Learning-Augmented Decentralized Online Convex Optimization in Networks

PENGFEI LI, University of California, Riverside, United States
JIANYI YANG, University of Houston, United States
ADAM WIERMAN, California Institute of Technology, United States
SHAOLEI REN, University of California, Riverside, United States

This paper studies learning-augmented decentralized online convex optimization in a networked multi-agent system, a challenging setting that has remained under-explored. We first consider a linear learning-augmented decentralized online algorithm (LADO-Lin) that combines a machine learning (ML) policy with a baseline expert policy in a linear manner. We show that, while LADO-Lin can exploit the potential of ML predictions to improve the average cost performance, it cannot have guaranteed worst-case performance. To address this limitation, we propose a novel online algorithm (LADO) that adaptively combines the ML policy and expert policy to safeguard the ML predictions to achieve strong competitiveness guarantees. We also prove the average cost bound for LADO, revealing the tradeoff between average performance and worst-case robustness and demonstrating the advantage of training the ML policy by explicitly considering the robustness requirement. Finally, we run an experiment on decentralized battery management. Our results highlight the potential of ML augmentation to improve the average performance as well as the guaranteed worst-case performance of LADO.

CCS Concepts: • **Computing methodologies** → **Machine learning**; **Distributed computing methodologies**.

ACM Reference Format:
Pengfei Li, Jianyi Yang, Adam Wierman, and Shaolei Ren. 2024. Learning-Augmented Decentralized Online Convex Optimization in Networks. *Proc. ACM Meas. Anal. Comput. Syst.* 8, 3, Article 38 (December 2024), 42 pages. https://doi.org/10.1145/3700420

1 Introduction

This paper studies the problem of decentralized online convex optimization in networks, where inter-connected agents must individually select actions with sequentially-revealed local online information and delayed feedback from their neighboring agents. We consider a setting where, at each step, agents must decide on an action using local information while collectively seeking to minimize a global cost consisting of the sum of (i) the agents' node costs, which capture the local instantaneous effects of the individual actions; (ii) temporal costs, which capture the (inertia) effects of local temporal action changes; and (iii) spatial costs, which characterize the loss due to unaligned actions of two connected neighboring agents in the network. This problem models a wide variety of networked systems with numerous applications, such as decentralized control in

Pengfei Li, Jianyi Yang, and Shaolei Ren were supported in part by the NSF under grants CNS-2007115 and CCF-2324941. Adam Wierman was supported by NSF grants CCF-2326609, CNS-2146814, CPS-2136197, CNS-2106403, and NGSDI-2105648 as well as funding from the Resnick Sustainability Institute.
Authors' Contact Information: Pengfei Li, pli081@ucr.edu, University of California, Riverside, Riverside, California, United States; Jianyi Yang, jyang239@ucr.edu, University of Houston, Houston, Texas, United States; Adam Wierman, adamw@caltech.edu, California Institute of Technology, Pasadena, California, United States; Shaolei Ren, sren@ece.ucr.edu, University of California, Riverside, Riverside, California, United States.

ACM 2476-1249/2024/12-ART38
https://doi.org/10.1145/3700420

power systems [1–3], spectrum management in multi-user wireless networks [4–6], multi-product pricing in revenue management [7, 8], among many others.

While centralized algorithms can effectively minimize the global cost, decentralized optimization offers advantages including resilience to single-point failure and lower computational complexity. Despite the recent progress (e.g., [9, 10]), online optimization in decentralized settings is inherently more challenging due to limited information availability. Agents must coordinate their actions across the network to minimize the global cost, while they usually lack complete knowledge of future costs or the actions of their neighbors. This information gap presents significant challenges for decentralized online optimization, compared to its centralized counterparts [11, 12].

To address these challenges, decentralized online convex optimization has been studied under various settings. For example, online algorithms for the special single-agent case [12–18] have been utilized as the basis for decentralized optimization to minimize the worst-case regret or competitive ratio in the multi-agent case [9, 10, 19–22]. However, because these algorithms must make conservative decisions to mitigate potentially adversarial uncertainties, they often do not perform well in terms of the average cost. In contrast, online optimizers based on machine learning (ML) can improve the average performance by exploiting the distributional information for various problems, e.g., [23–26], including multi-agent networked systems [27–29]. But, ML-based optimizers typically lack robustness guarantees and can result in a very high cost in the worst case (due to, e.g., out-of-distribution inputs), which makes them unsuitable for mission-critical applications.

The field of learning-augmented algorithms has emerged in recent years with the goal of providing "best of both worlds" guarantees: near-optimal performance with accurate ML predictions and guaranteed robustness with inaccurate predictions. These algorithms have demonstrated success in various online settings e.g., [11, 30–34]. However, existing learning-augmented algorithms [11, 31, 35] primarily focus on centralized scenarios, making their adaptation to decentralized setups technically challenging due to spatial uncertainties arising from limited information availability. Moreover, these algorithms predominantly focus on worst-case performance guarantees, with less emphasis on average cost performance.

Contributions. We study the challenging and under-explored setting of decentralized online optimization in networks. We first consider a linear learning-augmented decentralized online optimization algorithm (`LADO-Lin`), which linearly combines a potentially untrusted ML policy with a trusted baseline policy (called "expert"). We show that `LADO-Lin` can exploit the power of ML predictions to improve the average cost performance when the ML predictions are of sufficiently high quality, but it *cannot* offer guaranteed competitiveness or robustness in the worst case when ML predictions are of arbitrarily low quality.

To overcome `LADO-Lin`'s lack of guaranteed competitiveness, we introduce and analyze a novel algorithm, `LADO`, that adaptively combines an ML policy with an expert policy based on the actual online costs. The key idea behind `LADO` is to leverage the baseline expert policy to safeguard online actions to avoid too greedily following ML predictions that may not be robust. In a decentralized setting, the primary design challenge is managing spatial information inefficiency, as agents lack prior knowledge of their neighbors' actions. More concretely, the spatial cost is dependent on the actions of neighboring agents, making it difficult for a single agent to evaluate it in isolation. To address this and associated spatial cost uncertainties, we propose a novel spatial cost decomposition that adaptively splits the shared spatial cost between connected agents, enabling each agent to safeguard its own actions based on local information. To ensure non-empty action sets and maintain robustness when deviating from the trusted expert policy, we also introduce temporal reservation costs to address worst-case future cost uncertainties.

Our main results provide worst-case and average cost bounds for `LADO` (see Theorems 5.1, 5.2, and B.1). We also show the worst-case robustness and consistency of `LADO-Lin` and `LADO` (see

Corollary 3.4 and Theorem 5.3). Importantly, unlike most of the prior work that assumes a black-box ML model trained as a standalone optimizer, our results also provide average cost bounds for training ML by explicitly considering how the ML policy will be used. Our results quantify the improvement obtained by explicitly accounting for the robustness step in ML training.

To evaluate the effectiveness of LADO-Lin and LADO, we conduct experiments on decentralized battery management for sustainable data centers, with a networked battery system of up to 120 nodes. Our results demonstrate the empirical benefits of our algorithms over existing baselines across various network topologies. Both LADO-Lin and LADO, when augmented with ML, consistently achieve strong average cost performance under various network topologies. Moreover, LADO offers guaranteed robustness even when the ML predictions have low quality.

To summarize, the main contributions of our work are as follows. First, unlike existing learning-augmented algorithms, we focus on the more challenging setting of decentralized optimization, where agents make online decisions with delayed information about their neighbors' actions. Second, to guarantee worst-case robustness of LADO against a given policy in our decentralized setting, our design of robust action sets includes novel adaptive spatial cost splitting, which is a novel technique and differs from the design in a centralized setting. Last but not least, we rigorously analyze and also empirically show the performance of LADO-Lin and LADO in terms of their average-case and worst-case costs.

2 Problem Formulation

We study the setting introduced in [36] (where there is no ML policy augmentation) and consider decentralized online convex optimization in a network with $V = |\mathcal{V}|$ agents/nodes belonging to the set $\mathcal{V}$. If two agents have interactions with each other, there exists an edge between them. Thus, the networked system can be represented by an undirected graph $(\mathcal{V}, \mathcal{E})$, with $\mathcal{E}$ being the set of edges. Each problem instance (a.k.a. episode) consists of T sequential time steps.

At step $t = 1, \cdots, T$, each agent v selects an irrevocable action $x_t^v \in \mathbb{R}^n$. We denote $x_t = [x_t^1, \cdots, x_t^V]$ as the action vector for all agents at step t, where the superscript v represents the agent index whenever applicable. After x_t is selected for step t, the network generates a global cost $g_t(x_t)$ which consists of the following three parts.

• **Node cost $f_t^v(x_t^v)$:** Each individual agent incurs a node cost $f_t^v(x_t^v)$, which only relies on the action of a single agent v at step t and measures the effect of the agent's decision on itself.

• **Temporal cost $c_t^v(x_t^v, x_{t-1}^v)$:** It couples the two temporal-adjacent actions of a single agent v and represents the effect of temporal interactions to smooth actions over time.

• **Spatial cost $s_t^{(v,u)}(x_t^v, x_t^u)$:** It is incurred if an edge exists between two agents v and u, capturing the loss due to unaligned actions of two connected agents.

This formulation applies directly to many real-world applications [36]. For example, in geo-distributed cloud resource management, each data center is an agent whose server provisioning decision (i.e., the number of on/off servers) incurs a node cost that captures its local operational cost [37]. The temporal cost penalizes frequent servers on/off to avoid excessive wear-and-tear (a.k.a., switching costs) [37]. Meanwhile, each data center's decision results in an environmental footprint (e.g., carbon emission and water consumption) [38]. Thus, the added spatial cost mitigates inequitable environmental impacts in different locations to achieve environmental justice, which is a crucial consideration in many corporates' Environmental, Social, and Governance (ESG) strategies [39]. In Section 2.2, we provide more modeling details, and explore two other applications: decentralized battery management for sustainable computing and multi-product dynamic pricing.

Next, we make the following common assumptions for online optimization, e.g., [36, 40].

Assumption 2.1. *The node cost $f_t^v: \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is β-strongly convex and ℓ_f-smooth.*

Assumption 2.2. *The temporal interaction cost* $c_t^v: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ *is convex and* ℓ_T*-smooth.*

Assumption 2.3. *The spatial interaction cost* $s_t^{v,u}: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ *is convex and* ℓ_S*-smooth.*

The convexity assumption is needed for analysis, while smoothness (i.e., Lipschitz-continuous gradients) ensures that the costs will not vary unboundedly when the actions change [36].

The networked agents collaboratively minimize the total global cost over T time steps defined as:

$$cost(x_{1:T}) = \sum_{t=1}^{T} g_t(x_t) = \sum_{t=1}^{T} \sum_{v\in\mathcal{V}} f_t^v(x_t^v) + \sum_{t=1}^{T} \sum_{v\in\mathcal{V}} c_t^v(x_t^v, x_{t-1}^v) + \sum_{t=1}^{T} \sum_{(v,u)\in\mathcal{E}} s_t^{(v,u)}(x_t^v, x_t^u),$$

where $g_t(x_t) = \sum_{v\in\mathcal{V}} f_t^v(x_t^v) + \sum_{v\in\mathcal{V}} c_t^v(x_t^v, x_{t-1}^v) + \sum_{(v,u)\in\mathcal{E}} s_t^{(v,u)}(x_t^v, x_t^u)$ is the total cost at time t. In practice, we consider a weighted sum of the node, temporal, and spatial costs. These weights assigned to each cost component reflect their relative importance in the overall cost metric. For the convenience of presentation, we normalize the weight of the node cost to 1 and incorporate the remaining weights directly into the individual cost terms. With a slight abuse of notation, we also denote $g_t = \{f_t^v, c_t^v, s_t^{(u,v)}, v \in \mathcal{V}, (u,v) \in \mathcal{E}\}$ as the cost function information for step t, and $g_{1:T} = [g_1, \cdots, g_T] \in \mathcal{G}$ as all the exogenously-determined information for the entire problem instance where $\mathcal{G}$ is the set of all possible $g_{1:T}$.

Our goal is to find a decentralized learning-augmented online policy π_v for each agent v that maps the local available information (to be specified in Section 3.1) to its action x_t^v at time t. For notational convenience, we also denote $\pi = [\pi_1, \cdots, \pi_V]$ as the combined policy for the network.

2.1 Performance Metrics

We consider the following two performance metrics — average cost and λ-competitiveness.

Definition 2.4 (Average cost)**.** *Given a decentralized online policy* $\pi = [\pi_1, \cdots, \pi_V]$*, the average cost is* $AVG(\pi) = \mathbb{E}_{g_{1:T}}[cost(\pi, g_{1:T})]$*, where the information* $g_{1:T}$ *follows a distribution* $\mathcal{P}_{g_{1:T}}$*.*

Definition 2.5 (λ-competitive to $\pi^\dagger$)**.** *For* $\lambda > 0$*, an online policy* $\pi = [\pi_1, \cdots, \pi_V]$ *is* λ*-competitive against a baseline policy* $\pi^\dagger$ *if* $cost(\pi, g_{1:T}) \leq (1+\lambda)cost(\pi^\dagger, g_{1:T}))$ *holds for any* $g_{1:T} \in \mathcal{G}$*.*

The average cost measures the decision quality of the decentralized policy π in typical cases, whereas the λ-competitiveness shows the worst-case competitiveness in terms of the cost ratio of the global cost of π to a given trusted baseline policy $\pi^\dagger$ (which is also referred to as an *expert* policy). Our definition of λ-competitiveness against $\pi^\dagger$ is both general and common in the literature on learning-augmented online algorithms as well as online control [11, 41, 42], where competitiveness is defined against a given baseline policy $\pi^\dagger$ [11, 30, 43]. Importantly, for our problem, there exist various expert policies $\pi^\dagger$ (e.g., localized prediction control [36]) with bounded cost ratios against the oracle policy $OPT = \pi^*$ that minimizes the global cost with all offline information. As a result, by considering an expert policy with a competitive ratio of $\rho_{\pi^\dagger}$, our policy π is also competitive against the optimal oracle, i.e., $cost(\pi, g_{1:T}) \leq \rho_{\pi^\dagger}(1+\lambda)cost(OPT, g_{1:T}))$ for any $g_{1:T} \in \mathcal{G}$. Alternatively, the expert policy $\pi^\dagger$ can be viewed as a policy prior currently in use [44], while the new learning-augmented policy π must no worse than $(1+\lambda)$-times the policy prior in terms of the cost for any problem instance.

The average cost and worst-case competitiveness metrics are different and complementary to each other [11, 45]. Here, we take a competitiveness-constrained approach. Specifically, given both an ML-based optimizer and an expert algorithm as advice, we aim to find a learning-augmented policy $\pi = [\pi_1, \cdots, \pi_V]$ to minimize the average cost subject to the λ-competitiveness constraint:

$$\min_{\pi} \mathbb{E}_{g_{1:T}}[cost(\pi, g_{1:T})], \ s.t., \ cost(\pi, g_{1:T}) \leq (1+\lambda)cost(\pi^\dagger, g_{1:T}), \ \forall g_{1:T} \in \mathcal{G}. \tag{1}$$

While offline-trained ML-based policies (e.g., based on multi-agent reinforcement learning [27–29]) can potentially minimize the average cost, they may not satisfy λ-competitiveness in the worst case. In fact, it is well-known that due to the statistical nature, ML-based policies can have arbitrarily bad performance in certain (possibly rare) cases, especially when the testing problem instance is very distinct from those training instances [35]. Thus, we adopt a learning-augmented approach where we integrate an ML-based policy into decision-making while using a trusted expert policy to safeguard our online decisions.

2.2 Application Examples

To make our model concrete, we present the following application examples. Readers are also referred to [36] for additional examples.

Decentralized battery management for sustainable computing. Traditionally, data centers rely on fossil fuels such as coal or natural gas to power their operation. Thus, with the proliferating demand for cloud computing and artificial intelligence services, there have been increasing environmental concerns with data centers' growing carbon emissions. As such, it is important to find ways to reduce data centers' carbon footprint and mitigate their environmental impact — decarbonizing data centers. While renewable energy sources, such as solar and wind, are natural alternatives for sustainable data centers, their availability can be highly fluctuating subject to weather conditions, thus imposing significant challenges to meet data centers' energy demands. Consequently, large energy storage consisting of multiple battery units has become essential to leverage intermittent renewable energy to power data centers for sustainable computing. Nonetheless, it is challenging to manage a large energy storage system to achieve optimal efficiency. Specifically, while each battery unit is responsible for its own charging/discharging decisions to keep the energy level within a desired range (e.g., 20-80%) in decentralized battery management, the state-of-charge (SoC) levels across different battery units should also be maintained as uniform as possible to extend the overall battery lifespan and energy efficiency [46]. This problem can be well captured by converting a canonical form into our model: each battery unit decides its SoC level by charging/discharging and incurs a node cost (i.e., SoC level deviating from the desired range) and a temporal cost (i.e., SoC changes due to charging/discharging), and meanwhile there is a spatial cost due to SoC differences across different battery units.

More concretely, we consider an energy storage system that includes a set of battery units $\mathcal{V}$ interconnected through physical connections $\mathcal{E}$. For a battery unit $v \in \mathcal{V}$, the goal is to minimize the difference between the current SoC and a nominal value $\bar{x}_v$ plus a power grid's usage cost, which can be defined as a local objective: $\min_{u_{v,1:T}} \sum_{t=1}^{T} \|x_{v,t} - \bar{x}_v\|^2 + \sum_{t=1}^{T} b\|\xi_{v,t}\|^2$, where $\xi_{v,t}$ is the charging/discharging schedule from the power grid (i.e., $\xi_{v,t} > 0$ means drawing energy from the grid and $\xi_{v,t} < 0$ means returning energy to the grid) and b is the power grid's usage penalty cost. The time index for the first term starts at $t = 2$ as we assume a given initial state $x_{v,1}$ (i.e., the SoC cost at $t = 1$ is already given). The canonical form of the battery SoC dynamics follows by $x_{v,t} = A_v x_{v,t-1} + B_v \xi_{v,t} + C_v w_{v,t}$, where A_v denotes the self-degradation coefficient, B_v denotes the charging efficiency, $w_{v,t}$ is the data center's net energy demand from battery unit v (i.e., $w_{v,t} > 0$ means the data center's energy demand exceeds the available renewables and $w_{v,t} < 0$ otherwise), C_v denotes the conversion coefficient (inversely proportional to the capacity of battery unit v), which translates the net energy demand to the change in battery SoC.

Based on the physical connection $(u, v) \in \mathcal{E}$, the SoC difference between battery units u and v can lead to reduced performance and lifespan. For instance, the battery voltage difference caused by different SoCs may cause overheating problems or even battery damage [46]. Thus, to penalize the SoC difference between two interconnected battery units, we add a spatial cost $\sum_{(v,u)\in\mathcal{E}} c \cdot \|x_t^v - x_t^u\|^2$,

where c is the SoC difference penalty coefficient. Thus, the total control cost is

$$\min_{\{u_{v,1:T},\forall v\in\mathcal{V}\}} \sum_{t=1}^{T}\sum_{v\in\mathcal{V}} \|x_{v,t}-\bar{x}_v\|^2 + \sum_{t=1}^{T}\sum_{v\in\mathcal{V}} b\|\xi_{v,t}\|^2 + \sum_{t=1}^{T}\sum_{(v,u)\in\mathcal{E}} c\|x_t^v - x_t^u\|^2. \tag{2}$$

Next, we convert (2) into our formulation decentralized online convex optimization. At time t, we define $y_{v,t} = \bar{x}_v - A_v^t x_{v,1} - \sum_{i=1}^{t} A_v^{t-i} C_v w_{v,i}$ as the context parameter determined by all the previous states and online inputs, and $a_{v,t} = \sum_{i=1}^{t} A_v^{t-i} B_v \xi_{v,i}$ as the corresponding node v's online action in our model. Then, we define the node cost for v as $f_t^v(a_{v,t}) = \|a_{v,t} - y_{v,t}\|^2 = \|x_{v,t} - \bar{x}_v\|^2$, the temporal cost for v as $c_t^v(a_{v,t}, a_{v,t-1}) = \frac{b}{B_v^2}\|a_{v,t} - A_v a_{v,t-1}\|^2 = b\|\xi_{v,t}\|^2$, and the spatial cost for edge (v,u) as $s_t^{(v,u)} = c\|(a_{v,t} - a_{u,t}) - (y_{v,t} - y_{u,t}) + \bar{x}_v - \bar{x}_u\|^2 = c\|x_t^v - x_t^u\|^2$. By combining these three costs together, the total global cost becomes

$$\begin{aligned}\min_{\{a_{v,1:T},\forall v\in\mathcal{V}\}} \sum_{t=1}^{T}\Bigg(&\sum_{v\in\mathcal{V}} \|a_{v,t} - y_{v,t}\|^2 + \sum_{v\in\mathcal{V}} \frac{b}{B_v^2}\|a_{v,t} - A_v a_{v,t-1}\|^2 + \\ &\sum_{(v,u)\in\mathcal{E}} c\|(a_{v,t} - a_{u,t}) - (y_{v,t} - y_{u,t}) + \bar{x}_v - \bar{x}_u\|^2\Bigg),\end{aligned} \tag{3}$$

which has the same form as our formulation (1) if we view $a_{v,t}$ as node v's online action at time t.

Geographic server provisioning with environmental equity. Online service providers commonly rely on geographically distributed data centers in the proximity of end users to minimize service latency. Nonetheless, data centers are notoriously energy-intensive. Thus, given time-varying workload demands, the data center capacity (i.e., the number of active servers) needs to be dynamically adjusted to achieve energy-proportional computing and minimize the operational cost [37]. More specifically, each data center dynamically provisions its servers in a decentralized manner, based on which the incoming workloads are scheduled [47]. Naturally, turning on more servers in a data center can provide better service quality in general, but it also consumes more energy and hence negatively results in a higher environmental footprint (e.g., carbon and water, which both roughly increase with the energy consumption proportionally [38, 39]).

While it is important to reduce the total environmental footprint across geo-distributed data centers, addressing environmental *inequity* — mitigating locational disparity in terms of negative environmental consequences caused by data center operation — is also crucial as inequity can create significant business risks and unintended societal impacts [48]. Indeed, the emergence of data centers' environmental inequity has been recently compared to "historical practices of settler colonialism and racial capitalism" [49] and calls for attention from various environmental groups and policy think tanks [50, 51].

To address environmental inequity, we view each data center as a node v in our model. The data center v makes its own dynamic server provisioning decision x_t^v (i.e., the number of active servers, which can be treated as a continuous variable due to tens of thousands of servers in data centers), and incurs a node cost $f_t^v(x_t^v)$ that captures the local energy cost, environmental footprint, and service quality [37]. The temporal cost $c_t^v(x_t^v, x_{t-1}^v) = \|x_t^v - x_{t-1}^v\|^2$ captures the negative impact of switching servers on and off (e.g., wear-and-tear), which is also referred to as the switching cost in the data center literature [37]. Additionally, the spatial cost $s_t^{(v,u)}(x_t^v, x_t^u)$ can be written as $s_t^{(v,u)}(x_t^v, x_t^u) = \|e_t^v x_t^v - e_t^u x_t^u\|^2$ where e_t^v is the weighted environmental "price" (e.g., water usage efficiency scaled by the average per-server energy) in data center v. Thus, the spatial cost addresses environmental justice concerns by penalizing difference between data center v and data center u in terms of their environmental footprint. As a result, by considering weighted sums of the node

costs, temporal costs, and spatial costs, our model applies to the problem of geographic server provisioning with environmental justice, which is emerging as a critical concern in the wake of increasingly hyperscale data centers that may leave certain local communities to disproportionately bear the negative environmental consequences.

Multi-product dynamic pricing. As digital marketplaces continue to grow, offering a diverse range of products or services becomes inevitable for businesses that seek to cater to diverse consumer preferences. As such, a dynamic multi-product pricing policy is vital for revenue management. For instance, the online platform may modify product prices multiple times within a single day, considering the estimated user demand, competitor prices and inventory dynamics. Nevertheless, due to the intricate relationships products share within the marketplace, it is a challenging task to set prices. For example, for complementary products (e.g. laptop vs headphones), a special offer on a certain laptop may stimulate the demand for headphones and other accessories. Additionally, customers may also observe historical prices, which affects their willingness to buy. In other words, the current demand for a certain product can be temporally coupled with the previous prices [36]. In our framework, the temporal interaction and spatial interaction costs model the effects of multiple product relationships and user behaviors, respectively. More specifically, at time t, suppose that the price of product v is x_t^v. Then, under a linear demand model, the total revenue is represented by

$$\sum_{t=1}^{T}\left[\sum_{v\in\mathcal{V}} x_t^v(a_t^v - k_t^v x_t^v) + \sum_{v\in\mathcal{V}} x_t^v(b_t^v x_{t-1}^v) + \sum_{(u,v)\in\mathcal{E}} x_t^v(\xi_t^{(u,v)} x_t^u)\right] \tag{4}$$

where $a_t^v - k_t^v x_t^v$ models the nominal demand under price x_t, b_t^v quantifies the effect of the previous price, the coefficient $\xi_t^{(u,v)}$ denotes the spatial relationships between a product pair (u, v). Under realistic parameter settings, this problem can be converted into our model of decentralized online convex optimization (see [36] for details).

3 LADO-Lin: Linearly Combining ML Advice and Expert Advice

To begin, we study a simple approach toward designing a learning-augmented algorithm, which uses a fixed linear combination of ML-based untrusted policy and the trusted expert policy, i.e., Linear Learning-Augmented Decentralized Online Optimization (LADO-Lin). We analyze the performance of LADO-Lin and highlight its key limitation: the lack of guaranteed worst-case competitiveness.

3.1 Local Information Availability

Our goal is to effectively use both ML-based advice and expert advice to solve (1) in a decentralized online manner. In our setting, each agent $v \in \mathcal{V}$ has access to a decentralized online ML policy $\tilde{\pi}_v$ and a decentralized online expert policy $\pi_v^\dagger$, which produce actions $\tilde{x}_t^v$ and $x_t^{v,\dagger}$ at time $t = 1, \cdots, T$, respectively, based on local online information. Then, given $\tilde{x}_t^v$ and $x_t^{v,\dagger}$, the agent v chooses its actual action x_t^v using LADO-Lin.

More specifically, the following online information is revealed to each agent v at step t: node cost function f_t^v, temporal cost function c_t^v, spatial cost function $s_{t-1}^{(v,u)}$, connected agents' actions x_{t-1}^u and their corresponding expert actions $x_{t-1}^{u,\dagger}$ for $(v, u) \in \mathcal{E}$. That is, at the beginning of step t, each agent v receives its own node cost and temporal cost functions for time t, and also the spatial cost along with the actual/expert actions from the neighboring agents connected to agent v for time $t-1$. Thus, before choosing an action at time t, all the local information available to agent v can be summarized as

$$I_t^v = \{f_{1:t}^v, c_{1:t}^v, s_{1:t-1}^{(v,u)}, x_{1:t-1}^u, x_{1:t-1}^{u,\dagger}, Z_t^v, (v,u) \in \mathcal{E}\}, \tag{5}$$

Algorithm 1 Learning-Augmented Online Decentralized Optimization for Agent $v \in \mathcal{V}$

Require: Expert policy $\pi_v^\dagger$, and ML policy $\tilde{\pi}_v$

1: **for** $t = 1, \cdots, T$ **do**
2: Collect local online information I_t^v.
3: Obtain ML prediction $\tilde{x}_t^v$ and expert action $x_t^{v,\dagger}$ based on I_t^v, respectively.
4: Choose the action $x_t^v = \gamma\tilde{x}_t^v + (1-\gamma)x_t^{v,\dagger}$ in **LADO-Lin**, and $x_t = \psi_\lambda(\tilde{x}_t^v)$ by (9) in **LADO**.
5: **end for**

where Z_t^v captures the other applicable information (e.g., agent v's own actual/ML/expert actions in the past). Moreover, knowledge of cost functions over the next k temporal steps and/or r-hop agents in the network can further improve the competitiveness of expert policies [36] and, if available, be included in Z_t^v. Without loss of generality, we use I_t^v as the locally available information for agent v at time t. Additionally, the smoothness parameters ℓ_f, ℓ_c, and ℓ_s and robustness parameter λ are known to the agents as shared information.

Our information availability setting is in line with that considered by the prior literature on decentralized online convex optimization [36], except that each agent has access to both the ML advice and expert advice in our setting. Note also that there is a *separate* line of research of online convex optimization that assumes the node cost function f_t^v is only revealed to the agent at the end of time t [52], but they often have different design goals (e.g., sublinear regret compared to a *static* baseline policy) than our worst-case competitiveness guarantees against a *dynamic* baseline policy specified in Definition 2.5.

Most importantly, unlike in a centralized setting, an agent v must individually choose its irrevocable action x_t^v on its own — it cannot communicate its action x_t^v or its expert action $x_t^{v,\dagger}$ to its connected agent u until the next time step $t+1$. The one-step delayed feedback of the spatial costs and the actual/expert actions from the connected agents is commonly studied in decentralized online convex optimization [36] and crucially differentiates our work from the prior *centralized* learning-augmented algorithms, adding challenges for ensuring the satisfaction of the λ-competitiveness requirement.

3.2 Algorithm Design

In our problem, each individual agent $v \in \mathcal{V}$ is provided with the potentially untrusted ML advice $\tilde{x}_t^v$ and the trusted expert advice $x_t^\dagger$ at time time $t \in [1, T]$ based on its local online information I_t^v specified in (5). The assumption of an offline-trained predictor (i.e., ML policy in our case) is standard in learning-augmented algorithms [30, 32, 53, 54] as well as general learning-based optimizers [23, 26, 55]. For our problem, approaches such as multi-agent reinforcement learning [27–29] can be used to train ML policies for each agent. When the context is clear, we also interchangeably use ML *prediction* to refer to the ML action or advice.

Had we known which policy — the ML policy $\tilde{\pi}$ or the expert policy $\pi^\dagger$ — would be better for a problem instance in advance, the problem would become trivial and we just need to choose the better policy. But, this is not possible in an online setting. To exploit the potential of ML predictions by augmenting the expert advice $x_t^\dagger$ with the ML advice $\tilde{x}_t^v$, out first attempt is to construct a linear combination of the two advice for each agent, which is defined as follows:

$$x_t^v = \gamma\tilde{x}_t^v + (1-\gamma)x_t^{v,\dagger}, \quad \forall v \in \mathcal{V} \tag{6}$$

where $\gamma \in [0, 1]$ is the hyperparameter that reflects our confidence in ML predictions: the larger $\gamma \in [0, 1]$, the more we trust the ML advice. We refer to this algorithm as LADO-Lin, which is also

described in Algorithm 1. Note that, in Algorithm 1, we run the expert policy (e.g., the localized policy proposed in [36]) independently as if it is applied alone. Thus, the expert policy $\pi_v^\dagger$ does not need to use all the information in I_t^v.

3.3 Performance Analysis

We first analyze the performance of LADO-Lin in terms of its average cost bound as follows.

Theorem 3.1 (Average cost of LADO-Lin). *For any $\gamma \in [0,1]$, the average cost of* LADO-Lin *is upper bounded by*

$$AVG(\texttt{LADO-Lin}) \le \min\left\{\gamma AVG(\tilde{\pi}) + (1-\gamma)AVG(\pi^\dagger), \left(\sqrt{AVG(\tilde{\pi})} + (1-\gamma)\sqrt{\mathbb{E}_{g_{1:T}}\left[\sum_{t=1}^{T}\sum_{v\in\mathcal{V}}\frac{\ell_f + 2\ell_T + \ell_S D_v}{2}\|\tilde{x}_t^v - x_t^{v,\dagger}\|^2\right]}\right)^2\right\} \tag{7}$$

where $AVG(\tilde{\pi})$ and $AVG(\pi^\dagger)$ are the average costs of the ML policy and expert over the distribution $g_{1:T} \sim \mathcal{P}_{g_{1:T}}$, respectively.

In Theorem 3.1, the cost bound for LADO-Lin is given by the minimum of two terms: the first term is based on the convex property of the cost functions in our networked system, and the second term is derived in terms of the expected total distance between the ML and expert actions based on the smoothness of the costs. The proof is available in Appendix D.

Naturally, with a larger $\gamma \in [0,1]$, the cost of LADO-Lin is more determined by the cost of the ML policy $\tilde{\pi}$. In practice, the ML policy is often trained to minimize the average cost, while the expert policy is conservatively designed to address the worst case. Thus, for a well-trained ML policy, we typically have $\text{AVG}(\tilde{\pi}) < \text{AVG}(\pi^\dagger)$. This means that to minimize the average cost, we should choose $\gamma = 1$, i.e., purely following the ML advice.

While LADO-Lin can successfully exploit the potential of ML predictions by setting a large $\lambda \in [0,1]$, it hardly meets the λ-competitiveness constraint (Definition 2.5). Indeed, unless the ML policy itself is sufficiently close to the optimal policy for *any* problem instance $g_{1:T} \in \mathcal{G}$, LADO-Lin cannot meet the λ-competitiveness constraint. This is formalized as follows.

Theorem 3.2. *Given any problem instance $g_{1:T} \in \mathcal{G}$, denote $\tilde{x} = [\tilde{x}_1, \cdots, \tilde{x}_T]$ and $x^* = [x_1^*, \cdots, x_T^*]$ as the actions produced by the ML model and the offline optimal policy, respectively, where we suppress the dependency on $g_{1:T}$ for notational convenience. Suppose that the cost of the offline optimal policy is given by $cost(\pi^*, g_{1:T})$ For any linear combination hyperparameter $\gamma \in [0,1]$, if* LADO-Lin *satisfies the λ-competitiveness constraint in Definition 2.5, we must have*

$$\frac{\|\tilde{x} - x^*\|^2}{cost(\pi^*, g_{1:T})} \le \frac{2}{\beta}\left(\frac{1-\gamma}{\gamma}\sqrt{\rho_{\pi^\dagger} - 1} + \frac{1}{\gamma}\sqrt{(1+\lambda)\rho_{\pi^\dagger} - 1}\right)^2 \tag{8}$$

where $\beta > 0$ is the strong convex parameter of the node cost functions in Assumption 2.1, $\lambda > 0$ is the competitiveness constraint parameter, and $\rho_{\pi^\dagger} = \max_{g_{1:T}\in\mathcal{G}} \frac{cost(\pi^\dagger, g_{1:T})}{cost(\pi^*, g_{1:T})} > 1$ is the competitive ratio of the expert policy $\pi^\dagger$.

Theorem 3.2 is proved in Appendix D.2 and provides a necessary condition for LADO-Lin to satisfy the λ-competitiveness constraint with respect to the expert policy. The metric $\frac{\|\tilde{x}-x^*\|^2}{cost(\pi^*, g_{1:T})}$ in (8) measures the distance between the ML policy and the offline optimal policy (normalized by the optimal cost) and is also commonly used by prior studies [35, 56] to characterize the ML prediction

quality. Intuitively, as $\lambda > 0$ increases, the competitiveness constraint becomes more relaxed, and so does the requirement on the ML prediction quality. Additionally, as LADO-Lin relies less on the ML policy (i.e., $\gamma \in [0, 1]$ becomes smaller) and/or the expert policy itself has a higher competitive ratio $\rho_{\pi^\dagger}$, there is a less stringent requirement on $\frac{\|\tilde{x}-x^*\|^2}{cost(\pi^*, g_{1:T})}$ for λ-competitiveness with respect to the expert.

Importantly, Theorem 3.2 highlights that, unless we completely ignore the ML advice (i.e., setting $\gamma = 0$), the discrepancy between the ML policy and the optimal policy measured in terms of $\frac{\|\tilde{x}-x^*\|^2}{cost(\pi^*, g_{1:T})}$ must be upper bounded by (8) for λ-competitiveness given any problem instance $g_{1:T} \in \mathcal{G}$. The larger γ, the greater dependency on ML predictions to improve the average performance, but the more difficult to meet the worst-case λ-competitiveness constraint.

In practice, it is extremely challenging, if not impossible, to ensure that the ML predictions satisfy (8) for *any* problem instance. It is well-known that, although a trained ML model can perform well on average, its performance in certain (possibly rare) cases can have an arbitrarily bad quality, especially when the testing problem instance is very distinct from those training instances. This is also the key motivation for safeguarding ML predictions to guarantee the worst-case competitiveness.

The performance of a learning-augmented algorithm is also analyzed under two extreme cases when the ML policy is arbitrarily bad and when it is perfect (i.e., robustness-consistency analysis [53, 57]). Next, we show the robustness and consistency of LADO-Lin.

Definition 3.3 (Robustness-consistency)**.** *Suppose that the competitive ratios of the ML policy $\tilde{\pi}$ and a learning-augmented online policy π are $\rho_{\tilde{\pi}} = \max_{g_{1:T} \in \mathcal{G}} \frac{cost(\tilde{\pi}, g_{1:T})}{cost(\pi^*, g_{1:T})}$ and $\rho_\pi = \max_{g_{1:T} \in \mathcal{G}} \frac{cost(\pi, g_{1:T})}{cost(\pi^*, g_{1:T})}$, respectively, where π^* is the optimal offline policy. Then, ρ_π is called the robustness of the policy π when $\rho_{\tilde{\pi}} \to \infty$, and the consistency when $\rho_{\tilde{\pi}} = 1$.*

Corollary 3.4 (Robustness-consistency of LADO-Lin)**.** *When $\rho_{\tilde{\pi}} \to \infty$, the robustness of* LADO-Lin *is $\rho_{\text{LADO-Lin}} = \infty$ for $\gamma \in (0, 1]$ and $\rho_{\text{LADO-Lin}} = \rho_{\pi^\dagger}$ for $\gamma = 0$; when $\rho_{\tilde{\pi}} = 1$, the consistency of* LADO-Lin *is upper bounded by $\rho_{\text{LADO-Lin}} = \gamma + (1-\gamma)\rho_{\pi^\dagger}$ where $\rho_{\pi^\dagger} = \max_{g_{1:T} \in \mathcal{G}} \frac{cost(\pi^\dagger, g_{1:T})}{cost(\pi^*, g_{1:T})} > 1$ is the competitive ratio of the expert policy $\pi^\dagger$.*

Corollary 3.4 shows that while LADO-Lin can improve the competitive ratio over the (best) expert policy $\pi^\dagger$ for $\gamma \in (0, 1]$, it has an unbounded robustness when the ML policy has an arbitrarily high cost. This shows the tension between following ML predictions for improving the average cost performance and staying close to the expert policy for worst-case robustness. Thus, both Theorem 3.2 and Corollary 3.4 highlight the key limitation of LADO-Lin, i.e., lack of worst-case performance guarantees.

4 LADO: Adaptively Combining ML Advice and Expert Advice

The previous section highlights that LADO-Lin with a fixed linear combination of the ML prediction and expert advice *cannot* offer guaranteed competitiveness or robustness in the worst case when ML predictions are of arbitrarily low quality. To address this limitation, this section proposes an *adaptive* approach based on a novel spatial cost decomposition and temporal reservation cost. Specifically, we present learning-augmented decentralized online optimization (LADO), an algorithm that adaptively exploits the benefits of ML while guaranteeing λ-competitiveness against any given expert policy $\pi^\dagger$ in a network.

4.1 Algorithm Design

We present our learning-augmented decentralized online algorithm, LADO, in Algorithm 1, where an ML policy is trained offline and deployed online by each agent v as in LADO-Lin.

To address the limitation of LADO-Lin and guarantee λ-competitiveness to the expert, the crux of LADO is to carefully leverage ML predictions while being close enough to expert actions. Specifically, we design a novel robust action set that addresses the key challenge that only local online information I_t^v is available to each agent v in our decentralized setting. By choosing an action that falls into the robust action set while staying close to the ML prediction, LADO guarantees λ-competitiveness and exploits the benefits of ML predictions, achieving the best of both worlds.

Concretely, we project the ML prediction $\tilde{x}_t^v$ into the robust action set denoted by $\mathcal{X}_{\lambda,t}^v$ as follows

$$x_t^v = \arg\min_{x \in \mathcal{X}_{\lambda,t}^v} \|x - \tilde{x}_t^v\|^2, \tag{9}$$

where the robust action set $\mathcal{X}_{\lambda,t}^v$ is convex and will be specified in Section 4.2. Thus, the projection in (9) can be efficiently performed by solving convex optimization at each individual agent v.

In contrast with the fixed linear combination of the ML prediction and expert advice in LADO-Lin that is provably insufficient for competitiveness guarantees, the novel robust action set we design for LADO is adaptively chosen based on the online costs of actual actions and the expert policy, guaranteeing λ-competitiveness for any $\lambda > 0$.

4.2 Designing a Robust Action Set

The core of LADO is an action set that "robustifies" ML predictions for λ-competitiveness. This is challenging due to the temporal and spatial information inefficiency — the λ-competitiveness requirement in (1) is imposed over the total global cost over T steps, whereas each agent must choose its action based on *local* and *online* information I_t^v.

To construct a robust action set $\mathcal{X}_{\lambda,t}^v$ locally computable by each agent, we first convert the λ-competitiveness constraint over T time steps to an equivalent *anytime* constraint below.

Proposition 4.1 (Anytime λ-competitiveness). *For any $\lambda > 0$, to guarantee the λ-competitiveness constraint $cost(\pi, g_{1:t}) \le (1+\lambda) cost(\pi^\dagger, g_{1:t})$, $\forall g_{1:t} \in \mathcal{G}$, a sufficient and necessary condition is*

$$cost(\pi, g_{1:t}) \le (1+\lambda) cost(\pi^\dagger, g_{1:t}), \quad \forall t \in [1, T], \tag{10}$$

where $cost(\pi, g_{1:t})$ is the cumulative global cost of a policy π up to time $t \in [1, T]$.

Proof. The sufficient part in Proposition 4.1 is straightforward, while the necessary part can be proved by constructing a counter-example as follows. Suppose that there is a time $t \in [1, T-1]$ such that $cost(\pi, g_{1:t}) \ge (1+\lambda) cost(\pi^\dagger, g_{1:t}) + \epsilon$, where $\epsilon > 0$. It is possible that the expert's future total cost $cost(\pi^\dagger, g_{t+1:T}) < \frac{\epsilon}{1+\lambda}$. Then, by the non-negativeness of the cost functions, the policy π's total cost $cost(\pi, g_{1:T}) \ge cost(\pi, g_{1:t}) \ge (1+\lambda) cost(\pi^\dagger, g_{1:t}) + \epsilon > (1+\lambda) \cdot \left[cost(\pi^\dagger, g_{1:t}) + cost(\pi^\dagger, g_{t+1:T})\right] = (1+\lambda) cost(\pi^\dagger, g_{1:T})$, violating λ-competitiveness. □

While Proposition 4.1 simplifies the λ-competitiveness constraint, the spatial cost in (10) cannot be locally computed by each agent in a decentralized manner without knowing its neighboring agents' actions. Moreover, due to the future uncertainties and coupling of actions in online optimization, it is very challenging to meet the constraints (10) for every $t \in [1, T]$. To address these challenges, we propose novel *adaptive spatial cost decomposition* and introduce *reservation costs* to safeguard online actions for λ-competitiveness.

$$\begin{aligned}&\sum_{\tau=1}^{t} f_\tau^v(x_\tau^v) + \sum_{\tau=1}^{t} c_\tau^v(x_\tau^v, x_{\tau-1}^v) + \sum_{\tau=1}^{t-1} \sum_{(v,u)\in\mathcal{E}} \kappa_\tau^{(v,u)} \cdot s_\tau^{(v,u)}(x_\tau^v, x_\tau^u) + R(x_t^v, x_t^{v,\dagger}) \\ \le &(1+\lambda)\Big(\sum_{\tau=1}^{t} f_\tau^v(x_\tau^{v,\dagger}) + \sum_{\tau=1}^{t} c_\tau^v(x_\tau^{v,\dagger}, x_{\tau-1}^{v,\dagger}) + \sum_{\tau=1}^{t-1} \sum_{(v,u)\in\mathcal{E}} \kappa_\tau^{(v,u)} \cdot s_\tau^{(v,u)}(x_\tau^{v,\dagger}, x_\tau^{u,\dagger})\Big)\end{aligned} \tag{13}$$

4.2.1 Spatial Cost Decomposition. Due to the decentralized setting, we first decompose the global cost $g_t(x_t)$ at time t into locally computable costs for individual agents $v \in \mathcal{V}$ expressed as

$$g_t^v(x_t^v) = f_t^v(x_t^v) + c_t^v(x_t^v, x_{t-1}^v) + \sum_{(u,v)\in\mathcal{E}} \kappa_t^{(v,u)} s_t^{(u,v)}(x_t^v, x_t^u), \tag{11}$$

where we use the weights $\kappa_t^{(v,u)} \ge 0$ and $\kappa_t^{(v,u)} \ge 0$, such that $\kappa_t^{(v,u)} + \kappa_t^{(u,v)} = 1$ for $(v,u) \in \mathcal{E}$, to adaptively split the shared spatial cost $s_t^{(u,v)}(x_t^v, x_t^u)$ between the two connected agents (i.e., $\kappa^{(u,v)}$ for agent v and $\kappa^{(v,u)}$ for agent u). We specify the choice of the weight $\kappa_t^{(v,u)}$ in (14) later.

Based on the cost decomposition in (11), the anytime λ-competitiveness constraint in (10) can be guaranteed if the action of each node v satisfies the following local constraint:

$$\sum_{i=1}^{t} g_i^v(x_i^v) \le (1+\lambda) \sum_{i=1}^{t} g_i^v(x_i^{v,\dagger}), \quad \forall t \in [1, T]. \tag{12}$$

At step t, however, agent v cannot evaluate its local cost $g_t^v(x_t^v)$, because it has no access to the actions x_t^u and expert actions $x_t^{u,\dagger}$ of its connected neighbors u and hence cannot calculate the actual or expert's spatial costs for $(v,u) \in \mathcal{E}$.

Additionally, even if agent v has the knowledge of $g_t^v(x_t^v)$, simply satisfying (12) at time t cannot guarantee that a feasible action exists to satisfy the local constraints for future steps $t+1, \cdots, T$ due to the temporal cost. To see this, consider a toy example with $T = 2$ and $c_t^v = \|x_t^v - x_{t-1}^v\|^2$. Assume that x_1^v is selected such that the first-step local constraint is satisfied by equality, i.e., $g_1^v(x_1^v) = (1+\lambda) g_i^v(x_1^{v,\dagger})$. Then, at the second step $t = 2$, it can happen that the node costs satisfy $f_2^v(x_1^{v,\dagger}) = 0$ and $f_2^v(x_1^v) > 0$, while the spatial costs are all zero. Then, with the expert action $x_2^{v,\dagger} = x_1^{v,\dagger}$, it follows that $g_2^v(x_2^v) > (1+\lambda) g_2^v(x_2^{v,\dagger}) = 0$ for any $x_2^v \in \mathcal{X}$, thus violating the local constraint (12) for agent v. By the same reasoning, the λ-competitiveness constraint can be violated for the whole network.

4.2.2 Robust Action Sets via Reservation Costs. To ensure non-empty sets of feasible actions satisfying the local constraints (12) for each time step t, we propose a *reservation cost* that safeguards each agent v's action against any possible uncertainties (e.g., connected agent u's current actions and future cost functions). Compared to a centralized setting [30, 35], designing a proper reservation cost in a decentralized creates substantial challenges, as it needs to hedge against both *spatial* and future *temporal* uncertainties.

With only local online information I_t^v available to agent v, the key insight of our added reservation cost at each time step t is to bound the maximum possible cost difference between agent v's cost $\sum_{i=1}^t g_i^v(x_i^v)$ and its corresponding cost constraint $(1+\lambda)\sum_{i=1}^t g_i^v(x_i^{v,\dagger})$ for future time steps. More concretely, we use a new constraint in (13) to define the robust action set for agent v at step t. In constraint (13), the weight $\kappa_\tau^{(v,u)}$ (attributed to agent v) for adaptively splitting the spatial cost $s_\tau^{(v,u)}$

between agent v and agent u for $\tau = 1, \cdots, t-1$ is

$$\kappa_\tau^{(v,u)} = \frac{\|x_\tau^v - x_\tau^{v,\dagger}\|^2}{\|x_\tau^v - x_\tau^{v,\dagger}\|^2 + \|x_\tau^u - x_\tau^{u,\dagger}\|^2}. \tag{14}$$

Additionally, the reservation cost[1] is

$$R(x_t^v, x_t^{v,\dagger}) = \frac{\ell_T + \ell_S \cdot D_v}{2}(1 + \frac{1}{\lambda_0})\|x_t^v - x_t^{v,\dagger}\|^2, \tag{15}$$

where ℓ_T and ℓ_S are smoothness parameters for the temporal and spatial cost functions, D_v is the degree of agent v (i.e., the number of agents connected to agent v), and $0 < \lambda_0 \leq \lambda$ is a hyperparameter to adjust the size of the robust action set (and will be optimally chosen as $\lambda_0 = \sqrt{1+\lambda} - 1$ in Theorems 5.2 and B.1). In the special case when both $x_t^v = x_t^{v,\dagger}$ and $x_t^u = x_t^{u,\dagger}$, we set $\kappa_\tau^{(v,u)} = \frac{1}{2}$ in (14).

Importantly, the new constraint (13) for agent v can be calculated purely based on local online information I_t^v; it only depends on the cumulative node and temporal costs up to time t, as well as the spatial costs (including the feedback of the connected neighboring agents' actions and their expert actions) up to time $t-1$. Thus, the overall cost to share information between two connected agents in the network is small. Moreover, the reservation cost $R(x_t^v, x_t^{v,\dagger})$ safeguards agent v's action not only against uncertainties in future temporal cost functions in online optimization, but also against delayed spatial costs resulting from decentralized optimization, which we further explain as follows.

• **Temporal uncertainties**. The temporal cost couples each agent's actions over time, but the online action needs to be chosen without knowing all the future costs. Consequently, as shown in the example in Section 4.2.1, simply satisfying the λ-competitiveness in terms of the cumulative cost up to t does not necessarily ensure λ-competitiveness in the future. To hedge against temporal uncertainties, our reservation cost $R(x_t^v, x_t^{v,\dagger})$ in (15) includes the term $\frac{\ell_T}{2}(1 + \frac{1}{\lambda_0})\|x_t^v - x_t^{v,\dagger}\|^2$, which bounds the maximum cost disadvantage for agent v: $c_t^v(x_t^v, x_{t+1}^v) - (1+\lambda)c_t^v(x_t^{v,\dagger}, x_{t+1}^{v,\dagger}) \leq \frac{\ell_T}{2}(1 + \frac{1}{\lambda_0})\|x_t^v - x_t^{v,\dagger}\|^2$. Thus, $x_{t+1}^v = x_{t+1}^{v,\dagger}$ is always a feasible robust action for agent v at time $t+1$.

• **Spatial uncertainties**. In our decentralized setting, agent v chooses its action based on the local online information I_t^v, which creates spatial uncertainties regarding its connected neighboring agents' actions and spatial costs. In our design, with the splitting weight $\kappa_{t-1}^{(v,u)}$ in (14) and the term $\frac{\ell_S \cdot D_v}{2}(1 + \frac{1}{\lambda_0})\|x_{t-1}^v - x_{t-1}^{v,\dagger}\|^2$ included in the reservation cost in (15) at time $t-1$, we ensure that our constraint in (13), if satisfied, can always guarantee the local constraint in (12) and hence also the λ-competitiveness constraint, due to the following inequality:

$$\begin{aligned}
&\sum_{(v,u)\in\mathcal{E}} \kappa_{t-1}^{(v,u)} \left(s_{t-1}^{(v,u)}(x_{t-1}^v, x_{t-1}^u) - (1+\lambda) s_{t-1}^{(v,u)}(x_{t-1}^{v,\dagger}, x_{t-1}^{u,\dagger}) \right) \\
\leq &\sum_{(v,u)\in\mathcal{E}} \kappa_{t-1}^{(v,u)} \frac{\ell_S}{2}(1 + \frac{1}{\lambda_0}) \left(\|x_{t-1}^v - x_{t-1}^{v,\dagger}\|^2 + \|x_{t-1}^u - x_{t-1}^{u,\dagger}\|^2 \right) = \sum_{v\in\mathcal{V}} \frac{\ell_S \cdot D_v}{2}(1 + \frac{1}{\lambda_0})\|x_{t-1}^v - x_{t-1}^{v,\dagger}\|^2.
\end{aligned} \tag{16}$$

Note that, as the degree D_v of node v increases, more agents are connected to agent v and hence spatial uncertainties also naturally increase, resulting in an increased reservation cost in (15).

[1]Despite the one-step delayed feedback of the neighbors' actions, knowing the spatial cost function $s_t^{v,u}(\cdot,\cdot)$ at the beginning of time t is still helpful. For example, the reservation cost for spatial cost uncertainties can be reduced if the smoothness constant of $s_t^{v,u}(\cdot,\cdot)$ is smaller than ℓ_s.

In summary, our novel robust action set for agent v at time step t is designed as

$$\mathcal{X}^v_{\lambda,t} = \{x^v_t \mid x^v_t \text{ satisfies (13) for step } t\}, \tag{17}$$

which, by convexity of cost functions, is convex and leads to computationally-efficient projection (9). For example, in our experiments, it takes about 1 second to run inference for 1000+ instances on a laptop.

5 Performance Bounds for LADO

We now analyze LADO in terms of its competitiveness, average cost, and robustness-consistency, proving that LADO is λ-competitive against any given expert and simultaneously achieves finite consistency.

5.1 λ-Competitiveness

We state λ-competitiveness of LADO as follows. The proof is provided in Appendix E.

Theorem 5.1. *(λ-competitiveness of* LADO*) Given any ML policy $\tilde{\pi}$ and expert policy $\pi^\dagger$, for any $\lambda > 0$ and $\lambda_0 \in (0, \lambda]$ in the robust action set in (17), the cost of* LADO *satisfies* $cost(\text{LADO}, g_{1:T}) \leq (1+\lambda) \cdot cost(\pi^\dagger, g_{1:T})$ *for any problem instance $g_{1:T} \in \mathcal{G}$.*

Theorem 5.1 guarantees that, for any problem instance $g_{1:T} \in \mathcal{G}$, the total global cost of LADO is always upper bounded by $(1+\lambda)$ times the global cost of the expert policy $\pi^\dagger$, regardless of the quality of ML predictions. This competitiveness guarantee is the first in the context of decentralized learning-augmented algorithms and attributed to our novel design of locally computable robust action sets in (17), based on which each agent individually safeguards its own online actions. Moreover, for our setting, there exist online policies (e.g., localized policy for multi-agent networks [36]) that have bounded competitive ratios against the offline oracle and hence can be readily applied as expert policies in LADO. Thus, their competitive ratios immediately translate with a scaling factor of $(1+\lambda)$ into competitiveness of LADO against the offline oracle.

5.2 Average Cost

A key goal of utilizing an ML policy is to improve the *average* performance over the expert policy. Thus, we first consider the average performance of LADO under a general ML policy. We rewrite LADO as $\text{LADO}(\tilde{\pi})$ to highlight its dependency on $\tilde{\pi}$ when applicable. The results are shown in Theorem 5.2, whose proof relies on the spatial cost decomposition developed in Section 4.2.1 and is deferred to Appendix F.1.

Theorem 5.2. *(Average Cost of* $\text{LADO}(\tilde{\pi})$*) Given an expert policy $\pi^\dagger$ and any ML policy $\tilde{\pi}$, for the context distribution $\mathcal{P}_{g_{1:T}}$, we define $AVG(\pi^\dagger)$, $AVG(\tilde{\pi})$ as the average costs of the expert policy and ML policy. For any $\lambda > 0$, by optimally setting $\lambda_0 = \sqrt{1+\lambda} - 1$, the average cost of* $\text{LADO}(\tilde{\pi})$ *is upper bounded by*

$$AVG(\text{LADO}(\tilde{\pi})) \leq \min\left\{(1+\lambda)AVG(\pi^\dagger), \left(\sqrt{AVG(\tilde{\pi})} + \sqrt{\sum_{v\in\mathcal{V}} \omega_v(\lambda, \tilde{\pi}, \pi^\dagger)}\right)^2\right\},$$

where $\omega_v(\lambda, \tilde{\pi}, \pi^\dagger) = \mathbb{E}_{g_{1:T}}\left\{\sum_{t=1}^T \left[\frac{\ell_f + 2\cdot\ell_T + \ell_S \cdot D_v}{2}\|\tilde{x}^v_t - x^{v,\dagger}_t\|^2 - (\sqrt{1+\lambda}-1)^2 \cdot cost^\dagger_{v,t}\right]^+\right\}$*, in which D_v is the degree of node v and $cost^\dagger_{v,t}$ denotes the sum of hitting cost and switching cost for the expert $\pi^\dagger$.*

Theorem 5.2 quantifies the tradeoff between exploiting the ML policy for average cost performance and following the expert policy for worst-case competitiveness in a decentralized setting.

Specifically, the average cost bound of $\texttt{LADO}(\tilde{\pi})$ is a minimum of two terms. The first term holds due to the guaranteed λ-competitiveness against the expert policy. The second term shows that, due to the competitiveness requirement, $\texttt{LADO}(\tilde{\pi})$ can deviate from the ML policy and hence have a higher average cost than $AVG(\tilde{\pi})$. The cost difference is primarily driven by the sum of $\omega_v(\lambda, \tilde{\pi}, \pi^\dagger)$ for all nodes in the set $\mathcal{V}$, which measures how well LADO follows the ML policy. For each node v, $\omega_v(\lambda, \tilde{\pi}, \pi^\dagger)$ is upper bounded by the expected distance between actions made by LADO and actions made by the pure ML policy. Naturally, as we impose a less stringent competitiveness constraint (i.e., smaller $\lambda > 0$) or the expert policy $\pi^\dagger$ and the ML policy $\tilde{\pi}$ are better aligned (i.e., smaller action distance $\|\tilde{x}_t^v - x_t^{v,\dagger}\|$), we can better exploit the power of the ML policy $\tilde{\pi}$ with a reduced $\omega_v(\lambda, \tilde{\pi}, \pi^\dagger)$. Another insight is that $\omega_v(\lambda, \tilde{\pi}, \pi^\dagger)$ decreases when the expert policy has a higher cost, which naturally provides more freedom to LADO to follow ML while still being able to satisfy the λ-competitiveness requirement.

Impact of network topologies. Given the same set of nodes but different numbers of node connections, the spatial costs can be significantly different. This impact is also captured by the term $\omega_v(\lambda, \tilde{\pi}, \pi^\dagger) = \mathbb{E}_{g_{1:T}}\left\{\sum_{t=1}^T \left[\frac{\ell_f+2\cdot\ell_T+\ell_S\cdot D_v}{2}\|\tilde{x}_t^v - x_t^{v,\dagger}\|^2 - (\sqrt{1+\lambda}-1)^2\cdot \text{cost}_{v,t}^\dagger\right]^+\right\}$, where D_v is the degree of agent v. Specifically, when agent v is connected with more nodes (i.e., greater D_v) while the other factors are held constant, the spatial costs and uncertainties also increase accordingly. The competitiveness guarantees compel agent v to more conservatively follow the expert policy and potentially deviate more from the ML policy. In other words, the cost gap bound between LADO and the ML policy $\omega_v(\lambda, \tilde{\pi}, \pi^\dagger)$ increases with the increased node degree. Thus, in general, when the graph density increases with more node connections, the total cost bound compared to the ML policy also increases because of more spatial cost uncertainties and hence potentially more perturbations added to the ML advice.

Interestingly, even given the same number of node connections (i.e., edges) and the same number of nodes, how the nodes are connected (e.g., linear chain vs. star graphs in Fig. 3) can play a role in the cost. For example, when every node has a small degree in a linear chain graph and the competitiveness constraint $\lambda \geq 0$ is not too small, $\omega_v(\lambda, \tilde{\pi}, \pi^\dagger)$ is generally smaller due to the ReLu operation, making it easier to follow the ML policy in LADO; on the other hand, when a node has a very high degree (in a star graph), the term $\omega_v(\lambda, \tilde{\pi}, \pi^\dagger)$ for the high-degree node is likely to be positive (unless λ is sufficiently large), i.e., this node's action likely deviates significantly from its ML policy. Consequently, when the other factors are held constant, LADO can more effectively adhere to the ML policy and achieve better average performance in a linear chain graph compared to a star graph, despite the identical number of spatial connections in both graghs. This phenomenon is empirically observed and discussed in our experiments, as illustrated in Figure 1.

To further highlight the impact of node connections, we extend the design and cost analysis of LADO from an undirected graph to a directed graph. The results are available in Appendix A.

5.3 Robustness and Consistency

We now show the robustness and consistency (Definition 3.3) for LADO as follows.

Theorem 5.3 (Robustness-consistency of LADO). *Define $\rho_{\tilde{\pi}}$, $\rho_{\pi^\dagger}$ and $\rho_{\texttt{LADO}}$ as the competitive ratios of the ML policy $\tilde{\pi}$, expert policy $\pi^\dagger$ and* LADO *against the offline optimal policy π^*, and $\ell = \frac{\ell_f+2\ell_T+\ell_S D_{\max}}{2}$ as the gradient Lipschitz constant of the global cost function, respectively. When $\rho_{\tilde{\pi}} \to \infty$, the robustness of* LADO *is upper bounded by $\rho_{\texttt{LADO}} \leq (1+\lambda)\rho_{\pi^\dagger}$ for any $\lambda > 0$; when $\rho_{\tilde{\pi}} = 1$, the consistency of* LADO

is upper bounded by

$$\rho_{\texttt{LADO}} \le \min\left\{\left[1+\sqrt{\max_{g_{1:T}\in\mathcal{G}}\frac{\sum_{v\in\mathcal{V}}\sum_{t=1}^{T}\left[\ell\|x_t^{v,*}-x_t^{v,\dagger}\|^2-(\sqrt{1+\lambda}-1)^2\cdot cost_{v,t}^{\dagger}\right]^+}{cost(x_{1:T}^*,g_{1:T})}}\right]^2, (1+\lambda)\rho_{\pi^\dagger}\right\}$$
$$\le \min\left\{\left(1+2\sqrt{\frac{\ell}{\beta}\cdot(\rho_{\pi^\dagger}-1)}\right)^2, (1+\lambda)\rho_{\pi^\dagger}\right\}. \tag{18}$$

where $\beta>0$ is the strong convex parameter (Assumption 2.1) and $cost_{v,t}^{\dagger}$ is the expert's cost of node $v\in\mathcal{V}$ at time $t\in[1,T]$.

Theorem 5.3 is proved in Appendix F.3 and highlights that LADO can achieve both finite robustness and consistency simultaneously. In contrast with LADO-Lin which fails to provide finite robustness, LADO prioritizes the worst-case competitiveness guarantees as a constraint parameterized by $\lambda>0$. Nonetheless, in general cases, LADO does not have a better consistency than LADO-Lin or the best expert policy when the ML predictions are perfect. While it remains an open problem to achieve the optimal robustness-consistency tradeoff (except for a few specific problems) in the learning-augmented literature [57], we note that our result is consistent with the fundamental impossibility in our problem setting. Specifically, even in the special single-agent case for our problem setting, the prior studies [35] have shown that it is *impossible* to achieve finite robustness while still having a consistency better than the best expert's competitive ratio $\rho_{\pi^\dagger}$ without further assumptions. As a result, LADO-Lin achieves a better consistency than best expert's competitive ratio (Theorem 3.4) and hence cannot guarantee finite robustness; LADO guarantees finite robustness (Theorem 5.1) and hence cannot offer a better consistency in general cases. Nonetheless, by making an additional assumption that the expert's $cost_{v,t}^{\dagger}$ of each node $v\in\mathcal{V}$ is always strictly positive at time $t\in[1,T]$, we see from the first inequality in (18) that LADO can achieve a lower consistency by increasing $\lambda>0$ and even simultaneously 1-consistency and finite robustness when $\lambda>0$ is sufficiently large (which pushes the term $\left[\ell\|x_t^{v,*}-x_t^{v,\dagger}\|^2-(\sqrt{1+\lambda}-1)^2\cdot cost_{v,t}^{\dagger}\right]^+\to 0$ in (18)).

Importantly, the robustness and consistency analysis is still for the *worst* case. By utilizing a well-trained ML model in LADO-Lin and LADO, we can still improve the average cost performance compared to the pure expert policy, highlighting the key advantage of ML predictions. This is discussed in Theorem 3.1 and Theorem 5.2, and also empirically demonstrated in Section 6.3.

Theorem 5.2 applies to any ML models, including ML models that are trained as standalone optimizers without considering the design of LADO and hence may have training-testing objective mismatches. To further improve the average cost performance, we consider a projection-aware ML policy $\tilde{\pi}_\lambda^\circ$ that is optimally trained to minimize the actual cost with explicit consideration of the downstream projection in LADO. Our performance analysis formally demonstrates the benefits of using $\tilde{\pi}_\lambda^\circ$ in LADO for average cost reduction and is available in Appendix B.

6 Case Study: Decentralized Battery Management for Sustainable Computing

To demonstrate the empirical benefits of LADO-Lin and LADO, we carry out experiments with the application of decentralized battery management for sustainable computing, as introduced in Section 2.2. Our results show that with learning augmentation, both LADO-Lin and LADO can empirically achieve a good average cost performance. Meanwhile, compared to LADO-Lin that lacks guaranteed competitiveness, LADO is less sensitive to the potentially low quality of ML predictions.

6.1 Experimental Setup

The recent surge in computing demands, such as large AI models for language services, has placed an urgent emphasis on decarbonizing data centers for sustainability. To highlight the potential of LADO for managing decentralized batteries in the context of sustainable data centers, we use a trace-based simulation in our experiments following the common practice in the literature [37, 58]. The data center workload trace is taken from Microsoft Azure [59], which contains the CPU utilization of 2,695,548 virtual machines (VM) for each 5-minute window. We estimate the energy consumption $P_{d,t}$ by summing up the CPU utilization of all VMs.

The weather-related parameters, i.e., wind speed, solar radiation and temperature data, are all collected from the National Solar Radiation Database [60]. Based on the weather information, we use empirical equations to model the wind and solar renewables generated at time step t. Specifically, the amount of solar energy generated at step t is given based on [61] by $P_{\text{solar},t} = \frac{1}{2}\kappa_{\text{solar}} A_{\text{array}} I_{\text{rad},t}(1 - 0.05 * (\text{Temp}_t - 25))$, where A_{array} is the solar array area (m^2), $I_{\text{rad},t}$ is the solar radiation (kW/m^2), and Temp_t is the temperature (°C) at time t, and κ_{solar} is the conversion efficiency (%) of the solar panel. The amount of wind energy is modeled based on [62] as $P_{\text{wind},t} = \frac{1}{2}\kappa_{\text{wind}} \varrho A_{\text{swept}} V^3_{\text{wind},t}$, where ϱ is the air density (kg/m^3), A_{swept} is the swept area of the turbine (m^2), κ_{wind} is the conversion efficiency (%) of wind energy, and $V_{\text{wind},t}$ is the wind speed (kW/m^2) at time t. Thus, at time t, the total energy generated by the solar and wind renewables is $P_{\text{r},t} = P_{\text{wind},t} + P_{\text{solar},t}$. By subtracting the renewables $P_{r,t}$ from the data center's energy demand $P_{d,t}$, we obtain the net demand as $P_{n,t} = P_{d,t} - P_{r,t}$, which is then normalized to $[-1, 1]$.

In our case study, we evaluate the performance of LADO and LADO-Lin on a diverse set of experimental settings, including heterogeneous graph nodes with various graph topologies. To represent the range of battery health, we assign different self-degradation coefficients A_v to these battery units. Beyond self-degradation coefficients, we further consider heterogeneity in the storage capacities and rated output powers from real-world energy storage systems. We begin with a fully connected graph of 3 battery units, then expand our experiments to 15-node graphs, exploring representative topologies (e.g., star, linear) and a variety of randomly generated graphs with varying densities. To assess the scalability of our algorithm, we generate fully-connected graphs with up to 120 nodes. More details on the extended experiments can be found in Appendix C.

6.2 Baselines

We compare LADO-Lin and LADO with the following baselines. In addition, we also evaluate LADO-OPT that uses the optimal projection-aware ML policy (Appendix B). These representative baselines are closely related to our problem and range from the simplest Greedy to the most powerful oracle OPT.

- **Offline optimal** (OPT): OPT obtains the offline optimal solution to (3) with the complete information for each problem instance.
- **Expert**: The state-of-the-art online algorithm for our problem is the localized prediction policy [36]. Here, we set the prediction window as 1 and refer to it as Expert.
- **ML optimizer** (ML): ML uses the same recursive neural network (RNN) model used by LADO, but is trained as a standalone policy without considering LADO.
- **Hitting cost optimizer** (HitOnly): HitOnly solely optimizes the node cost for each node, which aims at tracking the nominal SoC value exactly.
- **Single-step cost optimizer** (Greedy): Greedy myopically minimizes the node cost and temporal cost at each time for each node.

6.3 Results for Networks with 3 Nodes

Considering a simple 3-node network, we show the empirical average (AVG) and competitive ratio (CR) values in Table 1, where the best empirical AVG and CR values are marked in bold font. The CR values are the empirically worst cost ratio of an algorithm's cost to OPT's cost in our testing dataset. We see that LADO-Lin with $\gamma = 0.1$ achieves the best empirical CR, but unlike LADO, the empirical advantage does not have any theoretical guarantees. This is also partly because the empirical CR value can be affected by a single bad problem instance and hence is more volatile. For example, when we increase the reliance of LADO-Lin on the ML policy by increasing $\gamma \in [0, 1]$, the empirical of CR achieved by LADO-Lin also increases quickly and becomes higher than that of Expert and LADO-OPT. Although the pure ML-based optimizer achieves better average cost by leveraging historical data, its CR is significantly higher than Expert and even higher than Greedy.

	Expert	**ML**	**HitOnly**	**Greedy**	**LADO**			
					$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$
AVG	134.61	108.09	139.07	200.61	115.17	109.59	108.11	107.92
CR	1.738	5.294	8.993	3.264	1.843	2.174	2.535	3.617
	LADO-Lin				**LADO-OPT**			
	$\gamma = 0.1$	$\gamma = 0.3$	$\gamma = 0.5$	$\gamma = 0.9$	$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$
AVG	127.25	115.68	108.29	106.04	113.76	106.96	105.33	**105.06**
CR	**1.668**	1.914	2.506	4.587	1.812	1.972	2.451	3.032

Table 1. Default case for a network with 3 nodes. The average cost of OPT in the testing dataset is 88.10.

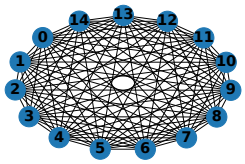
By projecting the ML actions into carefully designed robust action sets, LADO can significantly reduce the CR compared to ML, while improving the average cost performance compared to Expert. Our analysis in Theorem 5.2 proves that, with a larger λ, the average cost of LADO is closer to that of ML, while the guaranteed competitiveness becomes weaker.

Interestingly, in Table 1 by setting $\lambda = 2$, LADO can even achieve a lower average cost than ML, while still having a lower CR. The reason is that Expert performs much better than ML for some problem instances. Thus, the inclusion of Expert in LADO avoids those instances that would otherwise have a high cost if ML were used, and meanwhile a large $\lambda = 2$ also provides enough flexibility for LADO to exploit the benefits of ML in most other cases. Moreover, by training an ML policy that is explicitly aware of the projection, LADO-OPT can further reduce the average cost compared to LADO while having the same robustness guarantees. Additional results on different testing distributions are available in Appendix C.1.

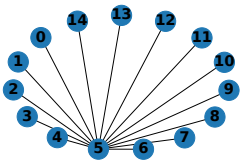
6.4 Results for Larger Networks

We now consider larger networks with more nodes to assess LADO. The setup is available in Appendix C.2.1. For three representative graph topologies (i.e., complete graph, star graph, and linear chain graph), the empirical average node, temporal, and spatial costs for each algorithm in a 15-node network are shown in Fig. 1.

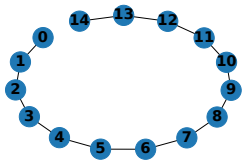
Notably, for these three graph topologies, the complete and star graphs have the same maximum node degree, while the star graph and linear chain graph share the same number of edges (or graph density). By comparing algorithm performance on these two pairs, we can gain insights into the impact of graph topologies. For example, the significantly lower graph density in a star graph than in a complete graph explains why all the algorithms considered exhibit lower total spatial costs on the star graph. This is consistent with our new theoretical analysis of the cost performance in Theorem 5.2. Compared to the complete graph, the spatial cost uncertainties are reduced due to fewer connections in the star graph. Thus, LADO can more effectively leverage the power of ML predictions, which leads to improved performance in both node cost and temporal cost of LADO.



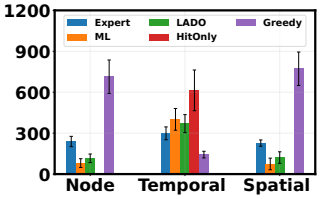
(a) Complete graph



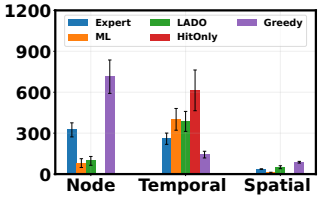
(b) Star graph



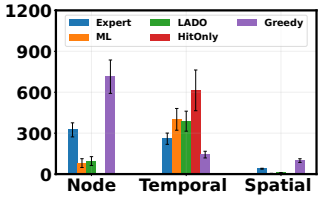
(c) Linear chain graph



(d) Individual Costs (complete graph)



(e) Individual Costs (star graph)

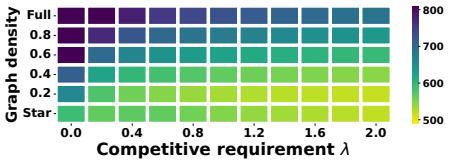


(f) Individual Costs (chain graph)

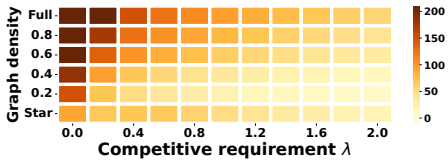
Fig. 1. The evaluation of LADO and baseline algorithms in terms of the node, temporal and spatial costs, with various graph topologies. By default, the competitiveness requirement λ is set to 1 in LADO for all the graphs.

In contrast to the linear chain graph, the star graph concentrates node degrees on a single node, resulting in distinct cost behaviors. As Theorem 5.2 indicates, the more uniform node degree distribution in the chain graph affords our algorithm greater flexibility to follow the ML policy by deviating more from the expert policy, ultimately reducing the cost increase term $\omega_v(\lambda, \tilde{\pi}, \pi^\dagger)$. Our empirical findings align with this theoretical analysis, demonstrating significantly lower overall costs for our algorithm in the linear chain graph due to its more effective exploitation of the ML policy. As presented in Fig. 1(f), the effect of reduced spatial costs is more pronounced for LADO in the linear chain graph.

Next, we evaluate LADO on graphs with the same number of nodes but varying random graph topologies. Starting from the star graph, we gradually add random edges between nodes to increase the graph density until the graph is fully connected. Additionally, we also consider different competitiveness requirements in these graph topologies.



(a) Total cost of LADO



(b) Regret of LADO compared to ML

Fig. 2. Impact of graph density and competitiveness requirement λ on the overall cost of LADO, along with the additional cost (regret) associated with the projection process compared to the ML policy.

As shown in Fig. 2, the total cost of LADO increases with graph density due to the additional spatial uncertainties introduced by denser connections. As Theorem 5.2 suggests, these increased spatial cost uncertainties lead to higher costs for LADO compared to the ML policy. To clarify this further, we compare the regret of LADO to the ML policy, defined as $\mathbb{E}_{g_{1:T}}\left[cost(\text{LADO}(\tilde{\pi})) - cost(\tilde{\pi})\right]$. This regret

quantifies the additional cost incurred due to the projection process required for competitiveness guarantees. As shown in Fig. 2(b), the regret of LADO typically increases with graph density, as denser graphs introduce more spatial cost uncertainties, making it more challenging to closely follow the ML policy. Similarly, with the same graph density (e.g. fully connected), the cost of LADO also increases as graph size increase from 3 to 120 nodes, shown as Fig. 5. This is because the growth of graph size naturally leads to more spatial connections, leading to elevated regret and overall cost, where detailed results can be found in Appendix C.2. Moreover, stricter competitiveness requirements necessitate a closer adherence to the expert, further hindering LADO's ability to follow the ML policy and resulting in higher regret.

7 Related Work

Smoothed online convex optimization. Smoothed online convex optimization in a centralized single-agent setting is a classic problem for which many algorithms have been designed to bound the worst-case performance, e.g., [12, 14–18]. Recently, a growing literature has begun to study online convex optimization in a decentralized networked system [10, 19, 36, 63]. Compared to the centralized setting, the decentralized setting is significantly more challenging, since an agent has no access to the information of other agents before making its action at each step. In this context, a recent work [36] proposes an online algorithm with a bounded competitive ratio and shows the dependency of the competitive ratio on cost predictions. Several other studies [10, 19–22] propose algorithms with bounded regrets. In all cases, these studies focus on the worst-case performance, which leads to conservative algorithms that may not achieve a low average global cost. To address this limitation, in this work we exploit the benefits of untrusted ML predictions to improve the average cost performance, while leveraging a robust policy to achieve the worst-case robustness.

ML-based optimizers. ML policies have been used for exploiting the statistical information and improve the average performance of various (online) optimization problems, including scheduling, resource management, and secretary problems [23–26]. There also exist ML-based optimizers, such as multi-agent learning [27–29], in the context of decentralized optimization where agents have limited or delayed communications. However, a crucial drawback of pure ML-based optimizers is that they may have very high or even unbounded costs in the worst case, making them unsuitable for mission-critical applications. We provide an approach to empowering such ML-based algorithms with worst-case robustness guarantees.

Learning-augmented algorithms. Learning-augmented algorithms have been proposed as a way to add worst-case robustness guarantees on top of ML policies in a variety of settings, e.g., [11, 30–32, 32–34]. To guarantee worst-case competitiveness, it is crucial to address the potential risks associated with following the ML predictions, which is also the key challenge for learning-augmented algorithm designs. More recently, learning-augmented algorithms have been designed for smoothed online optimization with switching costs [11, 56, 57, 64]. However, learning-augmented algorithm designs in decentralized settings remain largely unexplored and are more challenging due to limited information availability. Thus, our study addresses this gap by introducing a novel, worst-case guaranteed learning-augmented algorithm specifically designed for decentralized environments.

Our work differs from the standard constrained online optimization (e.g., [13]) in that LADO is a meta algorithm leveraging one robust policy to safeguard another policy which can potentially perform better on average. Additionally, besides our novel decentralized setting and algorithm design based on spatial cost decomposition, LADO considers worst-case robustness and hence substantially differs from conservative bandits/reinforcement learning that focus on average or high-probability performance constraints [65–67].

8 Concluding Remarks

This paper studies learning-augmented decentralized online convex optimization in networks. We begin with LADO-Lin by linearly combining the ML policy and the expert policy. It is proved that, while LADO-Lin can exploit the potential of ML to improve the average cost performance, it does not have guaranteed worst-case performance. Then, we propose LADO, a novel algorithm that improves the average performance while guaranteeing worst-case robustness. LADO addresses the key challenges of temporal and spatial information inefficiency and constructs novel robust action sets that allow agents to choose individual actions based on local online information. We prove bounds on the guaranteed competitiveness and the average performance of LADO. Finally, we run an experiment of decentralized battery management for sustainable computing. Our results highlight the potential of ML augmentation to improve the average performance in LADO-Lin and LADO as well as the guaranteed worst-case performance of LADO.

References

[1] Hamidreza Shahbazi and Farid Karbalaei. Decentralized voltage control of power systems using multi-agent systems. *Journal of Modern Power Systems and Clean Energy*, 8(2):249–259, 2020.

[2] Yuanyuan Shi, Guannan Qu, Steven Low, Anima Anandkumar, and Adam Wierman. Stability constrained reinforcement learning for real-time voltage control. *arXiv preprint arXiv:2109.14854*, 2021.

[3] Saghar Hosseini, Airlie Chapman, and Mehran Mesbahi. Online distributed convex optimization on dynamic networks. *IEEE Transactions on Automatic Control*, 61(11):3545–3550, 2016.

[4] Amal Feriani and Ekram Hossain. Single and multi-agent deep reinforcement learning for ai-enabled wireless networks: A tutorial. *IEEE Communications Surveys & Tutorials*, 23(2):1226–1252, 2021.

[5] Yasar Sinan Nasir and Dongning Guo. Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks. *IEEE Journal on Selected Areas in Communications*, 37(10):2239–2250, 2019.

[6] Fuqiang Yao and Luliang Jia. A collaborative multi-agent reinforcement learning anti-jamming algorithm in wireless networks. *IEEE wireless communications letters*, 8(4):1024–1027, 2019.

[7] Felipe Caro and Jérémie Gallien. Clearance pricing optimization for a fast-fashion retailer. *Operations research*, 60(6):1404–1422, 2012.

[8] Ozan Candogan, Kostas Bimpikis, and Asuman Ozdaglar. Optimal pricing in networks with externalities. *Operations Research*, 60(4):883–905, 2012.

[9] Kaixiang Lin, Shu Wang, and Jiayu Zhou. Collaborative deep reinforcement learning. *arXiv preprint arXiv:1702.05796*, 2017.

[10] Xuanyu Cao and Tamer Başar. Decentralized online convex optimization with feedback delays. *IEEE Transactions on Automatic Control*, 67(6):2889–2904, 2021.

[11] Nicolas Christianson, Tinashe Handina, and Adam Wierman. Chasing convex bodies and functions with black-box advice. In *COLT*, 2022.

[12] Gautam Goel, Yiheng Lin, Haoyuan Sun, and Adam Wierman. Beyond online balanced descent: An optimal algorithm for smoothed online optimization. In *NeurIPS*, volume 32, 2019.

[13] Mehrdad Mahdavi, Rong Jin, and Tianbao Yang. Trading regret for efficiency: Online convex optimization with long term constraints. *J. Mach. Learn. Res.*, 13(1):2503–2528, sep 2012.

[14] Gautam Goel and Adam Wierman. An online algorithm for smoothed online convex optimization. *SIGMETRICS Perform. Eval. Rev.*, 47(2):6–8, December 2019.

[15] Lijun Zhang, Wei Jiang, Shiyin Lu, and Tianbao Yang. Revisiting smoothed online learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[16] Guanya Shi, Yiheng Lin, Soon-Jo Chung, Yisong Yue, and Adam Wierman. Online optimization with memory and competitive control. *Advances in Neural Information Processing Systems*, 33:20636–20647, 2020.

[17] Weici Pan, Guanya Shi, Yiheng Lin, and Adam Wierman. Online optimization with feedback delay and nonlinear switching cost. *Proc. ACM Meas. Anal. Comput. Syst.*, 6(1), Feb 2022.

[18] Niangjun Chen, Gautam Goel, and Adam Wierman. Smoothed online convex optimization in high dimensions via online balanced descent. In *COLT*, 2018.

[19] Alec Koppel, Felicia Y. Jakubiec, and Alejandro Ribeiro. A saddle point algorithm for networked online convex optimization. *IEEE Transactions on Signal Processing*, 63(19):5149–5164, 2015.

[20] Xiuxian Li, Xinlei Yi, and Lihua Xie. Distributed online convex optimization with an aggregative variable. *IEEE Transactions on Control of Network Systems*, 2021.

[21] Xuanyu Cao and Tamer Başar. Decentralized online convex optimization based on signs of relative states. *Automatica*, 129:109676, 2021.
[22] Saghar Hosseini, Airlie Chapman, and Mehran Mesbahi. Online distributed convex optimization on dynamic networks. *IEEE Transactions on Automatic Control*, 61(11):3545–3550, 2016.
[23] Weiwei Kong, Christopher Liaw, Aranyak Mehta, and D. Sivakumar. A new dog learns old tricks: RL finds classic optimization algorithms. In *ICLR*, 2019.
[24] Thomas Barrett, William Clements, Jakob Foerster, and Alex Lvovsky. Exploratory combinatorial optimization with reinforcement learning. In *AAAI*, 2020.
[25] Han Zhang, Wenzhong Li, Shaohua Gao, Xiaoliang Wang, and Baoliu Ye. Reles: A neural adaptive multipath scheduler based on deep reinforcement learning. In *INFOCOM*, 2019.
[26] Zhihui Shao, Jianyi Yang, Cong Shen, and Shaolei Ren. Learning for robust combinatorial optimization: Algorithm and application. In *INFOCOM*, 2022.
[27] Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Basar. Fully decentralized multi-agent reinforcement learning with networked agents. In *International Conference on Machine Learning*, pages 5872–5881. PMLR, 2018.
[28] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, pages 321–384, 2021.
[29] Afshin Oroojlooy and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *Applied Intelligence*, pages 1–46, 2022.
[30] Pengfei Li, Jianyi Yang, and Shaolei Ren. Robustified learning for online optimization with memory costs. In *INFOCOM*, 2023.
[31] Alexander Wei and Fred Zhang. Optimal robustness-consistency trade-offs for learning-augmented online algorithms. In *NeurIPS*, 2020.
[32] Joan Boyar, Lene M. Favrholdt, Christian Kudahl, Kim S. Larsen, and Jesper W. Mikkelsen. Online algorithms with advice: A survey. *SIGACT News*, 47(3):93–129, August 2016.
[33] Étienne Bamas, Andreas Maggiori, and Ola Svensson. The primal-dual method for learning augmented algorithms. *Advances in Neural Information Processing Systems*, 33:20083–20094, 2020.
[34] Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *J. ACM*, 68(4), July 2021.
[35] Daan Rutten, Nicolas Christianson, Debankur Mukherjee, and Adam Wierman. Smoothed online optimization with unreliable predictions. *Proc. ACM Meas. Anal. Comput. Syst.*, 7(1), mar 2023.
[36] Yiheng Lin, Judy Gan, Guannan Qu, Yash Kanoria, and Adam Wierman. Decentralized online convex optimization in networked systems. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 13356–13393. PMLR, 17–23 Jul 2022.
[37] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska. Dynamic right-sizing for power-proportional data centers. In *INFOCOM*, 2011.
[38] Mohammad A. Islam, Kishwar Ahmed, Hong Xu, Nguyen H. Tran, Gang Quan, and Shaolei Ren. Exploiting spatio-temporal diversity for water saving in geo-distributed data centers. *IEEE Transactions on Cloud Computing*, 6(3):734–746, 2018.
[39] Meta. Sustainability report. https://sustainability.fb.com/, 2021.
[40] Gautam Goel, Yiheng Lin, Haoyuan Sun, and Adam Wierman. Beyond online balanced descent: An optimal algorithm for smoothed online optimization. *Advances in Neural Information Processing Systems*, 32, 2019.
[41] Richard Cheng, Abhinav Verma, Gabor Orosz, Swarat Chaudhuri, Yisong Yue, and Joel Burdick. Control regularization for reduced variance reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1141–1150. PMLR, 09–15 Jun 2019.
[42] Hoang M. Le, Andrew Kang, Yisong Yue, and Peter Carr. Smooth imitation learning for online sequence prediction. In *ICML*, 2016.
[43] Pengfei Li, Jianyi Yang, and Shaolei Ren. Learning for edge-weighted online bipartite matching with robustness guarantees. In *ICML*, 2023.
[44] Yunchang Yang, Tianhao Wu, Han Zhong, Evrard Garcelon, Matteo Pirotta, Alessandro Lazaric, Liwei Wang, and Simon Shaolei Du. A reduction-based framework for conservative bandits and reinforcement learning. In *International Conference on Learning Representations*, 2022.
[45] Jakub Chłędowski, Adam Polak, Bartosz Szabucki, and Konrad Tomasz Żołna. Robust learning-augmented caching: An experimental study. In *ICML*, 2021.
[46] Le Yi Wang, Caisheng Wang, George Yin, Feng Lin, Michael P. Polis, Caiping Zhang, and Jiuchun Jiang. Balanced control strategies for interconnected heterogeneous battery systems. *IEEE Transactions on Sustainable Energy*, 7(1):189–199, 2016.

[47] Ana Radovanović, Ross Koningstein, Ian Schneider, Bokan Chen, Alexandre Duarte, Binz Roy, Diyue Xiao, Maya Haridasan, Patrick Hung, Nick Care, Saurav Talukdar, Eric Mullen, Kendal Smith, MariEllen Cottman, and Walfredo Cirne. Carbon-aware computing for datacenters. *IEEE Transactions on Power Systems*, 38(2):1270–1280, 2023.
[48] Pengfei Li, Jianyi Yang, Adam Wierman, and Shaolei Ren. Towards environmentally equitable AI via geographical load balancing. In *e-Energy*, 2024.
[49] Amba Kak and Sarah Myers West. AI Now 2023 landscape: Confronting tech power. *AI Now Institute*, April 2023.
[50] Alejandro Garofali Acosta, Shaun Riordan, and Mario Torres Jarrín. The environmental and ethical challenges of artificial intelligence. *ThinkTwenty (T20) Policy Brief*, July 2023.
[51] UNESCO. Recommendation on the ethics of artificial intelligence. In *Policy Recommendation*, 2022.
[52] Elad Hazan. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.
[53] Keerti Anand, Rong Ge, Amit Kumar, and Debmalya Panigrahi. A regression approach to learning-augmented online algorithms. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.
[54] Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ml predictions. In *NeurIPS*, 2018.
[55] Goran Zuzic, Di Wang, Aranyak Mehta, and D. Sivakumar. Learning robust algorithms for online allocation problems using adversarial training. In *https://arxiv.org/abs/2010.08418*, 2020.
[56] Pengfei Li, Jianyi Yang, and Shaolei Ren. Expert-calibrated learning for online optimization with switching costs. In *SIGMETRICS*, 2022.
[57] Antonios Antoniadis, Christian Coester, Marek Elias, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. In *ICML*, 2020.
[58] Andrew A Chien, Liuzixuan Lin, Hai Nguyen, Varsha Rao, Tristan Sharma, and Rajini Wijayawardana. Reducing the carbon impact of generative AI inference (today and in 2035). In *Proceedings of the 2nd Workshop on Sustainable Computer Systems*, HotCarbon '23, New York, NY, USA, 2023. Association for Computing Machinery.
[59] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 153–167, 2017.
[60] Manajit Sengupta, Yu Xie, Anthony Lopez, Aron Habte, Galen Maclaurin, and James Shelby. The national solar radiation data base (nsrdb). *Renewable and sustainable energy reviews*, 89:51–60, 2018.
[61] Can Wan, Jian Zhao, Yonghua Song, Zhao Xu, Jin Lin, and Zechun Hu. Photovoltaic and solar power forecasting for smart grid energy management. *CSEE Journal of Power and Energy Systems*, 1(4):38–46, 2015.
[62] Asis Sarkar and Dhiren Kumar Behera. Wind turbine blade efficiency and power calculation with electrical analogy. *International Journal of Scientific and Research Publications*, 2(2):1–5, 2012.
[63] Zhipeng Tu, Xi Wang, Yiguang Hong, Lei Wang, Deming Yuan, and Guodong Shi. Distributed online convex optimization with compressed communication. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 34492–34504. Curran Associates, Inc., 2022.
[64] Daan Rutten, Nico Christianson, Debankur Mukherjee, and Adam Wierman. Online optimization with untrusted predictions. *CoRR*, abs/2202.03519, 2022.
[65] Yifan Wu, Roshan Shariff, Tor Lattimore, and Csaba Szepesvári. Conservative bandits. In *International Conference on Machine Learning*, pages 1254–1262. PMLR, 2016.
[66] Yunchang Yang, Tianhao Wu, Han Zhong, Evrard Garcelon, Matteo Pirotta, Alessandro Lazaric, Liwei Wang, and Simon Shaolei Du. A reduction-based framework for conservative bandits and reinforcement learning. In *International Conference on Learning Representations*, 2021.
[67] Evrard Garcelon, Mohammad Ghavamzadeh, Alessandro Lazaric, and Matteo Pirotta. Conservative exploration in reinforcement learning. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1431–1441. PMLR, 26–28 Aug 2020.
[68] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
[69] Noelle Walsh. How Microsoft measures datacenter water and energy use to improve Azure Cloud sustainability. *Microsoft Azure Blog*, April 2022.
[70] Tesla. Tesla powerwall 2 datasheet - North America. https://www.tesla.com/sites/default/files/pdfs/powerwall/Powerwall%202_AC_Datasheet_en_northamerica.pdf.
[71] LG Electronics. LG electronics home series energy storage system datasheet. https://www.lg.com/us/ess/pdf/Resi_LGEUS_Home_8_Spec_0524.pdf.

[72] SolarEdge. Solaredge energy bank datasheet. https://knowledge-center.solaredge.com/sites/kc/files/se-energy-bank-battery-datasheet-nam.pdf.
[73] IQ Battery System. IQ battery 10t datasheet. https://www.switchsolarusa.com/wp-content/uploads/2023/02/IQ-Battery-10T-DS-EN-US-10-25-2021.pdf.
[74] FranklinWH. Franklin home power datasheet, https://www.franklinwh.com/document/franklin-home-power-v11-datasheet.
[75] Moritz Hardt and Max Simchowitz. Convex optimization and approximation. https://ee227c.github.io/notes/ee227c-notes.pdf, 2018.

A Extension of LADO to Directed Graphs

In some real-world applications (e.g., wireless networks), the connections between nodes are directional, instead of the bi-directional connections in undirected graphs. Thus, we extend LADO to a directed graph setting. We first show how to modify the design of adaptive spatial cost splitting and reservation cost for a directed graph, followed by an average cost performance bound.

Consider a network with a finite set $\mathcal{V}$ of nodes. We model the network as a directed graph, denoted by $(\mathcal{V}, \mathcal{E})$, where $\mathcal{E}$ represents the set of directional edges between nodes in $\mathcal{V}$. For each edge $(v,u) \in \mathcal{E}$, the spatial cost is denoted as $s_t^{(v,u)}(x_t^v, x_t^u)$, which depends on the actions of the node v and u at time t. Since the edge is directional, the spatial cost $s_t^{(v,u)}(x_t^v, x_t^u)$ may not be equal to the cost $s_t^{(u,v)}(x_t^u, x_t^v)$ incurred in the opposite direction.

In a directed graph, the locally computable constraint in Eqn. (13) for λ-competitiveness can be rewritten as below

$$\begin{aligned}
&\sum_{\tau=1}^{t} f_\tau^v(x_\tau^v) + \sum_{\tau=1}^{t} c_\tau^v(x_\tau^v, x_{\tau-1}^v) + \sum_{\tau=1}^{t-1}\Big(\sum_{(v,u)\in\mathcal{E}} \kappa_\tau^{(v,u)} \cdot s_\tau^{(v,u)}(x_\tau^v, x_\tau^u) + \sum_{(u,v)\in\mathcal{E}} \kappa_\tau^{(u,v)} \cdot s_\tau^{(u,v)}(x_\tau^u, x_\tau^v)\Big) \\
&+ R(x_t^v, x_t^{v,\dagger}) \le (1+\lambda)\Bigg(\sum_{\tau=1}^{t} f_\tau^v(x_\tau^{v,\dagger}) + \sum_{\tau=1}^{t} c_\tau^v(x_\tau^{v,\dagger}, x_{\tau-1}^{v,\dagger}) + \sum_{\tau=1}^{t-1}\Big(\sum_{(v,u)\in\mathcal{E}} \kappa_\tau^{(v,u)} \cdot s_\tau^{(v,u)}(x_{\tau-1}^{v,\dagger}, x_\tau^{u,\dagger}) \\
&+ \sum_{(u,v)\in\mathcal{E}} \kappa_\tau^{(u,v)} \cdot s_{\tau-1}^{(u,v)}(x_\tau^{u,\dagger}, x_\tau^{v,\dagger})\Big)\Bigg),
\end{aligned} \tag{19}$$

where the weight $\kappa_\tau^{(v,u)}$ for splitting the spatial cost $s_\tau^{(v,u)}$ is adaptively chosen as

$$\kappa_\tau^{(v,u)} = \frac{\|x_\tau^v - x_\tau^{v,\dagger}\|^2}{\|x_\tau^v - x_\tau^{v,\dagger}\|^2 + \|x_\tau^u - x_\tau^{u,\dagger}\|^2} \tag{20}$$

and the reservation cost is

$$R(x_t^v, x_t^{v,\dagger}) = \frac{\ell_T + \ell_S \cdot (D_v^{in} + D_v^{out})}{2}\Big(1 + \frac{1}{\lambda_0}\|x_t^v - x_t^{v,\dagger}\|^2\Big) \tag{21}$$

where D_v^{in} and D_v^{out} denote the in-degree and out-degree of node v. In other words, D_v^{in} represents the number of edges directed towards node v and D_v^{out} is the number of edges directed from node v. Additionally, based on Assumptions 2.1 - 2.3, the temporal and spatial costs are ℓ_T- and ℓ_S-smooth with respect to the actions, respectively.

If there exist bi-directional edges between node u and v, the weights for splitting spatial costs $s_{t-1}^{(v,u)}$ and $s_{t-1}^{(u,v)}$ are identical. This is because the weight $\kappa_\tau^{(v,u)}$ allocates the spatial cost according to the potential risk of spatial cost increases due to nodes v and u, as measured by the distances between their actions to the expert advice. Since the risk is independent of the direction, the spatial cost splitting weight $\kappa_\tau^{v,u}$ remains the same regardless of the edge direction.

Next, we analyze the average cost of LADO in a directed graph.

Corollary A.1. *(Average Cost of* $\texttt{LADO}(\tilde{\pi})$ *for directed graph) Given any ML policy* $\tilde{\pi}$*, for any* $\lambda > 0$*, by optimally setting* $\lambda_0 = \sqrt{1+\lambda} - 1$*, the average cost of* $\texttt{LADO}(\tilde{\pi})$ *is bounded by*

$$AVG(\texttt{LADO}(\tilde{\pi})) \le \min\left\{(1+\lambda)AVG(\pi^\dagger), \left(\sqrt{AVG(\tilde{\pi})} + \sqrt{\sum_{v\in\mathcal{V}} \omega_v(\lambda, \tilde{\pi}, \pi^\dagger)}\right)^2\right\},$$

where $AVG(\pi^\dagger)$ *and* $AVG(\tilde{\pi})$ *are the average costs of the expert policy and the ML policy, respectively, and* $\omega_v(\lambda, \tilde{\pi}, \pi^\dagger) = \mathbb{E}_{g_{1:T}}\left\{\sum_{t=1}^{T}\left[\frac{\ell_f+2\cdot\ell_T+\ell_S\cdot(D_v^{in}+D_v^{out})}{2}\|\tilde{x}_t^v - x_t^{v,\dagger}\|^2 - (\sqrt{1+\lambda}-1)^2 cost_{v,t}^\dagger\right]^+\right\}$ *in which* $cost_{v,t}^\dagger$ *is the expert's node and temporal cost for node* v *at time* t.

In Corollary A.1, the average cost of LADO is bounded by the expert's average cost scaled by $(1+\lambda)$ and the average cost of learning-based policy $\tilde{\pi}$ along with an additional cost introduced by the projection process, corresponding to the two terms in the min operators, respectively. When relaxing the parameter λ in the competitiveness guarantee, it grants more freedom for LADO to follow the ML policy with less restrictive constraints, resulting in a smaller $\omega_v(\lambda, \tilde{\pi}, \pi^\dagger)$ for all nodes in $\mathcal{V}$. Additionally, the node degree plays an important role, along with the overall action distance between the expert and ML policies. Importantly, the spatial cost is shared by connected nodes, according to the adaptive spatial cost splitting weight $\kappa_t^{(v,u)}$. Therefore, the spatial uncertainty of node v is determined jointly by the spatial connections originating from and directed towards that specific node. For the nodes with greater spatial uncertainties (quantified by the sum of in- and out-degrees) and/or larger policy misalignment in terms of $\|\tilde{x}_t^v - x_t^{v,\dagger}\|$, it is more challenging to adopt the ML policy, potentially incurring a higher cost during the projection. As the graph density increases, more spatial uncertainties are introduced to the connected nodes, thus increasing $\omega_v(\lambda, \tilde{\pi}, \pi^\dagger)$ incurred by the projection process. However, with the competitiveness guarantee, the average cost of LADO can always be bounded by the expert's cost up to a scaling factor of $(1+\lambda)$, regardless of the graph topology or the chosen ML policy.

B Optimal Projection-Aware ML Training

Theorem 5.2 applies to any ML models, including ML models that are trained as standalone optimizers without considering the design of LADO and hence may have training-testing objective mismatches. To improve the average cost performance, we consider the following ML policy $\tilde{\pi}_\lambda^\circ$ that is optimally trained with explicit consideration of the downstream projection in LADO:

$$\tilde{\pi}_\lambda^\circ = \arg\min_{\pi} \mathbb{E}_{g_{1:T}}\left[cost(\texttt{LADO}(\pi), g_{1:T})\right], \tag{22}$$

where the projected ML prediction by LADO is explicitly used as the action in the cost. The policy $\tilde{\pi}_\lambda^\circ$ can be trained offline using implicit differentiation (i.e., the added projection in Line 4 for LADO in Algorithm 1 can be implicitly differentiated based on KKT conditions) [68]. Like in other learning-augmented algorithms [11], we consider that $\tilde{\pi}_\lambda^\circ$ is already available for online inference by individual agents. Next, we use $\texttt{LADO}(\tilde{\pi}_\lambda^\circ)$ to emphasize the usage of $\tilde{\pi}_\lambda^\circ$ in LADO, and show its average cost bound. The proof is deferred to Appendix F.4.

Corollary B.1 (Average cost of $\texttt{LADO}(\tilde{\pi}_\lambda^\circ)$)**.** *Given the optimal projection-aware ML policy* $\tilde{\pi}_\lambda^\circ$*, for any* $\lambda > 0$*, by optimally setting* $\lambda_0 = \sqrt{1+\lambda}-1$*, the average cost of* $\texttt{LADO}(\tilde{\pi}_\lambda^\circ)$ *is upper bounded by*

$$\begin{aligned} AVG(\texttt{LADO}(\tilde{\pi}_\lambda^\circ)) \le \min\Bigg\{&(1-\alpha_\lambda)AVG(\pi^\dagger) + \alpha_\lambda AVG(\tilde{\pi}^*), \\ &\left(\sqrt{AVG(\tilde{\pi}^*)} + \sqrt{\sum_{v\in\mathcal{V}}\omega_v(\lambda, \tilde{\pi}^*, \pi^\dagger)}\right)^2\Bigg\} \end{aligned} \tag{23}$$

where $AVG(\pi^\dagger)$ *and* $AVG(\tilde{\pi}^*)$ *are the average costs of the expert and the optimal projection-unaware ML policy* $\tilde{\pi}^* = \arg\min_\pi \mathbb{E}_{g_{1:T}}\left[cost(\pi, g_{1:T})\right]$*, respectively,* $\alpha_\lambda = \min\left\{(\sqrt{1+\lambda}-1)\sqrt{\frac{2}{\ell_T+\ell_f+D_{\max}\cdot\ell_S}}\cdot\hat{C}, 1\right\}$ *with* $D_{\max} = \max_{v\in\mathcal{V}} D_v$ *being the maximum degree in the network and the expert's minimum*

single-node cumulative cost normalized by the cumulative expert-ML action distance is defined as $\hat{C} = \min_{g_{1:T}\in\mathcal{G}} \min_{v\in\mathcal{V},t\in[1,T]} \frac{cost_v(x^{v,\dagger}_{1:t})}{\sum_{i=1}^{t}\|x_i^{v,\dagger}-\tilde{x}_i^{v,*}\|^2}$. *Besides, we define*

$$\omega_v(\lambda,\tilde{\pi}^*,\pi^\dagger) = \sum_{t=1}^{T}\mathbb{E}_{g_{1:T}}\left\{\left[\frac{\ell_f+2\cdot\ell_T+\ell_S\cdot D_v}{2}\|\tilde{x}_t^{v,*}-x_t^{v,\dagger}\|^2-(\sqrt{1+\lambda}-1)^2\cdot cost^{\dagger}_{v,t}\right]^+\right\}, \tag{24}$$

where $cost^{\dagger}_{v,t}$ *is the expert's node and temporal cost for node* v *at time* t.

Corollary B.1 formally demonstrates the benefits of using the optimal projection-*aware* ML policy (22) compared to the optimal projection-*unaware* ML policy $\tilde{\pi}^* = \arg\min_\pi \mathbb{E}_{g_{1:T}}[cost(\pi, g_{1:T})]$. Specifically, by the optimality of $AVG(\tilde{\pi}^*)$ that does not consider λ-competitiveness, we naturally have $AVG(\tilde{\pi}^*) \le AVG(\pi^\dagger)$. Thus, the first term in (23) shows that, the average cost of $\texttt{LADO}(\tilde{\pi}^\circ_\lambda)$ is no greater than the expert by using the projection-aware ML policy $\tilde{\pi}^\circ_\lambda$. This is because the expert policy is intuitively a feasible solution in our λ-competitiveness ML policy space, while the policy $\tilde{\pi}^\circ_\lambda$ in (22) is the optimal one that specifically minimizes the average cost of $\texttt{LADO}(\tilde{\pi}^\circ_\lambda)$. By contrast, even by using the optimal projection-unaware ML policy $\tilde{\pi}^*$, the average cost of $\texttt{LADO}(\tilde{\pi}^*)$ is bounded by $(1+\lambda)AVG(\pi^\dagger)$ in the first term of Theorem 5.2, since the added projection during actual inference can void the optimality of $\tilde{\pi}^*$ and result in a higher average cost up to $(1+\lambda$ times of the expert's cost. The root reason for the advantage of the optimal projection-*aware* ML policy (22) in terms of the average cost is that its ML prediction is specifically customized to LADO. On the other hand, even though $\tilde{\pi}^* = \arg\min_\pi \mathbb{E}_{g_{1:T}}[cost(\pi, g_{1:T})]$ is the optimal-*unconstrained* ML policy on its own, its optimality can no longer hold when modified by LADO for λ-competitiveness during actual online inference.

Finally, the second term inside min in Corollary B.1 shows that the average cost of $\texttt{LADO}(\tilde{\pi}^\circ_\lambda)$ with the optimal projection-aware ML policy $\tilde{\pi}^\circ_\lambda$ in (22) is upper bounded by that of $\texttt{LADO}(\tilde{\pi}^*_\lambda)$, since $\texttt{LADO}(\tilde{\pi}^*_\lambda)$ is a feasible policy satisfying λ-competitiveness by our design. Like in Theorem 5.2, it reinforces the insight that LADO can better exploit the potential of ML predictions for average performance improvement when $\lambda > 0$ increases.

C Additional Experiments for Decentralized Battery Management

In this section, we present more results on different testing distributions in 3-node network and an extended set of experiments for large networks incorporating a wider variety of battery units. Beyond the self-degradation coefficient A_v, each battery unit also exhibits unique characteristics in terms of the storage capacity and maximum continuous discharge current. These values for the battery units are derived from publicly available data on energy storage systems. We further investigate the impact of network topology, on the overall cost of LADO and other baseline algorithms.

C.1 ML Model Architecture and More Results for Networks with 3 Nodes

C.1.1 ML Model Architecture. The ML predictions used in our algorithm are computed using a RNN with 2 recurrent layers, each with 8 hidden features. In all of our experiments, each problem instance spans 24 hours, and each time step represents one hour. For the training processes, we used the net energy demand trace from the first two months of 2015, which contains 1440 hourly data samples and produces a total of 1416 24-hour sequences. The ML model is optimized by Adam with a learning rate of 10^{-3} for 60 epochs in total. After training, the weights of the ML model are shared between all nodes with different coefficients A_v. On average, the training process takes 3 minutes on a 2020 MacBook Air with 8GB memory, and our testing process takes about 1 second. For testing, we use the net demand traces from April to March in the default case.

C.1.2 Results. Based on the setup in Section 6.1, we choose 3 fully connected battery units, where the self-degradation coefficients A_v of the battery units are set as 0.9, 0.93, 0.95, respectively. We present more results on different testing distributions in 3-node network.

In-distribution testing. First, we consider an ideal case, called in-distribution testing, where the ML model is trained and tested on the same data distribution. Naturally, the ML model is expected to perform very well. Table 2 shows that the ML model outperforms the expert in terms of the empirical CR. By increasing γ, LADO-Lin follows the ML more closely and hence also achieves a better average cost. Nonetheless, its advantage in terms of the average cost comes without competitiveness guarantees. By contrast, LADO and LADO-OPT can achieve both good average costs and guaranteed competitiveness simultaneously.

	Expert	ML	HitOnly	Greedy	LADO			
					$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$
AVG	127.79	95.45	119.06	192.11	106.53	98.93	96.41	95.66
CR	1.738	1.487	2.021	3.264	1.431	1.446	1.490	1.488
	LADO-Lin				LADO-OPT			
	$\gamma = 0.1$	$\gamma = 0.3$	$\gamma = 0.5$	$\gamma = 0.9$	$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$
AVG	120.20	107.91	99.50	**94.32**	105.33	97.46	95.47	95.00
CR	1.668	1.546	1.449	1.446	1.420	**1.400**	1.416	1.420

Table 2. In-distribution testing for a 3-node network. The average cost of OPT is 83.37.

Out-of-distribution testing. Next, we inject large Gaussian noise into the testing dataset and consider the out-of-distribution testing case where the testing and training distributions are different. The results are shown in Table 3. In this case, the ML model has a higher average cost as well as empirical CR than Expert. While LADO-Lin can reduce the average cost by slightly incorporating the ML prediction into its action ($\gamma = 0.1$), this advantage quickly vanishes as γ increases. On the other hand, by training the ML model in a projection-aware manner, LADO-OPT can keep its average cost low while still offering guaranteed competitiveness.

	Expert	ML	HitOnly	Greedy	LADO			
					$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$
AVG	141.08	188.86	320.20	200.03	138.93	156.71	174.00	184.80
CR	1.623	5.452	10.488	2.748	1.808	2.188	2.857	3.969
	LADO-Lin				LADO-OPT			
	$\gamma = 0.1$	$\gamma = 0.3$	$\gamma = 0.5$	$\gamma = 0.9$	$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$
AVG	**136.73**	134.12	139.62	174.95	137.60	145.93	150.89	152.98
CR	**1.605**	1.881	2.501	4.706	1.783	2.198	2.650	3.162

Table 3. Out-of-distribution testing for a 3-node network. The average cost of OPT is 95.77.

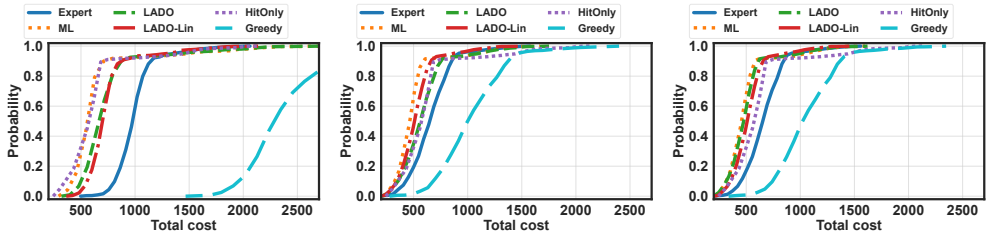
C.2 Results for Large Networks

We present an extended set of experiments for large networks utilizing a variety of battery units.

C.2.1 Experimental Setup. Following a similar experimental setup in the experiment of a 3-node network, the data center's energy demand $P_{d,t}$ is derived using the hourly workload trace from [69] and the renewable energy generation $P_{r,t}$ is estimated with the weather-related statistics from [60] . The net energy demand of the data center, $P_{n,t} = P_{d,t} - P_{r,t}$ is then served by the energy storage system, powered by a pool of battery units. Then, we use a sliding window to generate 24-hour net demand sequences as the datasets, where each sequence has 25 successive normalized net demands (from hour 0 to hour 24). In this experiment, we derive the storage capacity and rated output power

	Usable battery capacity (kWh)	Rated Charging Power (kW)	Rated Discharging Power(kW)	Peak Discharge Power (kW)
Tesla Powerwall 2 [70]	13.50	5.00	5.00	7.00
LG ESS (Home 8) [71]	14.40	5.40	7.50	9.00
Solar Edge (BAT-10K1P) [72]	9.70	5.00	5.00	7.50
IQ Battery 10T [73]	10.08	3.84	3.84	5.76
FranklinWH [74]	13.60	5.00	5.00	10.00

Table 4. Specifications of the commercially available home energy storage systems used in the experiment.



(a) Total cost (complete graph) (b) Total cost (star graph) (c) Total cost (chain graph)

Fig. 3. The total cost distribution of LADO and baseline algorithms with various graph topologies (15-node network). By default, the competitiveness requirement λ is set to 1 in LADO for all graphs.

from five different commercially available battery storage systems, including Tesla Powerwall+, LG ESS, Solar Edge, Enphase IQ and FranklinWH, where the detailed specifications can be found in Table 4. We normalize their storage capacity and continuous output power relative to the Tesla Powerwall+ for easy comparison. For an easy comparison, their relative storage capacities are 1, 1.07, 0.72, 0.78, 1.01, and their relative continuous output powers are 1, 1.07, 0.71, 0.55, 0.71, respectively. To account for variations in the battery health of these battery units, we employ three self-degradation coefficients A_v, set as 0.9, 0.93, 0.95, consistent with the main experiment. By default, we set $b = 5$ and $c = 2$ as the weights for the temporal and spatial costs, respectively. In total, by considering the various battery characteristics along with the self-degradation coefficients, we create 15 distinct battery node configurations for our experiment.

C.2.2 Results. For the three representative network topologies (i.e., the complete graph, star graph, and linear chain graph), the empirical distribution of the total cost for each algorithm compared is shown in Fig. 3. Moreover, Tables [5,6,7] summarize the average total costs and competitive ratios of all considered algorithms for the three representative topologies, respectively. Both the complete graph and the star graph have the same maximum node degree, whereas the graph density of the star graph is significantly lower. This explains why all the algorithms considered exhibit lower costs on the star graph, which is also consistent with our new theoretical analysis of the cost performance in Theorem 5.2. Furthermore, LADO can leverage the power of ML policy more efficiently with the reduced spatial cost uncertainties associated with fewer connections in the star graph.

Interestingly, the star graph and linear chain graph share the same number of edges (or graph density), while the star graph concentrates node degrees, leading to distinct cost behaviors. This is evident when comparing the average cost distributions in Fig. 3(b) and Fig. 3(c). The uniform node degree distribution in the chain graph allows LADO to leverage the power of learning-based policy more efficiently and further reduce the total cost in the linear chain graph. This empirical

	Expert	ML	HitOnly	Greedy	LADO			
					$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$
AVG	662.38	489.03	613.73	1054.87	559.59	519.75	500.63	491.86
CR	2.065	4.653	7.380	4.227	1.948	2.234	2.422	3.068
	LADO-Lin				LADO-OPT			
	$\gamma = 0.1$	$\gamma = 0.3$	$\gamma = 0.5$	$\gamma = 0.9$	$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$
AVG	618.72	548.95	502.58	480.04	555.77	509.92	488.91	**481.59**
CR	1.972	2.021	2.382	4.042	**1.912**	2.032	2.186	2.639

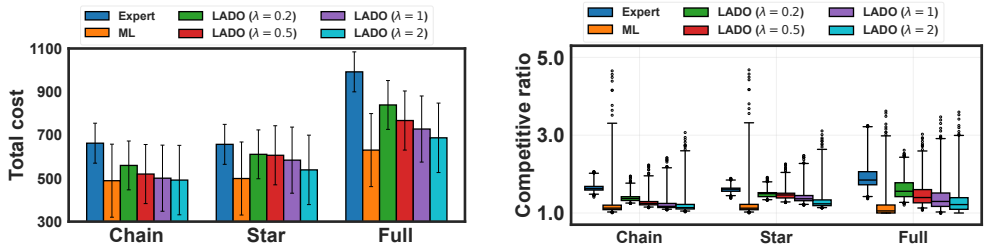
Table 5. AVG and CR comparison between different algorithms for a 15-node chain graph. The average cost of OPT in the testing dataset is 405.37.

	Expert	ML	HitOnly	Greedy	LADO			
					$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$
AVG	656.98	499.12	613.73	1030.78	610.87	606.46	583.98	539.01
CR	1.891	4.679	7.295	3.748	1.916	2.262	2.477	3.109
	LADO-Lin				LADO-OPT			
	$\gamma = 0.1$	$\gamma = 0.3$	$\gamma = 0.5$	$\gamma = 0.9$	$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$
AVG	614.60	547.57	504.18	497.77	602.06	571.04	528.68	**497.29**
CR	**1.802**	1.889	2.309	4.614	1.882	2.039	2.225	2.666

Table 6. AVG and CR comparison between different algorithms for a 15-node star graph, shown as Fig. 1(b) . The average cost of OPT in the testing dataset is 411.06.

	Expert	ML	HitOnly	Greedy	LADO			
					$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$
AVG	991.99	630.41	613.73	2401.08	839.00	767.09	727.77	687.17
CR	3.245	3.619	4.048	11.577	2.614	3.026	3.469	3.595
	LADO-Lin				LADO-OPT			
	$\gamma = 0.1$	$\gamma = 0.3$	$\gamma = 0.5$	$\gamma = 0.9$	$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$
AVG	914.90	787.99	697.48	625.63	650.70	637.81	632.44	**622.09**
CR	3.042	2.718	2.670	3.377	**2.047**	2.090	2.478	2.901

Table 7. AVG and CR comparison between different algorithms for a 15-node complete graph. The average cost of OPT in the testing dataset is 524.79.



(a) Total cost with different λ

(b) Competitive ratio with different λ

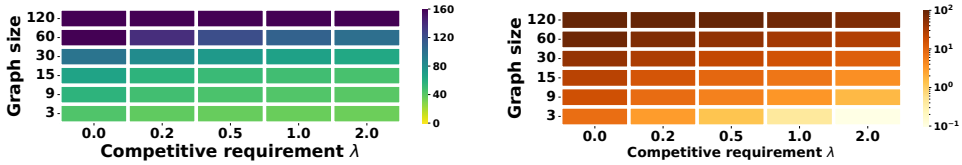
Fig. 4. The comparison of total cost and competitive ratio distribution between LADO and other baseline algorithms for a 15-node network.

observation aligns with Theorem 5.2, which suggests a more distributed node degree reduces a lower term $\omega_v(\lambda, \tilde{\pi}, \pi^\dagger)$ due to perturbations to the ML policy.

Impact of competitiveness requirement λ: Next, we empirically evaluate the robustness-consistency trade-off of LADO under different graph topologies. For all the three graphs, the empirical average cost of learning-based policy (ML) outperforms the state-of-the-art expert policy, which highlights the importance of reducing the deviation of LADO from the ML policy to improve the average cost performance. As suggested by Theorem 5.2, reducing the spatial cost uncertainty (fewer node connections) or relaxing the competitiveness requirement (larger λ) leads to a smaller deviation of LADO from the ML policy, which in turn reduces the perturbations to the ML policy because of projection. Fig. 4(a) clearly illustrates this trend by comparing the average cost of LADO under different settings. However, such performance improvement comes at an expense in terms of the competitiveness guarantees. As illustrated in Fig. 4(b), a larger λ weakens the competitiveness guarantees of LADO, reducing its worst-case protection and potentially leading to a higher competitive ratio.

Impact of network topologies: In this experiment, we first evaluate LADO on graphs with the same number of nodes but varying random graph topologies. Specifically, we consider 15 battery units with varying characteristics and randomly selected spatial connections between these nodes. The minimum number of edges is set the same as a star graph instead of zero, since otherwise the nodes' decisions would become uncorrelated without any spatial connections. Starting from the star graph, we gradually add random edges between nodes to increase the graph density until the graph is fully connected. Additionally, we experiment with different competitiveness requirements under these graph topologies. The total cost and regret of LADO compared to the ML policy are shown in Fig. 2.

As the graph density increases with more spatial connections between nodes, the total cost of LADO rises monotonically, showing a direct correlation between increased node connectivity and greater spatial cost uncertainties. To focus on the impacts solely due to the projection process, we compare the regret of LADO against the ML policy, where both algorithms are evaluated under the same graph topologies. As shown in Fig. 2(b), the increased spatial cost uncertainty associated with denser graphs increases the difficulty for LADO to follow the ML policy, leading to a larger regret or cost increase compared to the ML policy. Moreover, as the competitiveness requirement becomes more stringent and LADO needs to stay closer to the expert, it is more difficult for LADO to closely follow the ML policy.



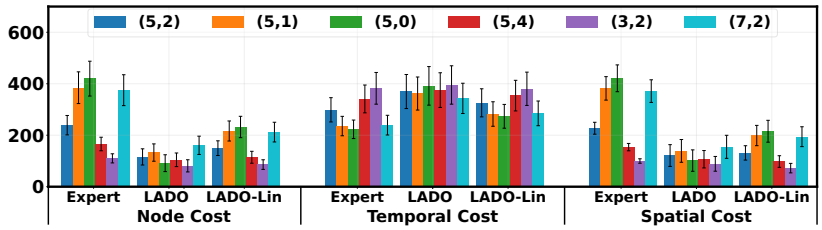
(a) Total cost of LADO

(b) Regret of LADO compared to ML

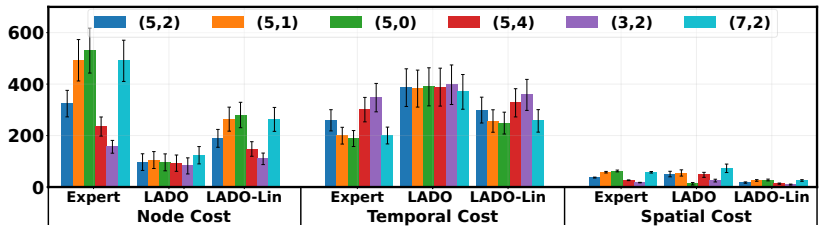
Fig. 5. Impact of graph sizes and competitiveness requirement λ on the overall cost of LADO, along with the additional cost (regret) incurred by the projection process compared to the ML policy. The overall cost and regret are normalized by the number of nodes for a consistent comparison across different graph sizes.

Impact of graph sizes: Next, we conduct a comprehensive comparison between the overall cost and regret of LADO (compared to the ML policy) over a wide range of graph sizes, ranging from 3 to 120 nodes. For consistency, all the graphs are fully connected. We normalize the overall costs and regrets by the number of nodes. As illustrated in Fig. 5, this normalization enables a meaningful comparison between algorithms over different graph sizes. Similar to our previous findings, the total

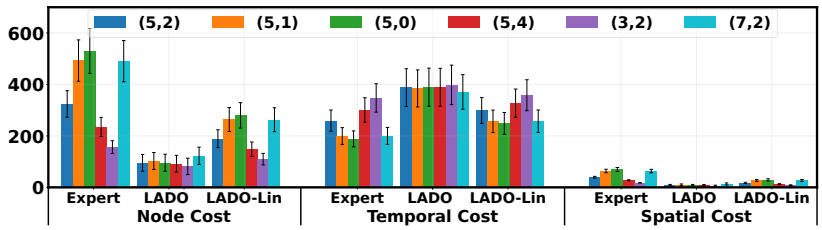
cost and regret of LADO compared to the ML policy decrease as the competitiveness requirement relaxes with greater λ, even for the largest graph. In fully-connected graphs, a larger graph implies a higher degree of node connectivity. As indicated by the term $\omega_v(\lambda, \tilde{\pi}, \pi^\dagger)$ in Theorem 5.2, a larger node degree makes it more difficult for LADO to follow the ML policy. Consequently, given a fixed competitiveness requirement λ, the larger graph sizes, the greater node degrees, and the higher regret of LADO compared to the ML policy as suggested by Theorem 5.2.



(a) Complete graph



(b) Star graph



(c) Linear chain graph

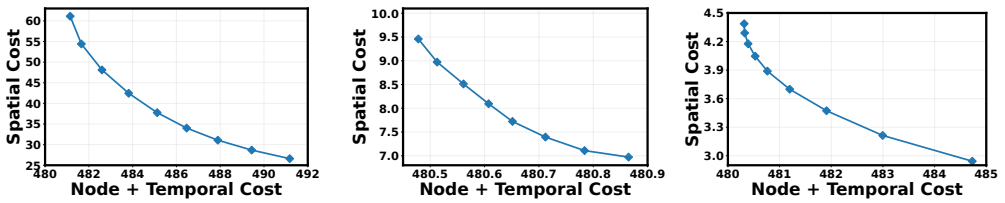
Fig. 6. Comparison of node, temporal, and spatial costs for LADO with varying weights (b, c) in the decentralized battery management formulation (shown as Eqn. (2)). By default, we consider a 15-node network and set the competitiveness requirement λ as 1. To enhance visual clarity, the temporal costs are scaled up by 10 to align with the scale of node and spatial costs. Results for LADO-Lin and Expert are also included for reference.

LADO with different weights for temporal and spatial costs: As shown in Eqn. (2), the total cost is parameterized by the weights for the temporal and spatial costs, denoted as (b, c) respectively. By normalizing the node cost weight to 1, the magnitudes of b and c directly reflect the relative importance of temporal and spatial decision smoothness versus reducing the node cost for achieving the desired state of charge at each battery node. It is crucial to note that these weights represent relative preferences rather than absolute cost scales. For instance, even if we set both b and c as 1, the temporal and spatial costs are necessarily equal to the node cost.

We compare the performance of LADO, LADO-Lin and Expert under a variety of weight combinations, as shown in Fig. 6. In this experiment, the ML algorithm is not fine-tuned based on the

weights of b and c for the temporal and spatial costs, and the other baseline algorithms are not affected by these weights. By keeping the weight of switching cost constant, a greater c prioritizes the spatial cost, thus leading to reduced spatial costs for all the algorithms. Conversely, a smaller c gives more emphasis to the temporal cost. Additionally, LADO exhibits less sensitivity to weight variations (b, c) compared to LADO-Lin and Expert, which is more evident in the complete graph. This robustness stems from LADO's design, where the expert advice constructs a robust action set. Consequently, LADO's spatial cost is less influenced by the Expert policy changes than the static linear combination in LADO-Lin.

In practice, the spatial cost parameter is adjusted to strike a balance between two competing objectives: the local performance of each node (measured by node and temporal costs) and the spatial consistency among connected nodes. For the graphs with heterogeneous nodes, these objectives may conflict. Prioritizing spatial consistency can make it more difficult to achieve optimal local performance for individual nodes compared to the scenario where each node operates independently. As shown in Fig. 7, by increasing the spatial cost parameter c, LADO enhances SoC consistency among connected battery units. To directly compare the actual SoC difference between battery units across different scenarios, spatial costs are evaluated using a constant parameter of $c = 1$. Consequently, for all the three graph topologies, we observe that the local costs (e.g. node and temporal costs) of individual nodes increase as the spatial difference decreases. Moreover, graphs with more spatial connections, such as the complete graph in Fig 7(a), exhibit a greater sensitivity to the spatial cost parameter c. This is because introducing additional spatial considerations between nodes amplifies the impact of c on local costs.



(a) Complete graph (b) Star graph (c) Linear chain graph

Fig. 7. The tradeoff curve between spatial cost and the sum of node and temporal cost of LADO by adjusting the coefficient c in spatial cost, which penalizes the SoC difference between connected battery units. The parameter b for the temporal cost is set as 1 by default.

D Proofs of Results in Section 3

We begin with a technical lemma.

Lemma D.1. *If the spatial cost is non-negative, convex and ℓ_S-smooth w,r,t the vector (x_v, x_u), then for any $\lambda > 0$, it holds that*

$$s_t^{(v,u)}(x_t^v, x_t^u) - (1+\lambda)s_t^{(v,u)}(x_t^{v,\dagger}, x_t^{u,\dagger}) \leq \frac{\ell_S}{2}(1+\frac{1}{\lambda})\left(\|x_t^v - x_t^{v,\dagger}\|^2 + \|x_t^u - x_t^{u,\dagger}\|^2\right). \tag{25}$$

Proof. By the definition of smoothness, we have

$$\begin{aligned}
&s_t^{(v,u)}(x_t^v, x_t^u)\\
\leq& s_t^{(v,u)}(x_t^{v,\dagger}, x_t^{u,\dagger}) + < \nabla s_t^{(v,u)}(x_t^{v,\dagger}, x_t^{u,\dagger}), (x_t^v - x_t^{v,\dagger}, x_t^u - x_t^{u,\dagger}) > + \frac{\ell_S}{2}\|(x_t^v, x_t^u) - (x_t^{v,\dagger}, x_t^{u,\dagger})\|^2\\
\leq& s_t^{(v,u)}(x_t^{v,\dagger}, x_t^{u,\dagger}) + \|\nabla s_t^{(v,u)}(x_t^{v,\dagger}, x_t^{u,\dagger})\| \cdot \|(x_t^v - x_t^{v,\dagger}, x_t^u - x_t^{u,\dagger})\| + \frac{\ell_S}{2}\|(x_t^v, x_t^u) - (x_t^{v,\dagger}, x_t^{u,\dagger})\|^2\\
\leq& s_t^{(v,u)}(x_t^{v,\dagger}, x_t^{u,\dagger}) + \frac{\lambda}{2\ell_S}\|\nabla s_t^{(v,u)}(x_t^{v,\dagger}, x_t^{u,\dagger})\|^2 + (1+\frac{1}{\lambda})\frac{\ell_S}{2}\|(x_t^v, x_t^u) - (x_t^{v,\dagger}, x_t^{u,\dagger})\|^2
\end{aligned} \tag{26}$$

The second inequality comes from the property of inner product. The third inequality is based on AM-QM inequality. Besides, if $(\hat{x}_t^v, \hat{x}_t^u)$ is a minimizer of the spatial cost, by Lemma 2.9 in [75], we have

$$s_t^{(v,u)}(x_t^{v,\dagger}, x_t^{u,\dagger}) \geq s_t^{(v,u)}(\hat{x}_t^v, \hat{x}_t^u) + 0 + \frac{1}{2\ell_S}\|\nabla s_t^{(v,u)}(x_t^{v,\dagger}, x_t^{u,\dagger})\|^2 \geq \frac{1}{2\ell_S}\|\nabla s_t^{(v,u)}(x_t^{v,\dagger}, x_t^{u,\dagger})\|^2 \tag{27}$$

By substituting Eqn (27) back to Eqn (26), we have

$$s_t^{(v,u)}(x_t^v, x_t^u) \leq (1+\lambda)\cdot s_t^{(v,u)}(x_t^{v,\dagger}, x_t^{u,\dagger}) + (1+\frac{1}{\lambda})\frac{\ell_S}{2}(\|(x_t^v - x_t^{v,\dagger}\|^2 + \|x_t^u - x_t^{u,\dagger}\|^2). \tag{28}$$

□

D.1 Proof of Theorem 3.1

Since LADO-Lin selects action as $x_t^v = \gamma\tilde{x}_t^v + (1-\gamma)x_t^{v,\dagger}$ at each round, by convexity of the global cost, we have

$$cost(\texttt{LADO-Lin}, g_{1:T}) \leq \gamma cost(\tilde{\pi}, g_{1:T}) + (1-\gamma)cost(\pi^\dagger, g_{1:T}). \tag{29}$$

By taking expectation on both sides, the cost of LADO-Lin is bounded by the first term in the min operator. Next, we prove the cost of LADO-Lin is also bounded by the second term in the min operator.

First, we can write the norm of the difference between the actions of LADO-Lin and expert $\pi^\dagger$ as

$$\|x_t^v - \tilde{x}_t^v\| = (1-\gamma)\|x_t^{v,\dagger} - \tilde{x}_t^v\|. \tag{30}$$

Based on Lemma D.1, for any $\lambda_2 > 0$, for any $v \in \mathcal{E}$, we have

$$\begin{aligned}
&\left(\sum_{\tau=1}^T f_\tau^v(x_\tau^v) + \sum_{\tau=1}^T c_\tau^v(x_\tau^v, x_{\tau-1}^v)\right) - (1+\lambda_2)\left(\sum_{\tau=1}^T f_\tau^v(\tilde{x}_\tau^v) + \sum_{\tau=1}^T c_\tau^v(\tilde{x}_\tau^v, \tilde{x}_{\tau-1}^v)\right)\\
\leq&(1+\frac{1}{\lambda_2})\left(\frac{\ell_f}{2}\sum_{\tau=1}^T\|x_\tau^v - \tilde{x}_\tau^v\|^2 + \frac{\ell_T}{2}\sum_{\tau=1}^T\|x_\tau^v - \tilde{x}_\tau^v\|^2 + \frac{\ell_T}{2}\sum_{\tau=0}^{T-1}\|x_\tau^v - \tilde{x}_\tau^v\|^2\right)\\
\leq&(1+\frac{1}{\lambda_2})\frac{\ell_f + 2\cdot\ell_T}{2}\sum_{\tau=1}^T\|x_\tau^v - \tilde{x}_\tau^v\|^2 \leq (1-\gamma)(1+\frac{1}{\lambda_2})\frac{\ell_f + 2\cdot\ell_T}{2}\sum_{\tau=1}^T\|x_t^{v,\dagger} - \tilde{x}_t^v\|^2
\end{aligned} \tag{31}$$

By summing up the spacial costs over time, we have

$$\begin{aligned}
&\sum_{(v,u)\in\mathcal{E}}\sum_{\tau=1}^{T} s_\tau^{(v,u)}(x_\tau^v, x_\tau^u) - (1+\lambda_2)\sum_{(v,u)\in\mathcal{E}}\sum_{\tau=1}^{T} s_\tau^{(v,u)}(\tilde{x}_\tau^v, \tilde{x}_\tau^u)\\
\le&(1+\frac{1}{\lambda_2})\frac{\ell_S}{2}\sum_{(v,u)\in\mathcal{E}}\sum_{\tau=1}^{T}\left(\|x_\tau^v-\tilde{x}_\tau^v\|^2+\|x_\tau^v-\tilde{x}_\tau^v\|^2\right) \qquad (32)\\
=&(1+\frac{1}{\lambda_2})\sum_{v\in\mathcal{V}}\frac{D_v\cdot\ell_S}{2}\sum_{\tau=1}^{T}\|x_\tau^v-\tilde{x}_\tau^v\|^2 \le (1-\gamma)^2(1+\frac{1}{\lambda_2})\sum_{v\in\mathcal{V}}\frac{D_v\cdot\ell_S}{2}\sum_{\tau=1}^{T}\|x_t^{v,\dagger}-\tilde{x}_t^v\|^2
\end{aligned}$$

By adding Eqn (32) to Eqn (31), we can bound the sum cost error as

$$cost(\texttt{LADO-Lin}) - (1+\lambda_2)cost(\tilde{\pi}) \le (1-\gamma)^2(1+\frac{1}{\lambda_2})\sum_{v\in\mathcal{V}}\frac{\ell_f+2\cdot\ell_T+D_v\cdot\ell_S}{2}\sum_{\tau=1}^{T}\|x_t^{v,\dagger}-\tilde{x}_t^v\|^2 \qquad (33)$$

By taking expectation on both sides, we have

$$AVG(\texttt{LADO-Lin}) - (1+\lambda_2)AVG(\tilde{\pi}) \le (1-\gamma)^2(1+\frac{1}{\lambda_2})\mathbb{E}_{g_{1:T}}\left[\sum_{v\in\mathcal{V}}\frac{\ell_f+2\cdot\ell_T+D_v\cdot\ell_S}{2}\sum_{\tau=1}^{T}\|x_t^{v,\dagger}-\tilde{x}_t^v\|^2\right] \qquad (34)$$

By optimally setting $\lambda_2 = \sqrt{(1-\gamma)^2\mathbb{E}_{g_{1:T}}\left[\sum_{v\in\mathcal{V}}\frac{\ell_f+2\cdot\ell_T+D_v\cdot\ell_S}{2}\sum_{\tau=1}^{T}\|x_t^{v,\dagger}-\tilde{x}_t^v\|^2\right]\frac{1}{AVG(\tilde{\pi})}}$, we can bound the sum cost as

$$cost(\texttt{LADO-Lin}) \le \left(\sqrt{AVG(\tilde{\pi})} + (1-\gamma)\sqrt{\mathbb{E}_{g_{1:T}}\left[\sum_{t=1}^{T}\sum_{v\in\mathcal{V}}\frac{\ell_f+2\ell_T+D_v\ell_S}{2}\|\tilde{x}_t^v - x_t^{v,\dagger}\|^2\right]}\right)^2 \qquad (35)$$

D.2 Proof of Proposition 3.2

If LADO-Lin satisfies the λ−competitiveness constraint, we have

$$cost(\texttt{LADO-Lin}) \le (1+\lambda)cost(\pi^\dagger) \le (1+\lambda)\rho_{\pi^\dagger}cost(\pi^*) \qquad (36)$$

Since the cost function $cost$ is β−strongly convex, the gradient of $cost$ at x^* is $\nabla cost(x^*) = 0$ and it holds that

$$\begin{aligned}
cost(\texttt{LADO-Lin}) &\ge cost(\pi^*) + \frac{\beta}{2}\|\gamma\tilde{x}_t + (1-\gamma)x_t^\dagger - x^*\|^2\\
&= cost(\pi^*) + \frac{\beta}{2}\|\gamma(\tilde{x}_t - x^*) + (1-\gamma)(x_t^\dagger - x^*)\|^2
\end{aligned} \qquad (37)$$

Substituting (37) to (36), we have

$$\frac{\beta}{2}\|\gamma(\tilde{x}_t - x^*) + (1-\gamma)(x_t^\dagger - x^*)\|^2 \le ((1+\lambda)\rho_{\pi^\dagger} - 1)\, cost(\pi^*). \qquad (38)$$

By taking the squared root for both sides and applying triangle inequality for the left hand side, it holds that

$$\|\gamma(\tilde{x}_t - x^*)\| - \|(1-\gamma)(x_t^\dagger - x^*)\| \le \sqrt{\frac{2}{\beta}((1+\lambda)\rho_{\pi^\dagger} - 1)\, cost(cost(\pi^*)}. \qquad (39)$$

Since the cost function is $\beta-$strongly convex, we have

$$cost(\pi^*) + \frac{\beta}{2}\|x^\dagger - x^*\|^2 \leq cost(x^\dagger) \leq \rho_{\pi^\dagger} cost(\pi^*). \tag{40}$$

Substituting (40) into (39), we have

$$\|\tilde{x}_t - x^*\| \leq \left(\frac{1-\gamma}{\gamma}\sqrt{(\rho_{\pi^\dagger}-1)\frac{2}{\beta}} + \frac{1}{\gamma}\sqrt{\frac{2}{\beta}\left((1+\lambda)\rho_{\pi^\dagger}-1\right)}\right)\sqrt{cost(\pi^*)}. \tag{41}$$

The proposition is proved by moving items in the above inequality.

E Proof of Robustness in Theorem 5.1

To prove Theorem 5.1, the key point is to guarantee the robust action set (17) is non-empty. We will prove this through induction. For $t = 1$, it is obvious that $x_1^v = x_1^{v,\dagger}$ satisfies the constraint. We assume that the robustness constraint is satisfied up to time step $t-1$, which is

$$\begin{aligned}
&\sum_{\tau=1}^{t-1} f_\tau^v(x_\tau^v) + \sum_{\tau=1}^{t-2}\sum_{(v,u)\in\mathcal{E}} \kappa_\tau^{(v,u)} \cdot s_\tau^{(v,u)}(x_\tau^v, x_\tau^u) + \frac{\ell_T + \ell_S \cdot D_v}{2}(1+\frac{1}{\lambda_0}) \cdot \|x_{t-1}^v - x_{t-1}^{v,\dagger}\|^2 \\
&+ \sum_{\tau=1}^{t-1} c_\tau^v(x_\tau^v, x_{\tau-1}^v) \leq (1+\lambda)\left(\sum_{\tau=1}^{t-1} f_\tau^v(x_\tau^{v,\dagger}) + \sum_{\tau=1}^{t-1} c_\tau^v(x_\tau^{v,\dagger}, x_{\tau-1}^{v,\dagger}) + \sum_{\tau=1}^{t-2}\sum_{(v,u)\in\mathcal{E}} \kappa_\tau^{(v,u)} \cdot s_\tau^{(v,u)}(x_\tau^{v,\dagger}, x_\tau^{u,\dagger})\right)
\end{aligned} \tag{42}$$

Based on Lemma D.1 and $\kappa_{t-1}^{(v,u)} = \frac{\|x_{t-1}^v - x_{t-1}^{v,\dagger}\|^2}{\|x_{t-1}^v - x_{t-1}^{v,\dagger}\|^2 + \|x_{t-1}^u - x_{t-1}^{u,\dagger}\|^2}$, we have

$$\kappa_{t-1}^{(v,u)} \cdot \left(s_{t-1}^{(v,u)}(x_{t-1}^v, x_{t-1}^u) - (1+\lambda)s_{t-1}^{(v,u)}(x_{t-1}^{v,\dagger}, x_{t-1}^{u,\dagger})\right) \leq \frac{\ell_S}{2}(1+\frac{1}{\lambda})\|x_{t-1}^v - x_{t-1}^{v,\dagger}\|^2 \tag{43}$$

For time step t, if we choose $x_t^v = x_t^{v,\dagger}$, by the smoothness assumption, we have

$$c_t^v(x_t^{v,\dagger}, x_{t-1}^v) - (1+\lambda)c_t^v(x_t^{v,\dagger}, x_{t-1}^{v,\dagger}) \leq \frac{\ell_T}{2}(1+\frac{1}{\lambda})\|x_{t-1}^v - x_{t-1}^{v,\dagger}\|^2 \tag{44}$$

Since the node cost is non-negative, by (43) and (44), we have

$$\begin{aligned}
&f_t^v(x_t^{v,\dagger}) + \sum_{(v,u)\in\mathcal{E}} \kappa_{t-1}^{(v,u)} \cdot s_{t-1}^{(v,u)}(x_{t-1}^v, x_{t-1}^u) + c_t^v(x_t^{v,\dagger}, x_{t-1}^v) - \frac{\ell_T + \ell_S \cdot D_v}{2}(1+\frac{1}{\lambda_0})\|x_{t-1}^v - x_{t-1}^{v,\dagger}\|^2 \\
&- (1+\lambda)\left(f_t^v(x_t^{v,\dagger}) + c_t^v(x_t^{v,\dagger}, x_{t-1}^{v,\dagger}) + \sum_{(v,u)\in\mathcal{E}} \kappa_{t-1}^{(v,u)} \cdot s_{t-1}^{(v,u)}(x_{t-1}^{v,\dagger}, x_{t-1}^{u,\dagger})\right) \\
\leq &\frac{\ell_T + \ell_S \cdot D_v}{2}(1+\frac{1}{\lambda})\|x_{t-1}^v - x_{t-1}^{v,\dagger}\|^2 - \frac{\ell_T + \ell_S \cdot D_v}{2}(1+\frac{1}{\lambda_0})\|x_{t-1}^v - x_{t-1}^{v,\dagger}\|^2 \\
\leq &0,
\end{aligned} \tag{45}$$

where the last inequality holds by $\lambda \geq \lambda_0$. By adding Eqn (45) back to Eqn (42) and moving items, we recover the robustness constraint for time step t if $x_t^v = x_t^{v,\dagger}$,

$$\sum_{\tau=1}^{t} f_\tau^v(x_\tau^v) + \sum_{\tau=1}^{t-1} \sum_{(v,u)\in\mathcal{E}} \kappa_\tau^{(v,u)} \cdot s_\tau^{(v,u)}(x_\tau^v, x_\tau^u) + \sum_{\tau=1}^{t} c_\tau^v(x_\tau^v, x_{\tau-1}^v) + \frac{\ell_T + \ell_S \cdot D_v}{2}(1 + \frac{1}{\lambda_0})\|x_t^v - x_t^{v,\dagger}\|^2$$
$$\leq (1+\lambda)\Big(\sum_{\tau=1}^{t} f_\tau^v(x_\tau^{v,\dagger}) + \sum_{\tau=1}^{t} c_\tau^v(x_\tau^{v,\dagger}, x_{\tau-1}^{v,\dagger}) + \sum_{\tau=1}^{t-1} \sum_{(v,u)\in\mathcal{E}} \kappa_\tau^{(v,u)} \cdot s_\tau^{(v,u)}(x_\tau^{v,\dagger}, x_\tau^{u,\dagger})\Big) \tag{46}$$

In other words, the expert's action $x_t^{v,\dagger}$ is always an action in the corresponding robust action set (17). Thus the robust action set is non-empty.

Since $\kappa_t^{(v,u)} + \kappa_t^{(u,v)} = 1$ holds for $(v,u) \in \mathcal{E}$, if all the nodes select actions from the robust action set (17) at each step, we can guarantee that $cost(\texttt{LADO}, g_{1:T}) \leq (1+\lambda) \cdot cost(\pi^\dagger, g_{1:T})$ is satisfied.

F Proof of Average Cost Bounds and Robust-Consistency of LADO

F.1 Proof of Theorem 5.2

We begin by stating and proving a technical lemma and then move to the proof of Theorem 5.2.

Lemma F.1. *We denote the actual actions from* LADO *as* $x_{1:T}^v = (x_1^v, \cdots, x_T^v)$*, the squared distance between actual action and ML advice is bounded by*

$$\sum_{t=1}^{T} \|x_t^v - \tilde{x}_t^v\|^2 \leq \sum_{t=1}^{T} \left[\|\tilde{x}_t^v - x_t^{v,\dagger}\|^2 - \frac{\lambda - \lambda_0}{1 + \frac{1}{\lambda_0}} \cdot \frac{2}{\ell_f + 2\cdot\ell_T + \ell_S \cdot D_v} \cdot cost_{v,t}^\dagger\right]^+$$

where λ, λ_0, D, *and* ρ *are defined in Theorem 5.2.*

Proof. To prove this lemma, we first construct a sufficient condition to satisfy the original constraint in Eqn (13). Then we prove a distance bound in this sufficient condition, where the bound still holds for the original problem.

At time step t, we know the constraint in time step $t-1$ is already satisfied, so we obtain the following sufficient condition of the satisfaction of (13) as

$$f_t^v(x_t^v) + \sum_{(v,u)\in\mathcal{E}} \kappa_{t-1}^{(v,u)} \cdot s_{t-1}^{(v,u)}(x_{t-1}^v, x_{t-1}^u) + \frac{\ell_T + \ell_S \cdot D_v}{2}(1 + \frac{1}{\lambda_0})\left(\|x_t^v - x_t^{v,\dagger}\|^2 - \|x_{t-1}^v - x_{t-1}^{v,\dagger}\|^2\right)$$
$$+c_t^v(x_t^v, x_{t-1}^v) \leq (1+\lambda)\Big(f_t^v(x_t^{v,\dagger}) + c_t^v(x_t^{v,\dagger}, x_{t-1}^{v,\dagger}) + \sum_{(v,u)\in\mathcal{E}} \kappa_{t-1}^{(v,u)} \cdot s_{t-1}^{(v,u)}(x_{t-1}^{v,\dagger}, x_{t-1}^{u,\dagger})\Big) \tag{47}$$

With the convexity and smoothness assumptions, we have

$$c_t^v(x_t^{v,\dagger}, x_{t-1}^v) - (1+\lambda_0)c_t^v(x_t^{v,\dagger}, x_{t-1}^{v,\dagger}) \leq \frac{\ell_T}{2}(1 + \frac{1}{\lambda_0})\left(\|x_t^v - x_t^{v,\dagger}\|^2 + \|x_{t-1}^v - x_{t-1}^{v,\dagger}\|^2\right)$$
$$f_t^v(x_t^v) - (1+\lambda_0)f_t^v(x_t^{v,\dagger}) \leq \frac{\ell_f}{2}(1 + \frac{1}{\lambda_0})\|x_t^v - x_t^{v,\dagger}\|^2 \tag{48}$$

Then a sufficient condition that (47) holds becomes

$$\sum_{(v,u)\in\mathcal{E}} \kappa_{t-1}^{(v,u)} \cdot s_{t-1}^{(v,u)}(x_{t-1}^v, x_{t-1}^u) + (1+\frac{1}{\lambda_0})\left(\frac{\ell_f + 2\cdot\ell_T + \ell_S\cdot D_v}{2}\|x_t^v - x_t^{v,\dagger}\|^2 - \frac{\ell_S\cdot D_v}{2}\|x_{t-1}^v - x_{t-1}^{v,\dagger}\|^2\right)$$
$$\leq (\lambda-\lambda_0)\left(f_t^v(x_t^{v,\dagger}) + c_t^v(x_t^{v,\dagger}, x_{t-1}^{v,\dagger})\right) + (1+\lambda)\sum_{(v,u)\in\mathcal{E}} \kappa_{t-1}^{(v,u)} \cdot s_{t-1}^{(v,u)}(x_{t-1}^{v,\dagger}, x_{t-1}^{u,\dagger}) \tag{49}$$

From Eqn (43), we can further cancel out the spatial costs and get the sufficient condition of Eqn (49), shown as below

$$\begin{aligned}&(1+\frac{1}{\lambda_0})\left(\frac{\ell_f + 2\cdot\ell_T + \ell_S\cdot D_v}{2}\|x_t^v - x_t^{v,\dagger}\|^2\right)\\ \leq\ &(\lambda-\lambda_0)\left(f_t^v(x_t^{v,\dagger}) + c_t^v(x_t^{v,\dagger}, x_{t-1}^{v,\dagger}) + \sum_{(v,u)\in\mathcal{E}} \kappa_{t-1}^{(v,u)} \cdot s_{t-1}^{(v,u)}(x_{t-1}^{v,\dagger}, x_{t-1}^{u,\dagger})\right)\end{aligned} \tag{50}$$

For the expert policy $\pi^\dagger$ at time t, we define the sum of node cost and temporal cost as $\text{cost}_{v,t}^\dagger = f_t^v(x_t^{v,\dagger}) + c_t^v(x_t^{v,\dagger}, x_{t-1}^{v,\dagger})$. Therefore, a sufficient condition of Eqn (49) is

$$\|x_t^v - x_t^{v,\dagger}\|^2 \leq \frac{\lambda-\lambda_0}{1+\frac{1}{\lambda_0}} \cdot \frac{2}{\ell_f + 2\cdot\ell_T + \ell_S\cdot D_v} \cdot \text{cost}_{v,t}^\dagger \tag{51}$$

By summarizing the sufficient condition in Eqn (47-51), we conclude that for any x_t^v satisfying Eqn (51) must satisfy the original constraint Eqn (13). If the ML advice $\tilde{x}_t^v$ satisfies the inequality in Eqn (51), we can completely follow ML advice without any modification for node v. Otherwise, we construct a $\hat{x}_t^v$ achieves equity in Eqn (51) and satisfies $\|\hat{x}_t^v - \tilde{x}_t^v\| = \|\tilde{x}_t^v - x_t^{v,\dagger}\| - \|\hat{x}_t^v - \tilde{x}_t^v\|$. Therefore, the distance between the constructed action $\hat{x}_t^v$ and ML advice is given by

$$\|\hat{x}_t^v - \tilde{x}_t^v\| = \left[\|\tilde{x}_t^v - x_t^{v,\dagger}\| - \sqrt{\frac{\lambda-\lambda_0}{1+\frac{1}{\lambda_0}} \cdot \frac{2}{\ell_f + 2\cdot\ell_T + \ell_S\cdot D_v} \cdot \text{cost}_{v,t}^\dagger}\right]^+ \tag{52}$$

Since x_t^v is obtained by minimizing its distance to ML action $\tilde{x}_t^v$ under the original constraint in Eqn (13), it's obvious that $\|x_t^v - \tilde{x}_t^v\| \leq \|\hat{x}_t^v - \tilde{x}_t^v\|$. Besides, we have the following inequality

$$\begin{aligned}&\left(\left[\|\tilde{x}_t^v - x_t^{v,\dagger}\| - \sqrt{\frac{\lambda-\lambda_0}{1+\frac{1}{\lambda_0}} \cdot \frac{2}{\ell_f + 2\cdot\ell_T + \ell_S\cdot D_v} \cdot \text{cost}_{v,t}^\dagger}\right]^+\right)^2\\ \leq\ &\left[\|\tilde{x}_t^v - x_t^{v,\dagger}\|^2 - \frac{\lambda-\lambda_0}{1+\frac{1}{\lambda_0}} \cdot \frac{2}{\ell_f + 2\cdot\ell_T + \ell_S\cdot D_v} \cdot \text{cost}_{v,t}^\dagger\right]^+\end{aligned} \tag{53}$$

By summing up the inequalities over time, we complete the proof. □

Proof of Theorem 5.2. Based on Lemma D.1, for any $\lambda_2 > 0$ we have

$$\begin{aligned}&\left(\sum_{\tau=1}^T f_\tau^v(x_\tau^v) + \sum_{\tau=1}^T c_\tau^v(x_\tau^v, x_{\tau-1}^v)\right) - (1+\lambda_2)\left(\sum_{\tau=1}^T f_\tau^v(\tilde{x}_\tau^v) + \sum_{\tau=1}^T c_\tau^v(\tilde{x}_\tau^v, \tilde{x}_{\tau-1}^v)\right)\\ \leq&(1+\frac{1}{\lambda_2})\left(\frac{\ell_f}{2}\sum_{\tau=1}^T\|x_\tau^v - \tilde{x}_\tau^v\|^2 + \frac{\ell_T}{2}\sum_{\tau=1}^T\|x_\tau^v - \tilde{x}_\tau^v\|^2 + \frac{\ell_T}{2}\sum_{\tau=0}^{T-1}\|x_\tau^v - \tilde{x}_\tau^v\|^2\right)\\ \leq&(1+\frac{1}{\lambda_2})\frac{\ell_f + 2\cdot\ell_T}{2}\sum_{\tau=1}^T\|x_\tau^v - \tilde{x}_\tau^v\|^2\end{aligned} \tag{54}$$

By summing up the spacial costs over time, we have

$$\begin{aligned}
&\sum_{(v,u)\in\mathcal{E}}\sum_{\tau=1}^{T} s_\tau^{(v,u)}(x_\tau^v, x_\tau^u) - (1+\lambda_2)\sum_{(v,u)\in\mathcal{E}}\sum_{\tau=1}^{T} s_\tau^{(v,u)}(\tilde{x}_\tau^v, \tilde{x}_\tau^u)\\
\le&(1+\frac{1}{\lambda_2})\frac{\ell_S}{2}\sum_{(v,u)\in\mathcal{E}}\sum_{\tau=1}^{T}\left(\|x_\tau^v-\tilde{x}_\tau^v\|^2+\|x_\tau^v-\tilde{x}_\tau^v\|^2\right) \qquad (55)\\
=&(1+\frac{1}{\lambda_2})\sum_{v\in\mathcal{V}}\frac{D_v\cdot\ell_S}{2}\sum_{\tau=1}^{T}\|x_\tau^v-\tilde{x}_\tau^v\|^2
\end{aligned}$$

By adding Eqn (54) and Eqn (55), we can bound the cost error of node v as

$$\text{cost}(x_{1:T}) - (1+\lambda_2)\text{cost}(\tilde{x}_{1:T}) \le (1+\frac{1}{\lambda_2})\sum_{v\in\mathcal{V}}\frac{\ell_f+2\cdot\ell_T+D_v\cdot\ell_S}{2}\sum_{\tau=1}^{T}\|x_\tau^v-\tilde{x}_\tau^v\|^2, \qquad (56)$$

where $D_{\max}=\max_{v\in\mathcal{V}} D_v$ is the maximum degree of nodes. By substituting Lemma F.1 into Eqn (56), we have

$$\text{cost}(x_{1:T}) - (1+\lambda_2)\text{cost}(\tilde{x}_{1:T}) \le (1+\frac{1}{\lambda_2})\sum_{v\in\mathcal{V}}\sum_{t=1}^{T}\left[\frac{\ell_f+2\cdot\ell_T+\ell_S\cdot D_v}{2}\|\tilde{x}_t^v-x_t^{v,\dagger}\|^2-\frac{\lambda-\lambda_0}{1+\frac{1}{\lambda_0}}\cdot\text{cost}_{v,t}^{\dagger}\right]^+ \qquad (57)$$

where the right-hand side captures the cost increase brought by the robustification process, which is minimized by setting $\lambda_0=\sqrt{1+\lambda}-1$. By taking the expectation of Eqn (57) over the context distribution $\mathbb{P}_{g_{1:T}}$, we have

$$AVG(\texttt{LADO}(\tilde{\pi})) - (1+\lambda_2)AVG(\tilde{\pi}) \le (1+\frac{1}{\lambda_2})\sum_{v\in\mathcal{V}}\omega_v(\lambda,\tilde{\pi},\pi^\dagger) \qquad (58)$$

where $\omega_v(\lambda,\tilde{\pi},\pi^\dagger)=\mathbb{E}_{g_{1:T}}\left\{\sum_{t=1}^{T}\left[\frac{\ell_f+2\cdot\ell_T+\ell_S\cdot D_v}{2}\|\tilde{x}_t^v-x_t^{v,\dagger}\|^2-(\sqrt{1+\lambda}-1)^2\cdot\text{cost}_{v,t}^{\dagger}\right]^+\right\}$. By optimally setting $\lambda_2=\sqrt{\frac{\sum_{v\in\mathcal{V}}\omega_v(\lambda,\tilde{\pi},\pi^\dagger)}{AVG(\tilde{\pi})}}$

□

F.2 Proof of Corollary A.1

To prove the Corollary A.1, we first show the robustness is also guaranteed under the setting of directed graphs. Secondly, we will provide the upper bound of the distance between actual action taken by LADO and ML advice. Finally, we translate the action distance to the cost increase associated with the projection. Since most of the proof steps are similar to the problem with undirected graphs, we only highlight the difference here.

Based on Lemma D.1 and $\kappa_{t-1}^{(v,u)}=\frac{\|x_{t-1}^v-x_{t-1}^{v,\dagger}\|^2}{\|x_{t-1}^v-x_{t-1}^{v,\dagger}\|^2+\|x_{t-1}^u-x_{t-1}^{u,\dagger}\|^2}$, we have

$$\sum_{(v,u)\in\mathcal{E}}\kappa_{t-1}^{(v,u)}\cdot\left(s_{t-1}^{(v,u)}(x_{t-1}^v,x_{t-1}^u)-(1+\lambda)s_{t-1}^{(v,u)}(x_{t-1}^{v,\dagger},x_{t-1}^{u,\dagger})\right)\le\frac{D_v^{out}\ell_S}{2}(1+\frac{1}{\lambda})\|x_{t-1}^v-x_{t-1}^{v,\dagger}\|^2 \qquad (59)$$

$$\sum_{(u,v)\in\mathcal{E}}\kappa_{t-1}^{(u,v)}\cdot\left(s_{t-1}^{(u,v)}(x_{t-1}^u,x_{t-1}^v)-(1+\lambda)s_{t-1}^{(u,v)}(x_{t-1}^{u,\dagger},x_{t-1}^{v,\dagger})\right)\le\frac{D_v^{in}\ell_S}{2}(1+\frac{1}{\lambda})\|x_{t-1}^v-x_{t-1}^{v,\dagger}\|^2 \qquad (60)$$

By plugging the inequality back to Eqn (45), we can prove that if action constraint in Eqn (19) is satisfied up to time $t-1$, the expert action $x_t^{v,\dagger}$ is a feasible solution for the directed graph $\mathcal{G}$. In other words, the action set constructed by the constraint in Eqn (19) is nonempty for all $t \in [1, T]$.

If the constraint in Eqn. (19) is satisfied at time $t-1$, a sufficient condition of the constraint at time t is formulated as

$$(1+\frac{1}{\lambda_0})\left(\frac{\ell_f + 2\cdot\ell_T + \ell_S\cdot(D_v^{in}+D_v^{out})}{2}\|x_t^v - x_t^{v,\dagger}\|^2\right) \le (\lambda-\lambda_0)\Big(f_t^v(x_t^{v,\dagger}) + c_t^v(x_t^{v,\dagger}, x_{t-1}^{v,\dagger}) + \sum_{(v,u)\in\mathcal{E}} \kappa_{t-1}^{(v,u)}\cdot s_{t-1}^{(v,u)}(x_{t-1}^{v,\dagger}, x_{t-1}^{u,\dagger}) + \sum_{(u,v)\in\mathcal{E}} \kappa_{t-1}^{(u,v)}\cdot s_{t-1}^{(u,v)}(x_{t-1}^{u,\dagger}, x_{t-1}^{v,\dagger})\Big) \tag{61}$$

By defining $cost_{v,t}^{\dagger} = f_t^v(x_t^{v,\dagger}) + c_t^v(x_t^{v,\dagger}, x_{t-1}^{v,\dagger})$ as the sum of expert's node cost and temporal cost for node v at time t, a sufficient condition of Eqn (61) is

$$\|x_t^v - x_t^{v,\dagger}\|^2 \le \frac{\lambda-\lambda_0}{1+\frac{1}{\lambda_0}}\cdot\frac{2}{\ell_f + 2\cdot\ell_T + \ell_S\cdot(D_v^{in}+D_v^{out})}\cdot \text{cost}_{v,t}^{\dagger} \tag{62}$$

By substituting the action distance to Eqn (53) and summing up the distance over the horizon t and the entire graph, we obtain the total action distance between LADO and ML policy in the directed graph. Then we finish the proof by translating the distance to the cost increase ncaccording to the smoothness assumption of these three costs (Assumption 2.1 - 2.3)

F.3 Proof of Theorem 5.3

Denote $\ell = \frac{\ell_f + 2\cdot\ell_T + \ell_S\cdot D_{\max}}{2}$. By (57), when ML gives the offline-optimal actions, i.e. $\tilde{x}_{1:T} = x_{1:T}^*$, we have for any sequence $g_{1:T}$,

$$\begin{aligned}&\text{cost}(x_{1:T}) - (1+\lambda_2)\text{cost}(x_{1:T}^*)\\ \le&(1+\frac{1}{\lambda_2})\ell\sum_{v\in\mathcal{V}}\sum_{t=1}^{T}\left[\|x_t^{v,*} - x_t^{v,\dagger}\|^2 - \frac{1}{\ell}\cdot(\sqrt{1+\lambda}-1)^2\cdot\text{cost}_{v,t}^{\dagger}\right]^+\end{aligned} \tag{63}$$

By optimally setting λ_2, we have

$$\text{cost}(x_{1:T}) \le \left(\sqrt{\text{cost}(x_{1:T}^*)} + \sqrt{\ell\sum_{v\in\mathcal{V}}\sum_{t=1}^{T}\left[\|x_t^{v,*} - x_t^{v,\dagger}\|^2 - \frac{1}{\ell}\cdot(\sqrt{1+\lambda}-1)^2\cdot\text{cost}_{v,t}^{\dagger}\right]^+}\right)^2 \tag{64}$$

which translates to a competitive ratio of

$$\rho_{\text{LADO}} \le \min\left\{\left[1+\sqrt{\max_{g_{1:T}\in\mathcal{G}}\frac{\sum_{v\in\mathcal{V}}\sum_{t=1}^{T}\left[\ell\|x_t^{v,*} - x_t^{v,\dagger}\|^2 - (\sqrt{1+\lambda}-1)^2\cdot cost_{v,t}^{\dagger}\right]^+}{cost(x_{1:T}^*, g_{1:T})}}\right]^2, (1+\lambda)\rho_{\pi^\dagger}\right\} \tag{65}$$

.

By $\beta-$strongly convexity of the cost function, we have $\nabla cost(x_{1:T}^*) = 0$ and

$$cost(\pi^\dagger) \ge cost(\pi^*) + \frac{\beta}{2}\sum_{v\in\mathcal{V}}\sum_{t=1}^{T}\|x_t^{v,\dagger} - x_t^{v,*}\|^2. \tag{66}$$

Thus, the competitive ratio can be simplified as

$$\begin{aligned}\rho_{\texttt{LADO}} &\leq \left(1+\sqrt{\max_{g_{1:T}} 2\ell \frac{\sum_{v\in\mathcal{V}}\sum_{t=1}^{T}\|x_t^{v,\dagger}-x_t^{v,*}\|^2}{cost(x^*_{1:T},g_{1:T})}}\right)^2 \\ &\leq \left(1+\sqrt{\max_{g_{1:T}} \frac{4\ell}{\beta}\frac{cost(x^\dagger_{1:T},g_{1:T})-cost(x^*_{1:T},g_{1:T})}{cost(x^*_{1:T},g_{1:T})}}\right)^2 \\ &\leq \left(1+2\sqrt{\frac{\ell}{\beta}\cdot(\rho_{\pi^\dagger}-1)}\right)^2.\end{aligned} \tag{67}$$

F.4 Proof of Corollary B.1

In Corollary B.1, we assume that π°_λ in Eqn. (22) is used in LADO. To bound the average cost of $\texttt{LADO}(\tilde{\pi}^\circ_\lambda)$, we construct a policy that satisfies the constraint (13) for each step in each sequence. Then the average cost bound of the constructed policy is also the average cost upper bound of $\texttt{LADO}(\tilde{\pi}^\circ_\lambda)$ since $\texttt{LADO}(\tilde{\pi}^\circ_\lambda)$ is the policy that minimizes average cost while satisfying the constraint (13) for each step in each sequence if we assume that the ML model can represent any policy. The feasible policy is constructed as $\hat{\pi}=(1-\alpha)\pi^\dagger+\alpha\tilde{\pi}^*$ which gives action $\hat{x}_t^v=(1-\alpha)x_t^{v,\dagger}+\alpha\tilde{x}_t^v, \alpha\in[0,1]$, where $x_t^\dagger$, $\tilde{x}_t$ denotes expert action and the prediction from projection-unaware ML model $\tilde{\pi}^*$, respectively. We need to find the α that guarantees the satisfaction of the constraint (11). To do that, we rewrite the constraint as

$$\begin{aligned}&\sum_{\tau=1}^{t}\left(f_\tau^v(\hat{x}_\tau^v)-(1+\lambda_0)f_\tau^v(x_\tau^\dagger)\right)+\sum_{\tau=1}^{t}\left(c^v(\hat{x}_\tau^v,\hat{x}_{\tau-1}^v)-(1+\lambda_0)c^v(x_\tau^{v,\dagger},x_{\tau-1}^{v,\dagger})\right)+\sum_{\tau=1}^{t-1}\sum_{(v,u)\in\mathcal{E}}\kappa_\tau^{(v,u)}\cdot \\ &\left(s_\tau^{(v,u)}(\hat{x}_\tau^v,\hat{x}_\tau^u)-(1+\lambda_0)s_\tau^{(v,u)}(x_\tau^{v,\dagger},x_\tau^{u,\dagger})\right)+(1+\frac{1}{\lambda_0})\frac{\ell_T+D_v\cdot\ell_S}{2}\|x_t^v-x_t^{v,\dagger}\|^2 \\ \leq&(\lambda-\lambda_0)\left(\sum_{\tau=1}^{t}f_\tau^v(x_\tau^{v,\dagger})+\sum_{\tau=1}^{t}c^v(x_\tau^{v,\dagger},x_{\tau-1}^{v,\dagger})+\sum_{\tau=1}^{t-1}\sum_{(v,u)\in\mathcal{E}}\kappa_\tau^{(v,u)}s_\tau^{(v,u)}(x_\tau^{v,\dagger},x_\tau^{u,\dagger})\right),\forall t\in[1,T]\end{aligned} \tag{68}$$

Based on the smoothness assumption, we have

$$\begin{aligned}f_t^v(\hat{x}_t^v)-(1+\lambda_0)f_t^v(x_t^{v,\dagger}) \leq&(1+\frac{1}{\lambda_0})\frac{\ell_f}{2}\|\hat{x}_t^v-x_t^{v,\dagger}\|^2 \\ c^v(\hat{x}_t^v,\hat{x}_{t-1}^v)-(1+\lambda_0)c^v(x_t^{v,\dagger},x_{t-1}^{v,\dagger}) \leq&(1+\frac{1}{\lambda_0})\frac{\ell_T}{2}(\|\hat{x}_t^v-x_t^{v,\dagger}\|^2+\|\hat{x}_{t-1}^v-x_{t-1}^{v,\dagger}\|^2) \\ \kappa_t^{(v,u)}\left(s_t^{(v,u)}(\hat{x}_t^v,\hat{x}_t^u)-(1+\lambda_0)s_t^{(v,u)}(x_t^{v,\dagger},x_t^{u,\dagger})\right) \leq&(1+\frac{1}{\lambda_0})\frac{\ell_S}{2}(\|\hat{x}_t^v-x_t^{v,\dagger}\|^2)\end{aligned} \tag{69}$$

Then, a sufficient condition of Eqn (68) is

$$(1+\frac{1}{\lambda_0})\frac{\ell_f+2\ell_T+D_v\ell_S}{2}\sum_{\tau=1}^{t}\|\hat{x}_\tau^v-x_\tau^{v,\dagger}\|^2\leq(\lambda-\lambda_0)\text{cost}_v(x_{1:t}^{v,\dagger}),\forall t\in[1,T] \tag{70}$$

Since $D_{\max}=\max_{v\in\mathcal{V}}D_v$ is the maximum node degree in the whole graph, then the sufficient condition becomes

$$\alpha^2\sum_{\tau=1}^{t}\|\tilde{x}_\tau^v-x_\tau^{v,\dagger}\|^2\leq\frac{2}{\ell_f+2\ell_T+D_{\max}\cdot\ell_S}\cdot\frac{\lambda-\lambda_0}{1+\frac{1}{\lambda_0}}\text{cost}_v(x_{1:t}^{v,\dagger}),\forall t\in[1,T] \tag{71}$$

We define $\hat{C} = \min_{v\in\mathcal{V},t\in[1,T]} \frac{\text{cost}_v(x^{v,\dagger}_{1:t})}{\sum_{i=1}^{t}\|x^{v,\dagger}_i - \tilde{x}^v_i\|^2}$ as the minimum normalized baseline cost, then we can have

$$\alpha \leq \min\left\{1, \sqrt{\frac{2}{\ell_f + 2\ell_T + D_{\max}\cdot\ell_S}\cdot\frac{\lambda-\lambda_0}{1+\frac{1}{\lambda_0}}\cdot\hat{C}}\right\} = \alpha_\lambda \tag{72}$$

In other words, as long as $\alpha \in [0, \alpha_\lambda]$, the robustness constraint is always satisfied. Based on the convex assumption on hitting cost, temporal cost and spatial cost, we have

$$\text{cost}_v(\hat{x}^v_{1:t}) = \text{cost}_v((1-\alpha)x^{v,\dagger}_{1:t} + \alpha\tilde{x}^{v,\dagger}_{1:t}) \leq (1-\alpha)\text{cost}_v(x^{v,\dagger}_{1:t}) + \alpha\cdot\text{cost}_v(\tilde{x}^v_{1:t}). \tag{73}$$

By setting $\alpha = \alpha_\lambda$ and taking expectation of both side over the data distribution, we finish the proof of the first term in Theorem B.1. □

Received August 2024; revised September 2024; accepted October 2024