *mixed integer linear program (MILP)*, a constraint-programming model, a domain-specific heuristic, or a simulator for complex physical constraints. In the satellite scheduling domain, this allows the global DCOP to reason over high-level request assignment while local solvers handle constraints such as visibility, slewing, memory, downlinking, and geometric tiling. More generally, iterative pricing provides a generic interface between distributed task allocation and arbitrary local optimization, making it applicable to any DCOP domain in which these two levels are naturally separable. The idea of two levels of search, a global one for task allocation and local ones for optimizing the handling of tasks, has also been used successfully in other distributed optimization problems beyond DCOPs (Sandholm, 1993).

Because iterative pricing repeatedly solves an assignment DCOP, scalable DCOP algorithms become critical. We revisit the well-known connection between DCOPs and exact potential games (Maheswaran et al., 2004; Chapman et al., 2009; 2011), and argue that its implications for modern online learning algorithms as DCOP solvers have received insufficient attention. We adapt regret matching and several recent accelerations of it such as discounting (Tammelin, 2014; Brown & Sandholm, 2019), optimism (Farina et al., 2021; Syrgkanis et al., 2015), and extra-gradient updates for potential games (Anagnostides et al., 2026). We also analyze how these methods interact with DCOP stabilization techniques such as probability damping and inertia. While regret matching has recently been applied to DCOPs (Deng et al., 2021), that approach relies on pseudo-tree updates that require global coordination. Our methods instead require only local utility computations and neighbor communication, making them well suited to large-scale decentralized settings. We discuss further related work on DCOPs, online learning, and satellite scheduling in Section A.

We evaluate our methods on standard graph coloring benchmarks and on large-scale decentralized satellite scheduling problem instances. On graph coloring, our online learning methods are competitive with representative incomplete DCOP solvers while maintaining low communication overhead. For the satellite scheduling use case, our iterative pricing substantially improves request completion. Combining our online learning methods with our iterative pricing framework, we obtain near-optimal performance on real-world decentralized satellite scheduling problem instances, fulfilling over $99\%$ of observation requests compared with $87\%$ for state-of-the-art baselines. Our results establish online learning as a competitive paradigm for incomplete DCOP solving, particularly when paired with iterative pricing. Our methods also directly apply to the upcoming NASA FAME mission (Chien et al., 2025a), which will be the largest demonstration of multiagent AI in space to date.

Our experiments are modeled after the operational scenarios that will be tested in flight in 2027, and we seek to deploy our algorithms as part of the FAME mission.

## 2 PRELIMINARIES

A DCOP is formally defined as a five-tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{F}, \omega \rangle$ where $\mathcal{A}$ is the set of agents, $\mathcal{X}$ is the set of variables, and $\mathcal{D}$ is the set of finite domains for the variables, in which each variable $x_i \in \mathcal{X}$ has the corresponding domain $\mathcal{D}_i$. Each constraint $f \in \mathcal{F}$ has a scope, $\text{scope}(f) \subseteq \mathcal{X}$, and maps assignments of the variables in its scope to a real-valued utility, $f\colon \prod_{x_i \in \text{scope}(f)} \mathcal{D}_i \to \mathbb{R}$. The function $\omega\colon \mathcal{X} \to \mathcal{A}$ assigns each variable to the agent that controls it.

The objective is to find a joint variable assignment maximizing the sum of the utilities

$$X^* = \arg\max_{X \in \prod_i \mathcal{D}_i} \sum_{f \in \mathcal{F}} f\left(X|_{\text{scope}(f)}\right),$$

where $X$ ranges over the possible joint assignment of all variables, and $X|_{\text{scope}(f)}$ denotes the value of $X$ on the variables in $\text{scope}(f)$.

In the most basic DCOP setting, the mapping function $\omega$ is bijective; each agent controls exactly one variable. This is the case in some DCOP problems, such as graph coloring where each agent controls one node, and most DCOP algorithms are analyzed under this simplification. However, in many applications, including distributed scheduling, an agent controls many variables and $\omega$ is many-to-one.

A DCOP can be mapped to a *potential game* where each agent $a \in \mathcal{A}$ noncooperatively chooses an assignment $x_a \in \mathcal{D}_a = \prod_{i \in \omega^{-1}(a)} \mathcal{D}_i$ to maximize its local utility $u_a(x_a, X_{-a})$, where $X_{-a}$ denotes the assignments of all other agents (Maheswaran et al., 2004). The connection between DCOPs and exact potential games provides the foundation for the online learning methods developed in this paper. We define the global potential function $\Phi(X)$ as the sum of all utility functions,

$$\Phi(X) = \sum_{f \in \mathcal{F}} f(X).$$

A game is an *exact potential game* if for any deviation of an agent from $x_a$ to $x'_a$, the change in an agent $a$'s local utility reflects the change in the global potential:

$$u_a(x'_a, X_{-a}) - u_a(x_a, X_{-a}) = \Phi(x'_a, X_{-a}) - \Phi(x_a, X_{-a}).$$

In DCOP settings, this property is satisfied when $u_a$ is defined as the sum of utility functions $f \in \mathcal{F}$ that rely on the variable $x_a$. Every pure Nash equilibrium is a coordinate-wise local optimum of the DCOP, and every global optimum is a pure Nash equilibrium. A pure Nash equilibrium need not be globally optimal.

## 3 ONLINE LEARNING ALGORITHMS FOR DCOPS

The connection between equilibria of a potential game and local optima of a DCOP motivates using equilibrium finding to solve DCOPs. Online learning is the predominant paradigm used for equilibrium finding in games.

**Regret in DCOPs** Since our algorithms are based on online learning, we briefly review the notion of *regret* before introducing the algorithms. In an iterative DCOP setting, let $x_a^{(t)} \in \mathcal{D}_a$ be the value assigned to variable $x_a$ by agent $a$ at time step $t$. The local utility $u_a\left(x_a^{(t)}, X_{-a}^{(t)}\right)$ is the sum of all constraints $f \in \mathcal{F}$ involving $x_a$ given the other assignments $X_{-a}^{(t)}$ at that time.

The external regret of agent $a$ for not having chosen a fixed assignment $x'_a \in \mathcal{D}_a$ after $T$ iterations is

$$R_a^T(x'_a) = \sum_{t=1}^{T} u_a\left(x'_a, X_{-a}^{(t)}\right) - \sum_{t=1}^{T} u_a\left(x_a^{(t)}, X_{-a}^{(t)}\right).$$

An algorithm satisfies the *no-regret property* if, for every agent $a$, the average regret vanishes as $T$ grows,

$$\lim_{T\to\infty} \frac{1}{T} \max_{x'_a \in \mathcal{D}_a} R_a^T(x'_a) \leqslant 0.$$

Such algorithms are called *no-regret algorithms*.

**Using equilibrium finding for local search in DCOPs** A key property of exact potential games is that any strategy profile $X$ that is a pure Nash equilibrium is also a local optimum of the potential function $\Phi(X)$. This observation motivates using equilibrium finding as a *decentralized search procedure*. When every agent employs a no-regret algorithm, the empirical distribution of joint assignments converges to the set of coarse correlated equilibria. However, the no-regret learning guarantees concern empirical regret and equilibrium distributions, and do not guarantee optimality of the individual assignments produced. Our DCOP algorithms differ from equilibrium finding algorithms in two major ways. Since we desire variable assignments that produce high-utility solutions rather than a distribution over variable assignments, we sample from the distribution to obtain a solution. We also do not require that the final solution be sampled from an equilibrium: our use of equilibrium finding can be viewed as a heuristic for searching the space of solutions. We ultimately use the best solution found *during the path to an equilibrium*. Experimentally, as we will show, the equilibria our algorithms converge to result in high-quality solutions. In the remainder of this section, we present our DCOP algorithms adapted from the online learning algorithm *regret matching (RM)* (Hart & Mas-Colell, 2000).

**Regret matching (RM)** RM is a no-regret online learning algorithm commonly used for equilibrium finding in games. While regret-based local-search algorithms have previously been studied for DCOPs (Chapman et al., 2009; 2011), more recently, the online learning literature has introduced several RM variants with substantially improved practical performance in large games. We investigate whether these advances translate into improved decentralized DCOP algorithms.

RM guarantees $O(1/\sqrt{T})$ average external regret after $T$ iterations for settings with bounded utilities and finite actions. In potential games, recent work establishes additional convergence guarantees toward approximate Nash equilibria under suitable conditions (Anagnostides et al., 2026). We begin with the standard RM algorithm and then describe the variants evaluated in this paper.

For simplicity, we present the algorithms in the standard one-variable-per-agent setting, where agent $a$'s action is a value $x_a \in \mathcal{D}_a$. For multi-variable agents, the same update applies by replacing $x_a$ with a joint local assignment over all variables.

For any alternative assignment $x \in \mathcal{D}_a$, define the instantaneous regret at time $t$ as $\Delta R_a^{(t)}(x) = u_a(x, X_{-a}^{(t)}) - u_a(x_a^{(t)}, X_{-a}^{(t)})$. The cumulative regret after $t$ iterations is $R_a^{(t)}(x) = R_a^{(t-1)}(x) + \Delta R_a^{(t)}(x)$. An agent $a$ employing RM computes a distribution, denoted $\sigma_a^{(t)}$ over $\mathcal{D}_a$, proportional to the positive regrets of each value:

$$\sigma_a^{(t)}(x_a) = \begin{cases} \frac{R_a^{(t)}(x_a)^+}{\sum_{x'_a \in \mathcal{D}_a} R_a^{(t)}(x'_a)^+} & \text{if } \sum R_a^{(t)}(x'_a)^+ > 0 \\ \frac{1}{|\mathcal{D}_a|} & \text{otherwise} \end{cases}$$

where $R_a^{(t)}(x)^+ = \max\left(0, R_a^{(t)}(x)\right)$. Algorithm A1 presents the full RM algorithm for DCOP.

Like representative incomplete DCOP algorithms, RM requires only local communication with neighboring agents and has a message complexity of $O(|N(a)|)$ per iteration where $N(a)$ is the neighborhood of agent $a$. The computational cost is linear in both the neighborhood size and the local action space. We next describe several RM variants that have accelerated equilibrium finding in large games, and we adapt them to the DCOP setting.

**Regret matching+** RM+ (Tammelin, 2014) floors cumulative regrets at zero after each update,

$$R_a^{(t)}(x) \leftarrow \left[R_a^{(t-1)}(x) + \Delta R_a^{(t)}(x)\right]^+.$$

This prevents actions from accumulating large negative regrets, allowing actions that previously performed poorly to re-enter the strategy more quickly if their counterfactual utilities improve.

**Discounted RM** *Discounted RM (DRM)* (Brown & Sandholm, 2019) reduces the influence of older regret updates using discount parameters $\alpha$ and $\beta$ for positive and negative cumulative regrets. Let $d(t, \gamma) = \frac{t^\gamma}{t^\gamma+1}$. DRM updates the cumulative regret by $R_a^{(t-1)}(x) \cdot d(t, \alpha) + \Delta R_a^{(t)}(x)$ when $R_a^{(t-1)}(x) \geqslant 0$ and $R_a^{(t-1)}(x) \cdot d(t, \beta) + \Delta R_a^{(t)}(x)$ when $R_a^{(t-1)}(x) < 0$.

**Predictive RM** *Predictive RM (PRM)* (Farina et al., 2021) uses a prediction of the next iteration's regret to take a larger step at each strategy update. After updating $R_a^{(t)}$, PRM defines

$$\hat{R}_a^{(t)}(x) \leftarrow R_a^{(t)}(x) + m_a^{(t)}(x),$$

with prediction $m_a^{(t)}$ usually set to $\Delta R_a^{(t)}(x)$, and computes $\sigma_a^{(t+1)}$ using the RM update with $\widehat{R}_a^{(t)}$.

**Increasing-regret PRM** We also evaluate *increasing-regret PRM (IR-PRM)* (Anagnostides et al., 2026), inspired by recent extra-gradient regret-matching methods for potential games. These methods modify the predictive update so that the norm of the regret vector does not decrease, which adapts the step size of the regret dynamics while preserving convergence guarantees.

**Context-based RM** In recent work, RM was applied to DCOPs in a *context-based (CB)* manner, which keeps track of regrets conditional on the variable assignments of an agent's neighbors (Deng et al., 2021). The original implementation performs context-based updates using sequential sampling and backtracking over a pseudo-tree. Our implementation instead performs the updates using only local utility computations and communication between neighboring agents, eliminating the pseudo-tree dependency.

The above RM variants can be combined with each other. In particular, RM+ combines well with the other variants, and we include hybrids in our experiments. For example, the RM+ update with PRM results in the PRM+ algorithm (Farina et al., 2021).

**FTRL** We also compare against *follow the regularized leader (FTRL)* with the *multiplicative weights update (MWU)*, another widely used no-regret online learning algorithm that is outside the RM family. More details on this algorithm are provided in Section C.

The theoretical guarantees of no-regret algorithms concern regret and equilibrium notions over the sequence of play rather than the quality of individual assignments. In our implementations, each iteration produces a concrete assignment by sampling from the current strategies, and we evaluate the quality of these assignments empirically. This allows no-regret dynamics to serve as lightweight decentralized search procedures for incomplete DCOP solving.

**DCOP stabilization** We additionally evaluate two stabilization heuristics commonly used in incomplete DCOP algorithms: *damping* and *inertia*. Interestingly, unlike in other DCOP algorithms, we find that these heuristics *decrease* the quality of the solution when combined with online learning algorithms. We provide more details on these techniques, and prove that RM with inertia violates the no-regret property, in Section B. These results may suggest that moving toward an equilibrium is in fact a strong search heuristic for DCOPs.

## 4 Constellation observation scheduling

We conduct experiments on the canonical DCOP problem of *distributed graph coloring* as well as the real-world *multi-satellite constellation observation scheduling problem (COSP)*, a DCOP formulation of decentralized satellite scheduling introduced by Zilberstein et al. (2025b). A COSP instance consists of a scheduling horizon $H = [h_s, h_e]$, a set of satellites $\mathcal{A}$, and a set of observation requests $R$. Each request $r \in R$ specifies a ground target and a time window $h(r) \subseteq H$ during which the target should be observed.

For each satellite $a \in \mathcal{A}$, there is a set of candidate observation tasks $S_a$ determined by their orbital mechanics and sensor slewing capabilities. Each task $s$ is specified by a request $r(s) \in R$, an execution interval $h(s) \subseteq h(r(s))$, and an onboard data volume $m(s) \in \mathbb{R}_{\geqslant 0}$. We use a binary decision variable $x_{a,s} \in \{0, 1\}$, where $x_{a,s} = 1$ indicates that satellite $a$ schedules task $s$. Each

satellite also has a set of mandatory downlink opportunities $\mathcal{L}_a$. A downlink $\ell \in \mathcal{L}_a$ has a contact interval $h(\ell) \subseteq H$ and a maximum data volume $m(\ell)$.

The constraints of the problem dictate the feasibility of scheduling tasks. Each satellite has a set $\mathcal{C}_a$ of constraints capturing slewing maneuvers, onboard memory management, downlink capacity, and the requirement that a satellite cannot execute overlapping tasks. For example, if two candidate tasks overlap in time, then they cannot both be scheduled, corresponding to the constraints $x_{a,s}+x_{a,s'} \leqslant 1$ for all $s, s' \in S_a$ such that $h(s) \cap h(s') \neq \varnothing$. Let $\mathcal{S}_a^\ell \subseteq \mathcal{S}_a$ denote the tasks whose next downlink opportunity is $\ell$, then memory and downlink capacity impose the constraint $\sum_{s \in S_a^\ell} m(s) x_{a,s} \leqslant \min\{M_a, m(\ell)\}$ where $M_a$ is the total memory budget of satellite $a$.

The objective is to maximize the number of completed observation requests. Let $S(r) = \{(a, s) \mid a \in \mathcal{A},\ s \in S_a,\ r(s) = r\}$ be the set of candidate tasks that can satisfy request $r$. The COSP objective is

$$\mathcal{F}(X) = \sum_{r \in R} \left[ 1 - \prod_{(a,s) \in S(r)} (1 - x_{a,s}) \right],$$

which counts each request at most once, even if multiple satellites could observe it. The full scheduling problem can be written as

$$X^* \in \arg\max_X \mathcal{F}(X) \quad \text{s.t.} \quad X \text{ satisfies } \mathcal{C}_a \quad \forall a.$$

This formulation induces both a large number of decision variables and a dense constraint graph. Each satellite can perform many tasks, and local constraints couple tasks within a satellite, while the objective couples satellites that can observe the same target. In large constellations, many satellites have visibility windows for many of the same requests, producing a dense constraint graph. The next section focuses on decomposition methods for COSP to make decentralized scheduling tractable.

## 5 DCOP DECOMPOSITION METHODS

The direct COSP formulation is too large to solve as a monolithic DCOP, even for moderate-sized constellations. We therefore develop decomposition techniques motivated by COSP that apply to DCOPs that have a natural decomposition to two levels of optimization. This approach is similar in spirit to the hierarchical DCOP model MVA (Fioretto et al., 2016) for handling multivariable agents.

We decompose COSP into two interacting problems: a high-level request-assignment DCOP and independent local scheduling problems. The top-level DCOP decides which satellite should be responsible for each request. For request $r \in R$ and satellite $a \in \mathcal{A}$, let $z_{r,a} \in \{0, 1\}$ indicate whether $r$ is assigned to $a$. The assignment DCOP enforces that each request is assigned to at most one satellite[1], $\sum_{a \in \mathcal{A}} z_{r,a} \leqslant 1$ for all $r \in R$. Given an assignment $z$, let $B_a(z) = \{r \in R \mid z_{r,a} = 1\}$ be the set of requests assigned to satellite $a$. Satellite $a$ then invokes a local scheduler to determine which requests in $B_a(z)$ can be scheduled such that the constraints of $\mathcal{C}_a$ are satisfied.

We model the local scheduler for satellite $a$ as an oracle, denoted $\mathcal{O}_a$. Given a bundle of requests $B \subseteq R$ and (possibly) weights $w$, the oracle returns a feasible scheduled subset $Y_a \subseteq B$ together with the corresponding realized schedule, $\widehat{y}_a$. The oracle may be implemented as a MILP, a constraint-programming solver, a domain-specific heuristic, or a simulator that returns a feasible schedule under the local constraints. (In our experiments, we solve the local problem as a MILP.) This is in contrast to the monolithic DCOP formulation, where the scheduling problem is itself represented as part of the DCOP formulation.

We study two mechanisms for coupling the assignment DCOP with the local scheduling oracles. The first is a simple constraint generation baseline in which, over iterations of assignment and scheduling, constraints are learned and added to the assignment DCOP. The second is our iterative pricing method, which updates the assignment utilities using feedback from the local schedulers.

[1]The problem has the same optimal solutions without this constraint, but our experiments showed that including this constraints had no significant effect on performance.

### 5.1 CONSTRAINT GENERATION BASELINE

The assignment DCOP ignores the local scheduling constraints by design for scalability. It may assign a satellite a bundle of requests that cannot be jointly scheduled. A natural baseline is to iteratively add constraints that prevent the assignment DCOP from repeating infeasible bundles.

At iteration $t$, the assignment DCOP returns an assignment $z^{(t)}$. Each satellite receives the bundle $B_a^{(t)} = B_a(z^{(t)})$ and invokes its local scheduler. If the scheduler certifies that $B_a^{(t)}$ is feasible, then all assigned requests can be executed by $a$. Otherwise, the algorithm adds the constraint $\sum_{r \in B_a^{(t)}} z_{r,a} \leqslant |B_a^{(t)}| - 1$, which prevents the same infeasible bundle again.

The constraint is sound when the local scheduler can certify that $B_a^{(t)}$ is in fact infeasible, which can be achieved via a complete solver (*e.g.*, a MILP). If the local scheduler is heuristic, then the constraint generation is also heuristic, and may eliminate a feasible bundle that the scheduler failed to find. The procedure terminates when no new constraints are added or a maximum number of iterations are reached. We outline this procedure in Algorithm A3. For clarity, the decomposition procedure is shown as synchronous and the call to SOLVEDCOP denotes the execution of any distributed DCOP solver. The remaining steps are performed locally by each agent using its assigned bundle and local scheduler. The pseudocode is centralized only in notation.

### 5.2 ITERATIVE PRICING

Constraint generation gives the assignment DCOP only binary feedback about whether a bundle is infeasible. Iterative pricing accumulates penalties on individual assignments, providing progressively richer feedback to the assignment DCOP over iterations. Intuitively, if request $r$ is repeatedly assigned to satellite $a$ but cannot be scheduled, then the *price* of assigning $r$ to $a$ increases, making that assignment less attractive in future iterations.

Let $U_r$ be the baseline utility of completing request $r$ ($U_r = 1$ in COSP). For each $(r, a)$, iterative pricing maintains a nonnegative price $\lambda_{r,a}^{(t)}$, initialized to zero. At iteration $t$, the assignment DCOP solves

$$z^{(t)} \in \arg\max_{z} \sum_{r \in R} \sum_{a \in \mathcal{A}} \left( U_r - \lambda_{r,a}^{(t)} \right) z_{r,a}$$

subject to $\sum_{a \in \mathcal{A}} z_{r,a} \leqslant 1 \; \forall r \in R$. Assignments that have previously led to local scheduling failures become less rewarding to the global allocator. Given $z^{(t)}$, each satellite $a$ calls its local scheduler on the bundle $B_a^{(t)} = \{r \in R \mid z_{r,a}^{(t)} = 1\}$. The local scheduler returns a feasible subset $Y_a^{(t)} \subseteq B_a^{(t)}$, represented by indicators $y_{r,a}^{(t)}$.

The local scheduler makes use of weights, defined by $U_r + \lambda_{r,a}^{(t)}$, and solves

$$y_a^{(t)} \in \arg\max_{y_a} \sum_{r \in R} \left( U_r + \lambda_{r,a}^{(t)} \right) y_{r,a}$$

subject to $y_a^{(t)} \in \mathcal{C}_a$ and $y_{r,a}^{(t)} \leqslant z_{r,a}^{(t)}$ for all $r \in R$. The constraint $y_{r,a}^{(t)} \leqslant z_{r,a}^{(t)}$ restricts the scheduler to requests assigned to $a$. The utility $U_r$ encourages the scheduler to complete requests, while the price prioritizes requests that have previously been difficult to reconcile with the global assignment.

After all schedulers return feasible schedules, we compare the assignment $z^{(t)}$ to the schedule $y^{(t)}$. We define $g_{r,a}^{(t)} = z_{r,a}^{(t)} - y_{r,a}^{(t)}$. Since $y_{r,a}^{(t)} \leqslant z_{r,a}^{(t)}$, $g_{r,a}^{(t)}$ is always nonnegative. The price update is

$$\lambda_{r,a}^{(t+1)} = \lambda_{r,a}^{(t)} + \alpha \cdot g_{r,a}^{(t)},$$

where $\alpha > 0$ is a step size. If request $r$ is assigned to satellite $a$ but not scheduled, the price of that assignment increases. If the request is assigned and successfully scheduled, the price remains unchanged. The loop terminates when $z^{(t)} = y^{(t)}$, meaning every assigned request is successfully scheduled, or when a maximum number of iterations is reached. We present this procedure in Algorithm A2.

Our method is inspired by distributed Lagrangian pricing methods, which update multipliers associated with violated constraints. Unlike these dual-decomposition methods, iterative pricing retains

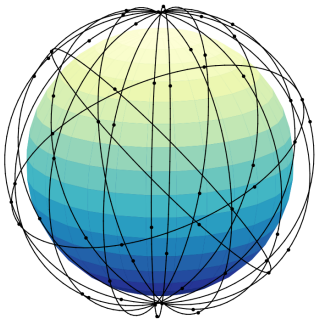


Figure 1: The 60-satellite Walker constellation.

a high-level assignment DCOP, restricts each local optimizer to the requests selected by that DCOP, and uses assignment-specific penalties. The resulting method is a coordination heuristic rather than an optimization algorithm for a Lagrangian dual. To our knowledge, this is the first price-guided interface between a high-level DCOP allocator and arbitrary local scheduling oracles.

## 6 RESULTS

We begin by comparing the RM and FTRL algorithms against a suite of prominent incomplete DCOP solvers on canonical graph coloring benchmarks. We evaluate *DSA-C*, *MGM2*, *GDBA*, and *Maxsum-ADVP*. We find that across network types and sizes, online learning algorithms converge to as good or better solutions faster than existing incomplete DCOP algorithms. These experiments and ablation studies are included for completeness in Sections E and F.

We then demonstrate the performance of our decomposition frameworks on COSP instances. We compare both the constraint generation baseline and our iterative pricing method to NSS, and evaluate each framework with a suite of DCOP algorithms including the RM variants. We provide details of the hyperparameter selection and compute environment in Sections G and H respectively.

**COSP** We simulate a low-Earth orbit Walker constellation composed of 60 satellites. The constellation has 8 orbital planes at an $88^\circ$ inclination each containing 6 satellites. There is an overlay of 2 planes at a $51.6^\circ$ inclination containing 6 satellites each. This configuration is motivated by the SkySat constellation (Planet, 2023), and is designed to match the number of spacecraft in the NASA FAME mission (Chien et al., 2025a). All satellites are homogeneous; they have a memory capacity of 125 GB and a sensor that can slew to $45^\circ$ off-nadir. Figure 1 visualizes the constellation.

We also simulate two downlink stations: the ASF Near Space Network Satellite Tracking Ground Station and the Guam Remote Ground Terminal System. A downlink is modeled by a bit stream of 62.5 MB/s.

For each simulation, we randomly select a horizon of six hours within a one week interval. Six hours corresponds to roughly four full orbits of Earth per satellite. We generate a set of repeat observations of a random subset of 634 of the most populous, globally distributed cities. For each ground target, we add an observation request in the first half, second half, or both halves of the horizon.

We generate 20 problem instances (observation campaigns) for evaluation, and run each algorithm within each framework for 20 trials on each instance. We report the utility of the final solution achieved by each algorithm in Figure 2. We use the context-based variants of the RM algorithms, as this resulted in the best performance when paired with the iterative pricing framework. Context-based variants had the opposite effect under constraint generation, so we display the standard variant of IR-PRM as a representative. We provide more details on the context-based results in Section F.2. We run each algorithm for 25 iterations, which we found to be enough for convergence (Section F.4). The iterative pricing step size, $\alpha$, is also a tunable parameter, and we show these results in Section F.3. With each algorithm, we report the optimal $\alpha$ found in the iterative pricing framework. We provide the final list of hyperparameters used in Table A2.

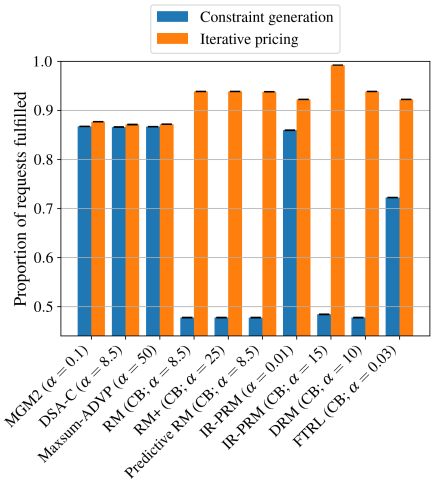



Figure 2: Average quality of satellite scheduling solution found. Standard error of the sample mean is displayed.

| **Framework** | **Proportion of requests fulfilled** | **Message volume** |
|---|---|---|
| NSS | 0.870 | **84,180** |
| Constraint generation (+ MGM2) | 0.867 | 696,087 |
| Iterative pricing (+ CB IR-PRM) | **0.992** | 1,331,750 |

Table 1: Average results of highest satisfaction variant of each framework across 20 problem instances.

The best algorithm within the iterative pricing framework was context-based IR-PRM, which fulfilled 99.2% of requests. The constraint generation framework with MGM2, fulfilled only 86.7% of the observation requests. The iterative pricing framework performed significantly better than constraint generation with all algorithms apart from DSA-C and Maxsum-ADVP. This difference is likely due to the combinatorial infeasibility of restricting enough schedules to force the algorithm to converge to a better solution. Within the iterative pricing framework, all online learning algorithms performed well, fulfilling over 90% of requests. The other incomplete DCOP algorithms failed to reach 90% request completion. The online learning variants performed substantially better within our framework.

Finally, we compare our frameworks to NSS (Table 1). We report the highest satisfaction variant of each algorithm, and use NSS with the GND(2) heuristic. Our iterative pricing approach was near-optimal and satisfied over 99% of requests. NSS and the constraint generation framework only achieved around 87% satisfaction. However, NSS, which focuses on scalability, reduced the message volume substantially. This indicates a natural tradeoff between our method and NSS: for closer-to-optimal solutions, one should select our iterative pricing framework, and for applications in which very low message volume is important, NSS is more appropriate.

## 7 Conclusions and Future Work

We presented two complementary contributions toward scalable distributed constraint optimization. We demonstrated that modern online learning algorithms can be adapted to search and are a competitive family of incomplete DCOP solvers. These methods achieved solution quality comparable to the predominant prior incomplete DCOP algorithms on graph coloring benchmarks. We introduced iterative pricing, a new price-guided decomposition framework for DCOPs. Rather than requiring complex local planning problems to be encoded directly within the DCOP, iterative pricing couples a high-level assignment DCOP with independent local optimizers through prices. On large-scale decentralized satellite scheduling problems, our framework substantially improved request completion over prior decomposition methods.

Although our experiments focused on satellite scheduling, our framework is applicable whenever a distributed allocation problem naturally decomposes into a global assignment problem together with rich local planning or scheduling problems. We anticipate that this abstraction will be useful across distributed planning domains, including mobile sensor teams (Pertzovsky et al., 2024), UAV coordination (Pujol-Gonzalez et al., 2013), and trucking task allocation, routing, and scheduling (Sandholm, 1993).

### AI USE STATEMENT

In this work, we used generative AI tools for developing the software used in the experiments and editing the manuscript for clarity and consistency. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat datasets, support qualitative and thematic data analysis, or interpret results. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

All proofs and pseudocode can be found in Sections B and D. We present all necessary details for reproducibility in Sections G and H including hyperparameter settings, software libraries, and compute details. We also provide the code and data used for the experiments as part of the submission.

## REFERENCES

Ioannis Anagnostides, Emanuel Tewolde, Brian Hu Zhang, Ioannis Panageas, Vincent Conitzer, and Tuomas Sandholm. Convergence of regret matching in potential games and constrained optimization. In *International Conference on Learning Representations (ICLR)*, 2026.

Sean Augenstein, Alejandra Estanislao, Emmanuel Guere, and Sean Blaes. Optimal scheduling of a constellation of Earth-imaging satellites, for maximal data throughput and efficient human management. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2016.

Romain Barrault, Cédric Pralet, Gauthier Picard, and Eric Sawyer. Hybridizing machine learning and optimization for planning satellite observations. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*, 2025.

James Boerkoel, James Mason, Daniel Wang, Steve Chien, and Adrien Maillard. An efficient approach for scheduling imaging tasks across a fleet of satellites. In *International Workshop on Planning and Scheduling for Space (IWPSS)*, 2021.

Grégory Bonnet and Catherine Tessier. Collaboration among a satellite swarm. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2007.

Grégory Bonnet and Catherine Tessier. Coordination despite constrained communications: A satellite constellation case. In *National Conference on Control Architectures of Robots (CAR)*, 2008.

Noam Brown and Tuomas Sandholm. Solving imperfect-information games via discounted regret minimization. In *Conference on Artificial Intelligence (AAAI)*, 2019.

Archie Chapman, Rosa Anna Micillo, Ramachandra Kota, and Nick Jennings. Decentralised dynamic task allocation: A practical game–theoretic approach. *Autonomous Agents and Multi-Agent Systems*, 2009.

Archie C Chapman, Alex Rogers, and Nicholas R Jennings. Benchmarking hybrid algorithms for distributed constraint optimisation games. *Autonomous Agents and Multi-Agent Systems*, 22(3): 385–414, 2011.

Abhijit Chatterjee and Ratnasingham Tharmarasa. Multi-stage optimization framework of satellite scheduling for large areas of interest. *Advances in Space Research*, 73(3), 2024.

Steve Chien, Alberto Candela, Itai Zilberstein, David Rijlaarsdam, Tom Hendrix, and Aubrey Dunne. Leveraging commercial assets, edge computing, and near real-time communications for an enhanced New Observing Strategies (NOS) flight demonstration. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2024.

Steve Chien, Itai Zilberstein, Alberto Candela, Domenico Barretta, David Rijlaarsdam, Tom Hendrix, Aubrey Dunne, Oriol Cortés Grauc, Alexandre Gol i Mestrec, Manel Pedra Bovec, Oriol Aragon, Juan Puig Miquel, Arvind Subramanian, Vishesh Vatsal, Adithya Kothandhapani, Jad Mogannam, and Mitchell Scher. Multi-asset New Observing Systems flight demonstration. In *International Conference on Space Operations*, 2025a.

Steve Chien, Itai Zilberstein, Alberto Candela, Domenico Barretta, David Rijlaarsdam, Tom Hendrix, Aubrey Dunne, Amaury Perrocheau, Cristopher Castro Traba, Oriol Cortes Grau, et al. Dynamic targeting–flight report. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (ISAIRAS)*, 2025b.

Yanchen Deng, Runshen Yu, Xinrun Wang, and Bo An. Neural regret-matching for distributed constraint optimization problems. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

Duncan Eddy and Mykel J Kochenderfer. A maximum independent set method for scheduling Earth-observing satellite constellations. *Journal of Spacecraft and Rockets*, 58(5):1416–1429, 2021.

Jean-Loup Farges, Filipo Studzinski Perotto, Cédric Pralet, Gauthier Picard, Cyril de Lussy, Jonathan Guerra, Philippe Pavero, and Fabrice Planchou. Going beyond mono-mission Earth observation: Using the multi-agent paradigm to federate multiple missions. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2024.

Gabriele Farina, Christian Kroer, and Tuomas Sandholm. Faster game solving via predictive Blackwell approachability: Connecting regret matching and mirror descent. In *Conference on Artificial Intelligence (AAAI)*, 2021.

Ferdinando Fioretto, William Yeoh, and Enrico Pontelli. Multi-variable agents decomposition for DCOPs. In *Conference on Artificial Intelligence (AAAI)*, 2016.

Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. Distributed constraint optimization problems and applications: A survey. *Journal of Artificial Intelligence Research (JAIR)*, 61:623–698, 2018.

Amir Gershman, Amnon Meisels, and Roie Zivan. Asynchronous forward bounding for distributed COPs. *Journal of Artificial Intelligence Research (JAIR)*, 34:61–88, 2009.

Al Globus, James Crawford, Jason Lohn, and Anna Pryor. A comparison of techniques for scheduling Earth-observing satellites. In *Conference on Innovative Applications of Artificial Intelligence (IAAI)*, 2004.

Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

Lei He, Xiaolu Liu, Gilbert Laporte, Yingwu Chen, and Yingguo Chen. An improved adaptive large neighborhood search algorithm for multiple agile satellites scheduling. *Computers & Operations Research*, 100:12–25, 2018.

Katsutoshi Hirayama and Makoto Yokoo. Distributed partial constraint satisfaction problem. In *International Conference on Principles and Practice of Constraint Programming (CP)*, 1997.

Katsutoshi Hirayama, Toshihiro Matsui, and Makoto Yokoo. Adaptive price update in distributed lagrangian relaxation protocol. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2009.

John N Hooker and Greger Ottosson. Logic-based Benders decomposition. *Mathematical Programming*, 96(1):33–60, 2003.

Akseli Kangaslahti, Itai Zilberstein, Alberto Candela, and Steve Chien. Dynamic targeting of satellite observations using supplemental geostationary satellite data and hierarchical planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

Grace Ra Kim, Duncan Eddy, Vedant Srinivas, and Mykel J Kochenderfer. Free-placement optimization of ground station locations for low-Earth orbit satellites. *arXiv preprint arXiv:2606.12667*, 2026.

Rajiv T Maheswaran, Jonathan P Pearce, and Milind Tambe. Distributed algorithms for DCOP: A graphical-game-based approach. In *International Conference on Parallel and Distributed Computing Systems*, 2004.

Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2005.

Sreeja Nag, Alan S Li, and James H Merrick. Scheduling algorithms for rapid imaging using agile cubesat constellations. *Advances in Space Research*, 61(3):891–913, 2018.

Duc Thien Nguyen, William Yeoh, Hoong Chuin Lau, and Roie Zivan. Distributed Gibbs: A linear-space sampling-based DCOP algorithm. *Journal of Artificial Intelligence Research (JAIR)*, 64: 705–748, 2019.

Steven Okamoto, Roie Zivan, and Aviv Nahon. Distributed breakout: Beyond satisfaction. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

Shreya Parjan and Steve Chien. Decentralized observation allocation for a large-scale constellation. *Journal of Aerospace Information Systems (JAIS)*, pp. 1–15, 2023.

Arseniy Pertzovsky, Roie Zivan, and Noa Agmon. Collision avoiding max-sum for mobile sensor teams. *Journal of Artificial Intelligence Research (JAIR)*, 79:1281–1311, 2024.

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

Gauthier Picard. Auction-based and distributed optimization approaches for scheduling observations in satellite constellations with exclusive orbit portions. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2022.

Planet. Our constellations. `https://www.planet.com/our-constellations`, 2023. Accessed: 2026-07-20.

Marc Pujol-Gonzalez, Jesus Cerquides, Pedro Meseguer, Juan Antonio Rodríguez-Aguilar, and Milind Tambe. Engineering the decentralized coordination of UAVs with limited communication range. *Advances in Artificial Intelligence*, 1:199–208, 2013.

David Rijlaarsdam, Tom Hendrix, Pablo T Toledano González, Alberto Velasco-Mata, Léonie Buckley, Juan Puig Miquel, Oriol Aragon Casaled, and Aubrey Dunne. The next era for Earth observation spacecraft: An overview of CogniSAT-6. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2024.

Tuomas Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Conference on Artificial Intelligence (AAAI)*, 1993.

Vishwa Shah, Vivek Vittaldev, Leon Stepan, and Cyrus Foster. Scheduling the world's largest Earth-observing fleet of medium-resolution imaging satellites. In *International Workshop on Planning and Scheduling for Space (IWPSS)*, 2019.

Samuel Squillaci, Stéphanie Roussel, and Cédric Pralet. Managing complex requests for a constellation of Earth-observing satellites. In *International Workshop on Planning and Scheduling for Space (IWPSS)*, 2021.

Samuel Squillaci, Cédric Pralet, and Stéphanie Roussel. Scheduling complex observation requests for a constellation of satellites: Large neighborhood search approaches. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*, 2023.

Vasilis Syrgkanis, Alekh Agarwal, Haipeng Luo, and Robert E Schapire. Fast convergence of regularized learning in games. In *Neural Information Processing Systems (NeurIPS)*, 2015.

Oskari Tammelin. Solving large imperfect information games using CFR+. *arXiv preprint arXiv:1407.5042*, 2014.

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

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

Itai Zilberstein and Steve Chien. Large-scale continual scheduling and execution for dynamic distributed satellite constellation observation allocation. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2026. Extended Abstract.

Itai Zilberstein, Alberto Candela, Steve Chien, David Rijlaarsdam, Tom Hendrix, Leonie Buckley, and Aubrey Dunne. Demonstrating onboard inference for Earth science applications with spectral analysis algorithms and deep learning. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (ISAIRAS)*, 2024.

Itai Zilberstein, Alberto Candela, and Steve Chien. Real-time instrument planning and perception for novel measurements of dynamic phenomena. In *Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*, October 2025a.

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

Roie Zivan, Tomer Parash, Liel Cohen, Hilla Peled, and Steven Okamoto. Balancing exploration and exploitation in incomplete min/max-sum inference for distributed constraint optimization. *Autonomous Agents and Multi-Agent Systems*, 31(5):1165–1207, 2017.

## A Additional related work

**Distributed constraint optimization** Optimal algorithms for DCOP typically operate on the *pseudo-tree* representation of a DCOP, incurring exponential time and message complexity by performing exhaustive search (Modi et al., 2005; Hirayama & Yokoo, 1997; Gershman et al., 2009). Incomplete DCOP algorithms are typically based on local search, sampling, or inference. We compare against representative methods from these families, including the *distributed stochastic search algorithm (DSA)* (Zhang et al., 2005), *generalized distributed breakout algorithm (GDBA)* (Okamoto et al., 2016), *maximum gain messaging (MGM)* (Maheswaran et al., 2004), *D-Gibbs* (Nguyen et al., 2019), and *Maxsum* (Zivan et al., 2017).

To handle complex local problem spaces, prior work introduced the *multi-variable agent (MVA)* method for DCOPs (Fioretto et al., 2016). MVA groups local variables controlled by the same agent and utilizes a centralized solver within the agent before participating in message passing. While MVA significantly improves scalability by allowing agents to solve local optimization problems internally, the local variables and constraints must still be explicitly represented within the DCOP model.

Our approach is motivated by the *distributed Lagrangian relaxation protocol (DisLRP)* (Hirayama et al., 2009), a pricing penalty method originally developed for the generalized mutual assignment problem. DisLRP updates Lagrange multipliers associated with relaxed constraints. Our approach is similarly inspired by pricing methods, but differs in three important respects. Iterative pricing retains a high-level assignment DCOP rather than solving a dual decomposition. Second, local schedulers are restricted to the requests assigned by the DCOP. And finally, prices are updated from scheduling outcomes rather than explicit constraint violations in the global model.

Our decomposition is also conceptually related to decomposition techniques from mathematical optimization, particularly *Logic-based Benders decomposition (LBBD)* (Hooker & Ottosson, 2003). LBBD separates a global master problem from richer local optimization subproblems that may be solved by arbitrary optimization procedures.

The connection between DCOPs and *potential games* was made early on in the literature, and regret-based local-search algorithms have been investigated for DCOPs (Chapman et al., 2009; 2011). More recently, Deng et al. (2021) proposed a context-based regret-matching algorithm that performs updates using sequential sampling and backtracking over a pseudo-tree. Our algorithms instead use only local utility evaluations and neighbor communication.

Recent advances in regret minimization such as optimism (Farina et al., 2021), reweighting (Brown & Sandholm, 2019), and discounting (Tammelin, 2014) have substantially improved practical equilibrium finding in large games. Theoretically, optimistic learning rates are known to accelerate convergence in potential games (Syrgkanis et al., 2015). However, the practical efficacy of these predictive dynamics in large multiagent systems with synchronous updates is less understood. Regret matching has also been recently studied theoretically in the context of potential games, where convergence guarantees were obtained (Anagnostides et al., 2026).

**Satellite scheduling** Satellite observation scheduling is primarily modeled as an optimization problem involving computational geometry, constrained task allocation, and coordination among downlink stations, satellites, and operation centers. The vast majority of prior work has focused on centralized paradigms for scheduling, and this is the standard in deployed approaches (Globus et al., 2004; Augenstein et al., 2016; Nag et al., 2018; He et al., 2018; Shah et al., 2019; Wang et al., 2020; Squillaci et al., 2021; Boerkoel et al., 2021; Eddy & Kochenderfer, 2021; Squillaci et al., 2023; Chatterjee & Tharmarasa, 2024; Barrault et al., 2025; Kim et al., 2026). Decentralized approaches include auction-based methods (Picard, 2022; Phillips & Parra, 2021) and heuristic search-based methods (Parjan & Chien, 2023; Bonnet & Tessier, 2007; 2008; Zilberstein et al., 2025b). We leverage the DCOP formulation of decentralized satellite scheduling from Zilberstein et al. (2025b), referred to as the *multi-satellite constellation observation scheduling problem (COSP)*.

COSP is a challenging application of DCOP methods due to its scale and structure. COSP instances are typically composed of tens to hundreds of satellites and hundreds to thousands of requests, resulting in up to millions of decision variables. Due to the short orbital period of low-Earth orbiting satellites, each agent gets many visibility windows of ground targets. The resulting constraint graph contains many high-degree nodes and dense subgraphs. These factors make DCOP approaches that rely on agents communicating with neighboring agents in the constraint graph computationally challenging due to the high degrees in the graph.

To solve COSP, prior work used decomposition-based methods to decompose the constraint graph by partitioning agents and requests in order to run DSA; this approach is referred to as *neighborhood stochastic search (NSS)* (Zilberstein et al., 2025b). NSS is empirically effective, but its geometric decomposition is tailored to COSP. In contrast, our method retains a global DCOP and uses feedback from local scheduling oracles to update assignment prices. This provides a more general interface between distributed task allocation and local constraint reasoning, and enables local scheduling problems to be modeled outside the DCOP representation.

It is also worth noting that our abstraction of the local scheduling problem is directly compatible with other single-agent technologies for spacecraft planning such as dynamic targeting (Zilberstein et al., 2025a; Chien et al., 2025b; Kangaslahti et al., 2026).

## B INERTIA AND DAMPING

*Damping* smooths changes in an agent's mixed strategy through the update $\sigma_a^{(t+1)} \leftarrow (1-\lambda)\sigma_a^{(t)} + \lambda\sigma_a^{(t+1)}$ for $\lambda \in [0,1]$. *Inertia* reduces thrashing by limiting how often agents change their assignments. For $p_I \in [0,1]$, agent $a$ keeps its previous action with probability $p_I$, $x_a^{(t+1)} \leftarrow x_a^{(t)}$. And, with probability $1-p_I$ an agent samples from the current strategy, $x_a^{(t+1)} \sim \sigma_a^{(t+1)}$.

We show that inertia does not generally preserve no-regret guarantees by giving a two-action counterexample in which the learner incurs linear external regret.

Consider a single agent with two actions, denoted $e_1 = (1, 0)$ and $e_2 = (0, 1)$. At each iteration $t \geqslant 2$, define the utility vector adversarially as a function of the agent's previous realized action:

$$u^{(t)} = \begin{cases} (1, -1) & \text{if } x^{(t-1)} = e_2, \\ (-1, 1) & \text{if } x^{(t-1)} = e_1. \end{cases}$$

Thus, the action opposite to the agent's previous action receives utility 1, while repeating the previous action receives utility $-1$.

Suppose the non-inertial update would select the opposite of the previous action, but inertia forces the agent to keep its previous action with probability $p$. Then, conditioned on the history before round $t$, the expected utility of the inertialized action is

$$\mathbb{E}\left[u^{(t)}(x^{(t)}) \mid \mathcal{H}_{t-1}\right] = (1 - p) \cdot 1 + p \cdot (-1) = 1 - 2p.$$

Therefore, for $T$ rounds,

$$\mathbb{E}\left[\sum_{t=2}^{T} u^{(t)}(x^{(t)})\right] = (T - 1)(1 - 2p).$$

Now consider the best fixed action in hindsight. For every realized utility vector $u^{(t)}$, the two action utilities sum to zero:

$$u^{(t)}(e_1) + u^{(t)}(e_2) = 0.$$

Hence, for every realized history,

$$\max_{i \in \{1,2\}} \sum_{t=2}^{T} u^{(t)}(e_i) \geqslant \frac{1}{2} \sum_{i=1}^{2} \sum_{t=2}^{T} u^{(t)}(e_i) = 0.$$

It follows that the expected external regret is at least

$$\mathbb{E}\left[\max_{i \in \{1,2\}} \sum_{t=2}^{T} u^{(t)}(e_i) - \sum_{t=2}^{T} u^{(t)}(x^{(t)})\right] \geqslant (2p - 1)(T - 1).$$

For any $p > 1/2$, inertia can incur linear external regret.

## C FOLLOW THE REGULARIZED LEADER

While RM focuses on the difference between counterfactual and actual utilities, *follow the regularized leader (FTRL)* is a general framework where agents choose a strategy that would have performed best across all previous iterations, modified by a regularization term to maintain stability. In FTRL, the strategy for agent $a$ at time $t + 1$ is defined as:

$$\sigma_a^{(t+1)} = \arg\max_{\sigma \in \Delta(\mathcal{D}_a)} \left( \sum_{\tau=1}^{t} \sum_{x \in \mathcal{D}_a} \sigma(x) u_a(x, X_{-a}^{(\tau)}) - \mathcal{R}(\sigma) \right)$$

where $\mathcal{R}(\sigma)$ is a strongly convex regularizer.

The *multiplicative weights update (MWU)* algorithm is one of the most common instances of FTRL. It utilizes negative Shannon entropy as the regularizer:

$$\mathcal{R}(\sigma) = \frac{1}{\eta} \sum_{x \in \mathcal{D}_a} \sigma(x) \ln \sigma(x)$$

where $\eta > 0$ is the learning rate. The MWU strategy at iteration $t + 1$ is then computed proportional to the softmax:

$$\sigma_a^{(t+1)}(x) = \frac{\exp\left(\eta R_a^t(x)\right)}{\sum_{x' \in \mathcal{D}_a} \exp\left(\eta R_a^t(x')\right)}.$$

MWU is also a no-regret online learning algorithm with the same $O(|N(a)|)$ message complexity as RM. However, MWU will maintain a strictly positive probability for every variable assignment.

## D ADDITIONAL PSEUDOCODE

**Algorithm A1** Distributed regret matching (RM) for agent $a$

**Initialize:**
Cumulative regrets $R_a(x) \leftarrow 0, \forall x \in \mathcal{D}_a$
Select initial assignment $x_a^{(1)} \in \mathcal{D}_a$ uniformly at random
Send $x_a^{(1)}$ to all neighbors $N(a)$
**for** each iteration $t = 1, \ldots, T$ **do**
  **Wait** for messages $x_j^{(t)}$ from all neighbors $j \in N(a)$
  Form the current neighborhood assignment $X_{-a}^{(t)}$
  Compute actual utility: $u_a(x_a^{(t)}, X_{-a}^{(t)})$
  **for all** $x \in \mathcal{D}_a$ **do**
    Compute counterfactual utility: $u_a(x, X_{-a}^{(t)})$
    Update cumulative regret:
    $R_a(x) \leftarrow R_a(x) + (u_a(x, X_{-a}^{(t)}) - u_a(x_a^{(t)}, X_{-a}^{(t)}))$
  **end for**
  $S_a \leftarrow \sum_{x' \in \mathcal{D}_a} \max(0, R_a(x'))$
  **for all** $x \in \mathcal{D}_a$ **do**
    **if** $S_a > 0$ **then**
      $\sigma_a^{(t+1)}(x) \leftarrow \frac{\max(0, R_a(x))}{S_a}$
    **else**
      $\sigma_a^{(t+1)}(x) \leftarrow \frac{1}{|\mathcal{D}_a|}$
    **end if**
  **end for**
  Sample next assignment $x_a^{(t+1)} \sim \sigma_a^{(t+1)}$
  Send $x_a^{(t+1)}$ to all neighbors $N(a)$
**end for**

## E GRAPH COLORING

Graph coloring is a common benchmark for distributed constraint optimization. We evaluate all algorithms against graph coloring instances on two network topologies: random graphs and scalefree networks. Random graphs independently construct an edge between each node in the graph with probability $p$. Scalefree networks model real-world graph structures by iteratively adding vertices to $m$ existing vertices with probability proportional to their degree. We generate a test set for each graph structure with $n \in \{10, 20, 30, 50, 100\}$ number of nodes for a 3-coloring problem. For each value of $n$, we generate 6 random (two for each choice of $p \in \{\frac{4}{n}, \frac{5}{n}, \frac{6}{n}\}$) and 6 scalefree networks (two for each of $m \in \{2, 3, 4\}$), for a total of 60 sample problems. We run each algorithm for 10 trials on all instances. Each algorithm runs until convergence or until 30 seconds have elapsed. We tune hyperparameters for each algorithm by conducting a grid search over relevant values. We provide more details in Section F.2, and a list of hyperparameters used in Table A1.

We report the costs of mid-run solutions found by each algorithm throughout the experiment, and plot these values across time (Figure A1). Since the network size $n$ vastly changes the scale of the solution costs, we first rescale the costs so that the experiments with a particular $n$ value have a mean value of 1. We use gaussian kernel smoothing to plot each curve, and display the standard error as a shaded region around each curve. We show a breakdown of the performance by network type in Figure A13. We also report the performance by network size in Figure A14.

We found that across network types and sizes, online learning algorithms performed as well or better than existing incomplete DCOP algorithms, barring DSA. Most online learning variants performed comparably, including RM+, Damped DRM+, IR-PRM+, and FTRL. The damped variants of the algorithms underperformed compared to their standard counterparts, indicating that damping is not an effective technique with the RM variants. Overall, these results support that online learning algorithms are viable for DCOPs.

**Algorithm A2** Iterative pricing

**Input:** DCOP solver SOLVEDCOP, local schedulers $\{\mathcal{O}_a\}_{a\in\mathcal{A}}$, utilities $U$, step size $\alpha$, iteration limit $K$
Initialize prices $\lambda_{r,a}^{(1)} \leftarrow 0$ for all $r \in R, a \in \mathcal{A}$
**for** $t = 1, \dots, K$ **do**
  $z^{(t)} \leftarrow \text{SOLVEDCOP}\big(\{U_r - \lambda_{r,a}^{(t)}\}_{r,a}\big)$
  **for all** $a \in \mathcal{A}$ **do**
    $B_a^{(t)} \leftarrow \{r \in R \mid z_{r,a}^{(t)} = 1\}$
    $w_{r,a}^{(t)} \leftarrow U_r + \lambda_{r,a}^{(t)}$ for all $r \in B_a^{(t)}$
    $\left(Y_a^{(t)}, \widehat{y}_a\right) \leftarrow \mathcal{O}_a(B_a^{(t)}, w_a^{(t)})$
    Set $y_{r,a}^{(t)} \leftarrow 1$ if $r \in Y_a^{(t)}$, and 0 otherwise
  **end for**
  $\widehat{y}^{(t)} \leftarrow \left(\widehat{y}_a^{(t)}\right)_{a\in\mathcal{A}}$
  **if** $z_{r,a}^{(t)} = y_{r,a}^{(t)}$ for all $r \in R, a \in \mathcal{A}$ **then**
    **return** $\widehat{y}^{(t)}$
  **end if**
  **for all** $(r, a) \in R \times \mathcal{A}$ **do**
    $g_{r,a}^{(t)} \leftarrow z_{r,a}^{(t)} - y_{r,a}^{(t)}$
    $\lambda_{r,a}^{(t+1)} \leftarrow \lambda_{r,a}^{(t)} + \alpha \cdot g_{r,a}^{(t)}$
  **end for**
**end for**
**return** $\widehat{y}^{(K)}$

**Algorithm A3** Constraint generation

**Input:** DCOP solver SOLVEDCOP, local schedulers $\{\mathcal{O}_a\}_{a\in\mathcal{A}}$, utilities $U$, iteration limit $K$
Initialize constraint set $\mathcal{G} \leftarrow \varnothing$
**for** $t = 1, \dots, K$ **do**
  $z^{(t)} \leftarrow \text{SOLVEDCOP}(\mathcal{G})$
  **for all** $a \in \mathcal{A}$ **do**
    $B_a^{(t)} \leftarrow \{r \in R \mid z_{r,a}^{(t)} = 1\}$
    $\left(Y_a^{(t)}, \widehat{y}_a^{(t)}\right) \leftarrow \mathcal{O}_a(B_a^{(t)}, U)$
    **if** $Y_a^{(t)} \neq B_a^{(t)}$ **then**
      Add constraint $\sum_{r\in B_a^{(t)}} z_{r,a} \leqslant |B_a^{(t)}| - 1$ to $\mathcal{G}$
    **end if**
  **end for**
  $\widehat{y}^{(t)} \leftarrow \left(\widehat{y}_a^{(t)}\right)_{a\in\mathcal{A}}$
  **if** no constraints were added **then**
    **return** $\widehat{y}^{(t)}$
  **end if**
**end for**
**return** $\widehat{y}^{(K)}$

# F ABLATION STUDIES

## F.1 HYPERPARAMETER CHOICE FOR GRAPH COLORING

For our initial graph coloring study, we consider a grid of relevant hyperparameter values for each tested algorithm, and select the best performing value to use for our main experiment. Results for hyperparameter searches are obtained from generated networks with $n \in \{10, 20, 30, 50, 100\}$ number of nodes. For each value of $n$, we generate 5 random and 5 scalefree networks, and run 10 trials on our 50 instances. We report mid-run solution costs on a log time scale.

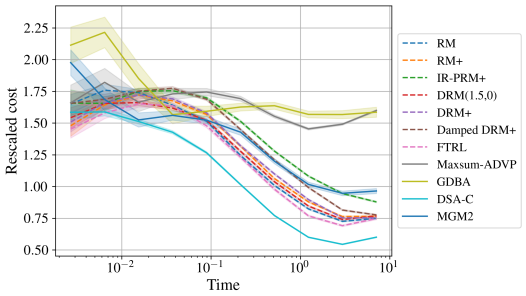



Figure A1: Cost of solution found over time for graph coloring problems across all network types.

### F.1.1 RM Damping

When comparing different levels of damping in the RM algorithm, we find that using no damping performs the best (Figure A2).

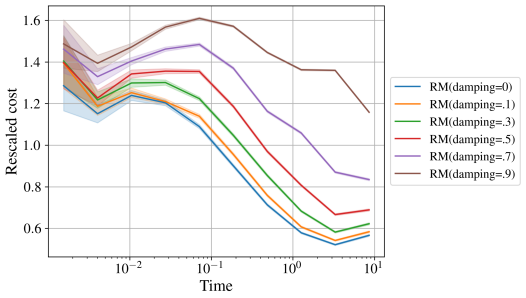



Figure A2: Cost of solution found over time for different levels of damping in RM.

### F.1.2 RM Inertia

When comparing different levels of inertia in the RM algorithm, we find that using no inertia performs the best (Figure A3). Combined with the previous result, this indicates that when paired with online algorithms, the increased stability of damping and inertia algorithm variants comes with a loss in performance.

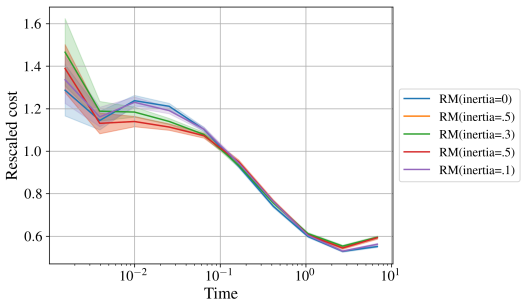



Figure A3: Cost of solution found over time for different levels of inertia in RM.

### F.1.3 DRM $\alpha$ and $\beta$

When comparing different values for $\alpha$ and $\beta$ in the DRM algorithm, we find there is no major difference in performance (Figure A4). In our main experiments, we use $\alpha = 1.5$, $\beta = 0$ since these values are often used.

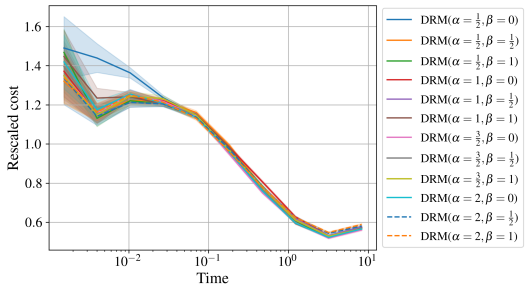



Figure A4: Cost of solution found over time for different $\alpha$ and $\beta$ values in DRM.

### F.1.4 DSA

When comparing different variants of DSA (as well as different $p$ parameters), we find that DSA-C with $p = 0.3$ or $p = 0.5$ perform best (Figure A5).

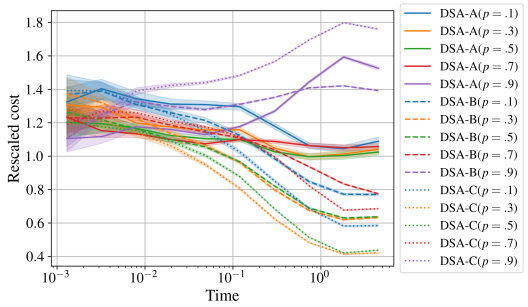



Figure A5: Cost of solution found over time for different $p$ values in DSA, across different variants.

### F.1.5 FTRL

When comparing different values for $\eta$ in the FTRL algorithm, we find there is no major difference in performance (Figure A6). In our main experiments, we use $\eta = 1$ since this value is often used.

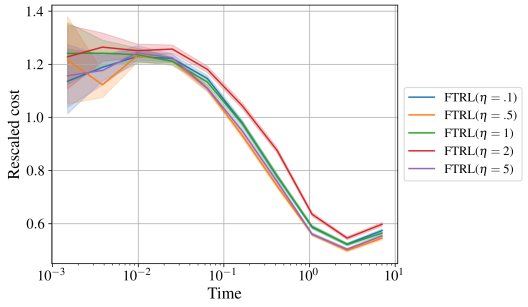



Figure A6: Cost of solution found over time for different $\eta$ values in FTRL.

### F.1.6 MAXSUM DAMPING

When comparing different levels of damping in the Maxsum algorithm, we find that using a damping parameter of $0.8$ performs the best (Figure A7).

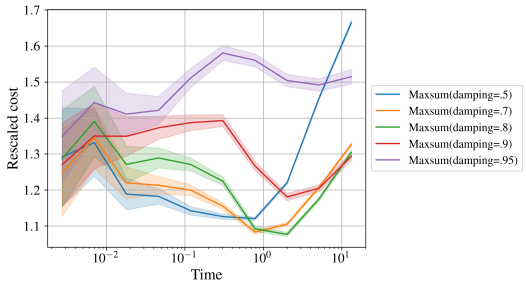



Figure A7: Cost of solution found over time for different levels of damping in Maxsum.

### F.1.7 MAXSUM STABILITY

When comparing different levels of stability in the Maxsum algorithm, we find that using a stability parameter of $0.01$ performs the best (Figure A8).

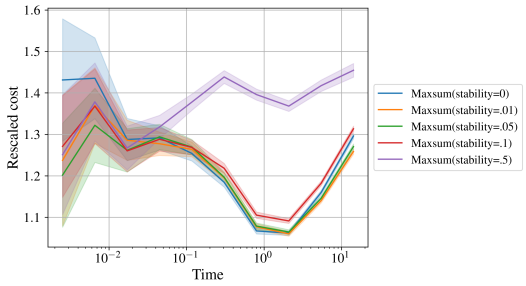



Figure A8: Cost of solution found over time for different stability values in Maxsum.

### F.1.8 MGM

When comparing different values of threshold in the MGM2 algorithm, we find that using a stability parameter of $0.5$ performs the best (Figure A9).

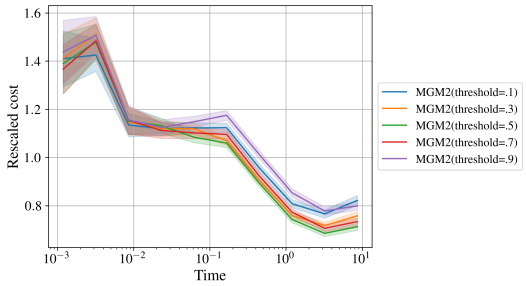



Figure A9: Cost of solution found over time for different threshold values in MGM2.

## F.2 NON CONTEXT-BASED ALGORITHM VARIANTS

We compare the difference between the context-based and non context-based versions of each algorithm when used in large-scale COSPs. For each algorithm variant, we set max iterations to 25 and tune the step size $\alpha$. We take the average across 20 COSP problems, with 20 trials each. We find that across all algorithms, using the context-based variant improves performance in the iterative pricing framework, and hurts performance in the constraint generation framework. The best combination

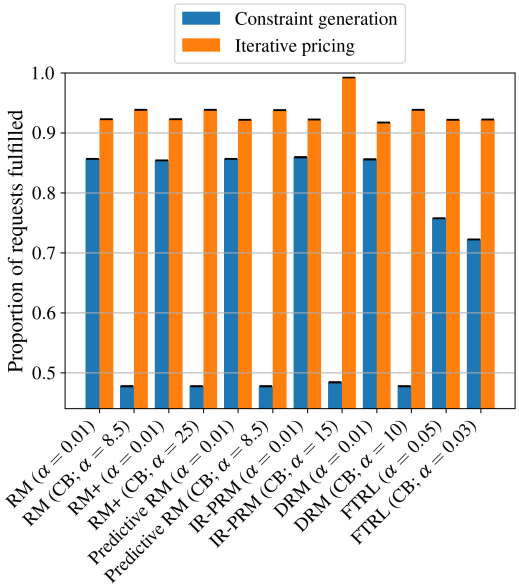
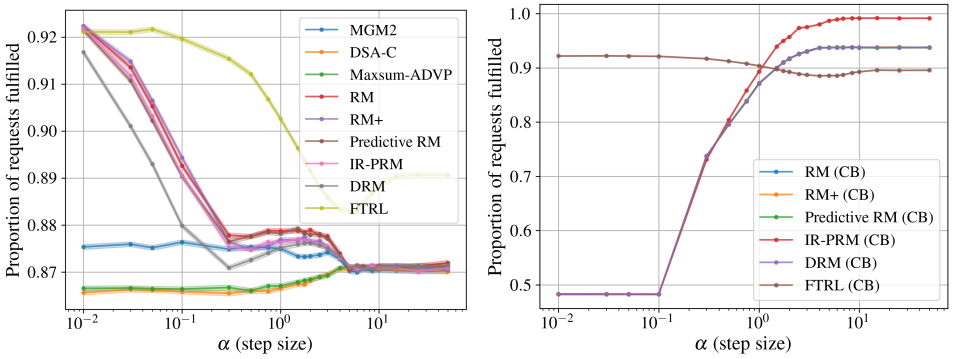



Figure A10: Utility achieved by standard and context-based (CB) variants of online learning algorithms. Standard error of the sample mean is displayed.

found is context-based IR-PRM used with iterative pricing, which fulfills 99.2% of requests on average (Fig. A10). These results support our findings in the paper, where iterative pricing outperforms constraint generation when paired with any online learning algorithm. To keep our plots clean, we display only the context-based variant of each algorithm in our main results.

### F.3 Step size for iterative pricing




(a) Default algorithm variants

(b) Context-based variants

Figure A11: Utility achieved by the iterative pricing framework using different values of $\alpha$. Standard error of the sample mean is displayed as the shaded region.

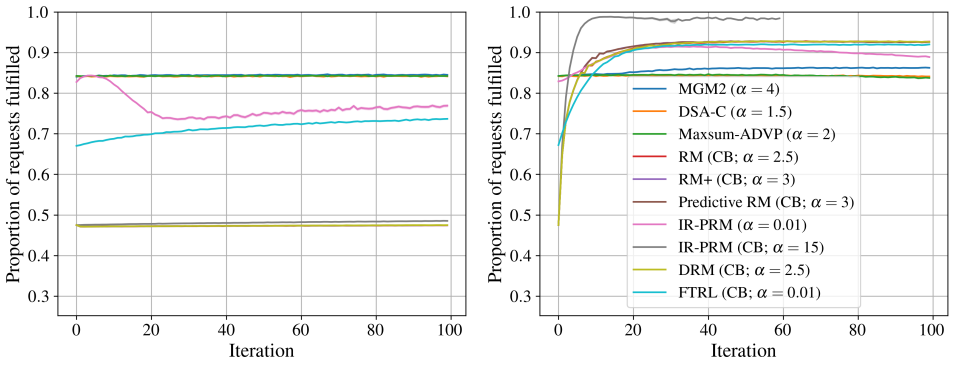
While constraint generation is parameter free, iterative pricing requires us to set a step size (which we denote $\alpha$, and keep fixed throughout the algorithm). Experimentally, we find that for iterative pricing in a large COSP, the optimal $\alpha$ depends on the algorithm used. When inspecting the graph of performance for each $\alpha$ value (Fig. A11), there appear to be four general shapes of curves.

- All context-based online learning algorithms apart from FTRL improve performance as $\alpha$ gets large.

- Context-based FTRL and all non context-based variants of online learning algorithms appear to have a peak near $\alpha = 0$, and a local optima at a larger value of $\alpha$ (around $\alpha = 15$ for FTRL, and $\alpha = 1.8$ for all others).
- Both DSA-C and Maxsum-ADBP have a slight boost in performance for $\alpha > 3$ but are otherwise unaffected.
- MGM2 has a slight dip in performance for $\alpha > 3$ and is otherwise unaffected.

In general, the $\alpha$ value chosen has drastic impact on the performances of most algorithms. To account for this, we use the optimal alpha values (among the sweep that we performed) for each algorithm when doing our comparisons.

### F.4 Number of iterations




(a) Constraint generation (b) Iterative Pricing

Figure A12: Utility achieved per iteration. Standard error of the sample mean is displayed as the shaded region.

To determine the setting of max iterations, we inspect the utility obtained per iteration. When plotting up to iteration 100 with optimal $\alpha$ values[2] (Fig. A12), we find that most of our considered algorithms have achieved close to their optimal score by about iteration 25. The only exception is context-based FTRL under constraint generation. Since that particular counterexample fails to achieve high performance anyway, we use 25 as our maximum number of iterations in our experiments.

## G Final hyperparameter settings

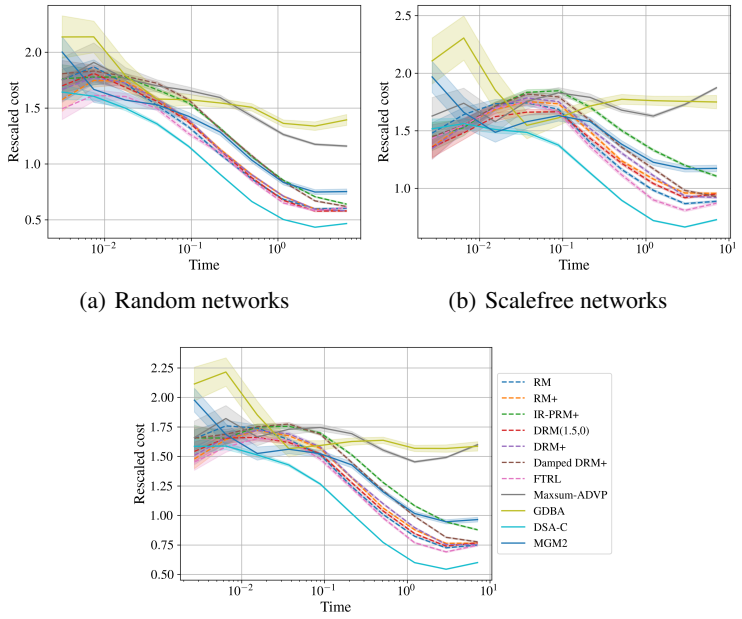
In tables A1 and A2, we list the final hyperparameters for each experiment in the paper.

## H Compute environment

All experiments were conducted in Python 3.8.20 on a Linux cluster. Compute nodes were equipped with dual AMD EPYC 7252 CPUs (16 physical cores, 32 hardware threads) and 512 GB of RAM. The graph coloring experiments were implemented in PyDCOP, an open-source Python library for DCOPs. The satellite scheduling experiments used a custom Python evaluation framework. The local scheduling oracles leverage the OR-Tools library and typically executed in milliseconds. All experiments were conducted using unseeded random number generators.

## I Omitted figures

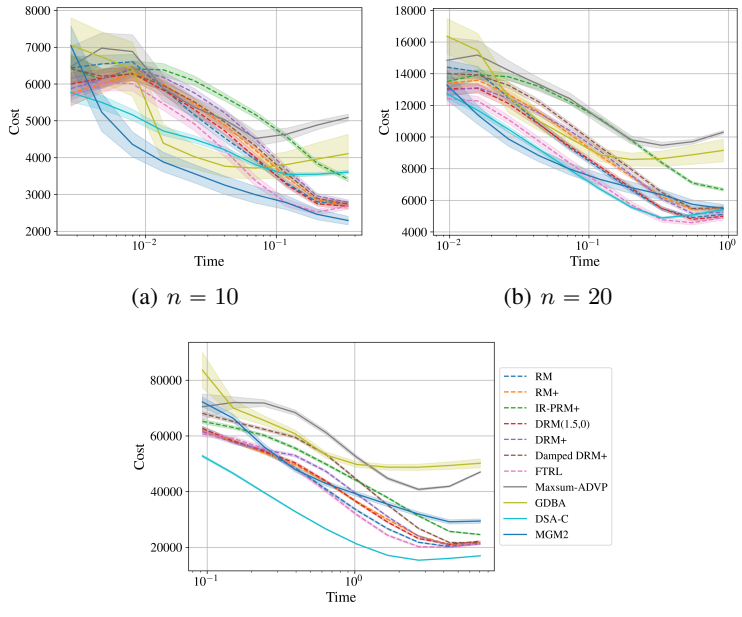
[2]These $\alpha$ values are optimized for max iterations set to 100, so they are different from our main experiments.




(a) Random networks (b) Scalefree networks

(c) All networks

Figure A13: Cost of solution found over time for graph coloring problems by network type.




(a) $n = 10$ (b) $n = 20$

(c) $n = 100$

Figure A14: Cost of solution found over time for graph coloring problems by network size.

| Algorithm | Hyperparameter | Value |
|---|---|---|
| DRM | $\alpha$ (Positive discount) | 1.5 |
| | $\beta$ (Negative discount) | 0 |
| DRM+ | $\alpha$ (Positive discount) | 1.5 |
| Damped DRM+ | $\alpha$ (Positive discount) | 1.5 |
| | Damping | 0.3 |
| FTRL | $\eta$ (Regularization) | 1 |
| Maxsum-ADVP | Damping | 0.8 |
| | Stability | 0.01 |
| GDBA | Cost modification | Additive |
| | Increase mode | E |
| | Constraint violation | Non-zero |
| DSA-C | Variant | C |
| | $p$ (Value change probability) | 0.5 |
| MGM2 | Threshold | 0.5 |
| | Move favored | Unilateral |

Table A1: Algorithm hyperparameters for graph coloring experiments.

| Algorithm | Hyperparameter | Value |
|---|---|---|
| MGM2 | Threshold | 0.5 |
| | Move favored | Unilateral |
| | $\alpha$ (Step size) | 0.1 |
| DSA-C | Variant | C |
| | $p$ (Value change probability) | 0.5 |
| | $\alpha$ (Step size) | 8.5 |
| Maxsum-ADVP | Damping | 0.5 |
| | Stability | 0.0001 |
| | $\alpha$ (Step size) | 50 |
| Context-based RM | $\alpha$ (Step size) | 8.5 |
| Context-based RM+ | $\alpha$ (Step size) | 25 |
| Context-based Predictive RM | $\alpha$ (Step size) | 8.5 |
| IR-PRM | $\alpha$ (Step size) | 0.01 |
| Context-based IR-PRM | $\alpha$ (Step size) | 15 |
| Context-based DRM | $\alpha$ (Positive discount) | 1.5 |
| | $\beta$ (Negative discount) | 0 |
| | $\alpha$ (Step size) | 10 |
| Context-based FTRL | $\alpha$ (Step size) | 0.03 |

Table A2: Algorithm hyperparameters for satellite scheduling experiments.