Hierarchical Decision Making with Structured Policies: A Principled Design via Inverse Optimization

Yuexuan Wang [1] **Jingyuan Zhou** [2] **Kaidi Yang** [2]

Abstract

Hierarchical decision-making frameworks are pivotal for addressing complex control tasks, enabling agents to decompose intricate problems into manageable subgoals. Despite their promise, existing hierarchical policies face critical limitations: (i) reinforcement learning (RL)-based methods struggle to guarantee strict constraint satisfaction, and (ii) optimal control (OC)-based approaches often rely on myopic and computationally prohibitive formulations. To reconcile these trade-offs, hierarchical RL-OC architectures have emerged as a promising paradigm. However, the formulation of the lower-level optimization within these frameworks remains underexplored, often relying on heuristic or myopic objectives. In this work, we propose a principled framework that systematically integrates upper-level goal abstraction with structured lower-level decision making. We adopt an inverse optimization approach to inform the structure of the lower-level problem from expert demonstrations, ensuring that the objective of the lower-level policy remains aligned with the overall long-term task goal. To validate the approach, our framework is evaluated on distinct decision making tasks: network-based resource allocation and continuous collision avoidance. Empirical results demonstrate that our method consistently outperforms strong baselines based on end-to-end RL, learning-augmented optimal control, and existing hierarchical RL approaches in both efficiency and decision quality.

[1]Institute of Operations Research and Analytics, National University of Singapore [2]Department of Civil and Environmental Engineering, National University of Singapore. Correspondence to: Kaidi Yang <kaidi.yang@nus.edu.sg>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

1. Introduction

Real-time decision-making in cyber-physical systems, such as robotics, power grids, and transportation (Jendoubi & Bouffard, 2023; Zhou et al., 2024; Liang et al., 2025), is inherently challenging due to high-dimensional state and action spaces and complex physical constraints. Existing solutions largely stem from optimal control (OC) and deep reinforcement learning (RL). OC-based methods aim to optimize system performance over infinite or long horizons while ensuring stability and feasibility. These methods are well-suited for safety-critical systems due to their theoretical guarantees, but may scale poorly in high-dimensional or nonlinear settings (Yu-Geng et al., 2013). In contrast, RL-based approaches directly learns a policy from interactions with the environment, which scale well to complex tasks. Nevertheless, these approaches require extensive training and lack safety or constraint satisfaction guarantees due to their black-box nature (Zhao et al., 2023; Wang et al., 2020). These trade-offs have motivated growing interest in combining OC-based and RL-based methods to exploit the strengths of both paradigms.

A promising approach for combining OC- and RL-based methods is through a hierarchical architecture that decomposes decision-making into two sequential subproblems (Lew et al., 2023; Karnchanachari et al., 2020). The respective strengths of RL and OC fit naturally into this framework: the upper-level employs an RL policy for strategic planning, such as generating subgoals, while the lower level uses OC to ensure safe and feasible execution. This hierarchical architecture not only enhances scalability and feasibility but also aligns with human cognition, as humans tend to perform abstract planning guided by intrinsic motivation, grounded by fast, lower-level execution (Aubret et al., 2019).

Despite the promise of hierarchical RL–OC frameworks, the formulation of the lower-level optimization problem remains underexplored. The lower-level controller must be both computationally efficient and aligned with the upper-level goals, since a poorly designed formulation may inadvertently exclude high-quality solutions. Existing approaches have several limitations. First, most hierarchical methods adopt long-horizon OC formulations at the lower level to

preserve stability and feasibility guarantees (Song & Scaramuzza; Cheng et al., 2024; Landgraf et al., 2022). However, such long-horizon OC formulations are well known to incur prohibitive computational complexity for real-time applications (Karamanakos et al., 2014; Krishnamoorthy et al., 2020). Second, recent efforts leverage single-step OC to enhance computation efficiency by directly generating the value or constraints on the desired next state (Gammelli et al., 2023; Schmidt et al., 2024). Nevertheless, these formulations typically rely on myopic objectives without an appropriately designed formulation, leading to suboptimal trajectories (Rawlings et al., 2020; Lowrey et al., 2018). The above challenges highlight the importance of formulating the lower-level optimization problem in a way that reduces sub-optimality while simultaneously ensuring tractable computational complexity.

To address these limitations, we propose an inverse optimization-guided approach that systematically informs the design of the lower-level policy using a small set of expert demonstrations (Figure 1), which provide valuable insights. Rather than relying on manually specified objectives, we cast the construction of the lower-level cost function as an inverse optimization problem, aiming to recover a formulation under which expert decisions are (approximately) optimal. For a special class of lower-optimization problems with linear cost functions, we provide a theoretical characterization of the conditions under which the expert demonstrations are optimal.

Once the formulation is established, we develop computationally efficient methods to solve the resulting inverse problem and integrate the learned lower-level structure into a hierarchical RL–OC framework. We evaluate the proposed approach on three representative decision-making tasks from various fields: autonomous vehicle rebalancing, supply chain inventory management, and mobile robot navigation. The improvements in learning the formulation are validated from quantitative and qualitative perspectives. In light of the above discussion, we summarize the main contributions of this work as follows:

- We propose an inverse optimization-based approach to systematically construct the lower-level optimization formulation within a hierarchical RL-OC framework.
- We provide theoretical analysis for a special class of problems with broad applications, proposing a tractable cost structure and efficient inverse optimization formulation which ensures inverse-feasibility, forward-stability, and computational tractability.
- We demonstrate the effectiveness of the proposed framework on several scenarios from various fields, showcasing its practical relevance and potential impact in real-world applications.

2. Related Work

This work is related to the literature on hierarchical structured control policies. Depending on whether learning-based or model-based approaches are used at each level, existing works can be broadly classified into two categories: (i) hierarchical reinforcement learning (HRL) that employs RL at each level, and (ii) learning-based optimal control, e.g., frameworks integrating RL and Model Predictive Control (MPC), whereby a upper-level policy learns desired states or goals, and a simplified lower-level MPC ensures safe and feasible execution.

Hierarchical Reinforcement Learning. HRL decomposes a complex, difficult-to-solve problem into multiple simpler, smaller problems by setting subgoals (Kulkarni et al., 2016; Vezhnevets et al., 2017; Ma et al., 2021; Xie et al., 2021; Eppe et al., 2022; Qi et al., 2022; Huang et al., 2022; Gu et al., 2023; Mao et al., 2024; Luo et al., 2024; Zhang et al., 2024b; Hirt et al., 2024). We focus on how hierarchical policies utilize various forms of intrinsic motivation by setting subgoals. For example, Naveed et al. (2021) uses a high-level policy to choose maneuvers for autonomous driving, while a low-level planner generates waypoints accordingly. Vezhnevets et al. (2017) proposes Feudal Networks, where a manager sets abstract goals that are conveyed to and enacted by the Worker module. Another common choice is to represent goals as desired states; e.g., Nachum et al. (2018) treats goal states as high-level actions and rewards the low-level policy for reaching them. In recent years, a growing body of work has investigated how to define subgoals and how to search efficiently within the subgoal space (Liu et al., 2021; Ma et al., 2023). Nevertheless, existing studies do not explicitly model how subgoals induce final actions via a well-defined lower-level optimization problem, often relying instead on implicit or black-box policy mappings. Moreover, as previously discussed, incorporating constraints to ensure safety strictly is challenging in reinforcement learning framework. In next section, we focus on how learning, especially reinforcement learning, interplays with optimization in the previous literature.

Learning-based OC. In the control community, various real-world control problems are solved by using learning-based OC. In many existing works, learning-based methods are often applied to learn cost functions or system dynamics (Lenz et al., 2015; Coulson et al., 2019; Hewing et al., 2020; Dogan et al., 2023; Zhang et al., 2024a; Lu et al., 2024; Zhang et al., 2024a; Dinkla et al., 2026). However, solving optimal control problems in real time still poses challenges when long control horizons are used, due to the high dimensionality of variables and the complexity of constraints. Prior work has attempted to reduce the computational burden by approximating the long-horizon OC problem with a single-step formulation and learning terminal cost to allevi-

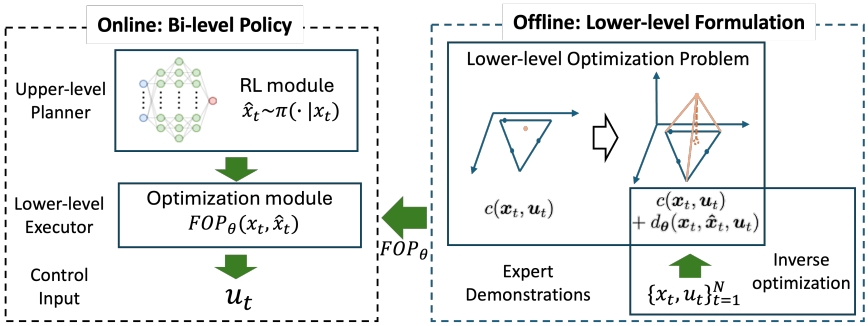


Figure 1. We propose an RL-OC hierarchical decision-making framework with lower-level policy informed by inverse optimization.

ate myopic behavior (Abdufattokhov et al., 2021; Alsmeier et al., 2024; Zhang et al., 2024a).

However, learning a terminal cost alone can be insufficient when the horizon is aggressively shortened, as the terminal term would need to encode most of the long-term planning signal, which is often highly state-dependent, and may be difficult to represent. This motivates RL-OC frameworks. Gammelli et al. (2023) propose to leverage reinforcement learning to learn upper-level actions to shorten the control horizon of network flow control problem. In this framework, reinforcement learning is applied to generate desired next states that guide the lower-level executor toward maximizing cumulative rewards. However, the lower-level optimization problem still hinges on myopic objectives and does not rely on an explicitly and appropriately designed formulation. In addition, Schmidt et al. (2024) study hierarchical RL-OC in an offline setting and focus on generating upper-level subgoals. Their framework assumes that the offline data are optimal for the lower-level control problem, which can be sometimes unrealistic and leaves the design of the lower-level OC formulation unaddressed.

Related to this line of research, our work investigates how to formulate the lower-level optimization problem in hierarchical RL-OC framework to alleviate sub-optimality issues stemming from structural myopia.

3. Methodology

3.1. Problem Setting and Preliminary

Let us consider a general multi-step decision-making problem formulated in (1).

$$\begin{aligned} \min_{\{\boldsymbol{u}_t\}_{t=0}^{\infty}} \quad & \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} c(\boldsymbol{x}_t, \boldsymbol{u}_t) \\ \text{s.t.} \quad & \boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t) + g(\boldsymbol{x}_t)\boldsymbol{u}_t, \quad \forall t \geq 0 \\ & \boldsymbol{x}_t \in \mathcal{X}_t, \quad \forall t \geq 0 \\ & \boldsymbol{u}_t \in \mathcal{U}_t, \quad \forall t \geq 0 \end{aligned} \tag{1}$$

where $\boldsymbol{x}_t \in \mathbb{R}^n$ is the system state at time step t, $\boldsymbol{u}_t \in \mathbb{R}^m$ is the control input, and $c(\boldsymbol{x}_t, \boldsymbol{u}_t)$ is the stage cost function. The system dynamics are assumed to be control-affine (i.e., linear in $\boldsymbol{u}_t$) with functions $f(\cdot)$ and $g(\cdot)$. The feasible sets $\mathcal{X}_t$ and $\mathcal{U}_t$ encode admissible states and inputs, respectively. The initial state x_0 is known.

Remark 3.1 (Generality of problem setting). We consider the control-affine dynamics in (1) for two reasons. First, control-affine models represent a fundamental and broad class of systems in cyber-physical applications (Schmitz et al., 2025; Kazemian et al., 2024). Even when a system is not canonically control-affine, prior work has shown that, for some systems, control-affine approximations can still be leveraged to enable effective control (Mao et al., 2017; Mutha et al., 1997). Second, this choice allows us to isolate and highlight the proposed inverse-optimization design framework, rather than on the additional algorithmic challenge of solving inverse problems for more general lower-level programs. Importantly, the framework can be extended to non-control-affine dynamics, at the cost of employing more sophisticated inverse-problem solvers, which have been extensively studied in prior work (Bertsimas et al., 2015; Schaefer, 2009).

Due to the infinite horizon and the presence of complex constraints, solving Problem (1) is often computationally intractable. A standard workaround is a finite-horizon approximation. However, OC problems with long time horizons can still be computationally challenging for large-scale systems, which do not satisfy the real-time requirements of practical applications. To address this issue, we generalize the bi-level decision-making framework proposed by Gammelli et al. (2023) to a more general problem setting.

3.2. Bi-level Framework

The hierarchical RL-OC framework is implemented as an end-to-end system shown in Problem (2).

$$\begin{aligned} \pi^* \in \arg\min_{\pi\in\Pi} \mathbb{E}_\tau & \left[\sum_{t=0}^{\infty} \gamma^t c(\boldsymbol{u}_t, \boldsymbol{x}_t)\right] \\ \text{s.t.} \quad & \boldsymbol{h}_t \sim \pi(\boldsymbol{h}_t \mid \boldsymbol{x}_t) \\ & \boldsymbol{u}_t = \mathrm{FOP}(h_t, \boldsymbol{x}_t) \end{aligned} \tag{2}$$

The overall policy π^* composes an upper-level RL policy π and the solution to a lower-level optimization problem FOP. The upper-level policy encodes task-relevant abstract information or goals to produce the intrinsic subgoal $\boldsymbol{h}_t$. The lower-level optimization module receives both the intrinsic subgoal h_t and the current state $\boldsymbol{x}_t$ to compute the control input $\boldsymbol{u}_t$ that is feasible.

Remark 3.2 (Practical requirements for RL-OC frameworks). There are two requirements for Problem (2). First, for real-time deployment, FOP should capture operational constraints and be computationally efficient to solve. Second, to enable stable training, the intrinsic subgoal $\boldsymbol{h}_t$ is preferably low-dimensional.

Following Remark 3.2, a commonly used form of subgoals is a linear transformation of the desired next state $\boldsymbol{x}_t^{\text{des}}$ using a known matrix C, denoted by $\hat{\boldsymbol{x}}_t = C\boldsymbol{x}_t^{\text{des}}$ (Gammelli et al., 2023; Schmidt et al., 2024). This subgoal guides the system toward the desired state by inducing appropriate lower-level actions u_t. Here, in scenarios with high state dimensions, the transformation matrix C serves to compress the action space of the upper-level RL by mapping the high-dimensional system state to a lower-dimensional planning space. The transformation matrix C is typically set as an identity matrix in scenarios with low state dimensions. Overall, the bi-level decision-making framework is given in (3) and (4).

$$\pi^* \in \arg\min_{\pi\in\Pi} \mathbb{E}_\tau \left[\sum_{t=0}^{\infty} \gamma^t c(\boldsymbol{u}_t, \boldsymbol{x}_t)\right] \tag{3a}$$

$$s.t. \quad \hat{\boldsymbol{x}}_t \sim \pi(\hat{\boldsymbol{x}}_t \mid \boldsymbol{x}_t) \tag{3b}$$

$$\boldsymbol{u}_t \in \mathrm{FOP}(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t) \tag{3c}$$

$$\mathrm{FOP}(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t) := \arg\min_{\boldsymbol{u}_t} \; c^{\mathrm{FOP}}(\boldsymbol{x}_t, \boldsymbol{u}_t) \tag{4a}$$

$$\text{s.t.} \quad Cf(\boldsymbol{x}_t) + Cg(\boldsymbol{x}_t)\boldsymbol{u}_t = \hat{\boldsymbol{x}}_t, \tag{4b}$$

$$f(\boldsymbol{x}_t) + g(\boldsymbol{x}_t)\boldsymbol{u}_t \in \mathcal{X}_{t+1}, \; \boldsymbol{u}_t \in \mathcal{U}_t \tag{4c}$$

where $c^{\mathrm{FOP}}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ denotes the objective of the lower-level optimization problem, which we assume to be convex to support real-time decision-making. Importantly, $c^{\mathrm{FOP}}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ is not required to be the same as the overarching task stage cost $c(\boldsymbol{x}_t, \boldsymbol{u}_t)$; instead, it can be viewed as a convex surrogate, while any mismatch, e.g., due to an intrinsically non-convex $c(\boldsymbol{x}_t, \boldsymbol{u}_t)$, can be learned from data.

By leveraging intrinsic subgoals learned by the upper-level policy, the lower-level executor is able to make near-instantaneous, subgoal-conditioned decisions, which is particularly advantageous in time-sensitive or high-dimensional environments. However, the formulation of FOP still remains ambiguous in the bi-level framework, as discussed in Remark 3.3.

Remark 3.3 (Importance of proper lower-level formulations). We highlight two considerations for Problem (4). First, the transformed desired next state $\hat{\boldsymbol{x}}_t$ produced by the upper-level RL may be infeasible in practice, e.g., planned trajectories in robotics tasks, which require the design of a cost function to penalize violations. Such a cost function must be aligned with the overarching objective of the decision-making problem to avoid sub-optimality. Second, even when $\hat{x}$ is feasible, the formulation of the lower-level optimization still requires careful design to mitigate myopic decisions, particularly when the transformation matrix C reduces the state dimension. Specifically, there may exist multiple actions $\boldsymbol{u}_t$ that satisfy the constraint $Cf(\boldsymbol{x}_t) + Cg(\boldsymbol{x}_t)\boldsymbol{u}_t = \hat{\boldsymbol{x}}_t$ but yield different next states under the true dynamics, which will influence future rewards. Therefore, it is important to properly design the lower-level problem to align the action selection with the overarching objective.

In the next subsection, we will present an inverse optimization approach to inform the design of the lower-level optimization problem.

3.3. Learning Problem Formulation: Inverse Optimization

3.3.1. General Framework

We assume access to a set of expert data $\{\boldsymbol{x}_t, \boldsymbol{u}_t\}_{t\in\mathcal{T}_e}$ generated by an existing decision-making policy, such as Model Predictive Control (MPC) or other high-quality heuristics ensuring strict safety guarantees, where $\mathcal{T}_e$ represents the set of time steps at which the dataset is collected. We make three remarks regarding the expert dataset. First, as high-quality solution methods may not be suitable for making real-time decisions in practice, the expert data can be derived offline rather than from real-world operations. Second, for the same reason, the dataset is assumed to be small, which makes supervised learning methods such as imitation learning less suitable. Third, $\mathcal{T}_e$ is chosen to cover diverse operating conditions and does not have to consist of consecutive time steps.

The core of our approach is to recover latent structure encoded in the expert data. Specifically, we parameterize the

lower-level optimization problem as

$$\mathrm{FOP}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t) := \arg\min_{\boldsymbol{u}_t}\{\tilde{c}(\boldsymbol{x}_t, \boldsymbol{u}_t) + d_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t, \boldsymbol{u}_t) \mid \boldsymbol{a}_t + B_t\boldsymbol{u}_t = \hat{\boldsymbol{x}}_t, \boldsymbol{u}_t \in \widehat{\mathcal{U}}_t\} \tag{5}$$

where $\boldsymbol{a}_t := Cf(\boldsymbol{x}_t)$, $B_t := Cg(\boldsymbol{x}_t)$, and $\widehat{\mathcal{U}}_t := \mathrm{co}\left(\mathcal{U}_t \cap \{\boldsymbol{u}_t | f(\boldsymbol{x}_t) + g(\boldsymbol{x}_t)\boldsymbol{u}_t \in \mathcal{X}_{t+1}\}\right)$, where $\mathrm{co}(\cdot)$ represents the convex hull. Specifically, the term $\tilde{c}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ corresponds to the exact stage cost $c(\boldsymbol{x}_t, \boldsymbol{u}_t)$ when it is convex; otherwise, it serves as a convex surrogate to preserve computational tractability. Typical choices for $\tilde{c}(\cdot)$ include quadratic approximations or convex envelopes. The term $d_{\boldsymbol{\theta}}(\cdot)$ parameterized by parameter $\boldsymbol{\theta}$ is also assumed to be convex and will be properly designed to align the lower-level optimization problem to the objective of the overarching decision-making problem.

We then convert the design of the lower-level optimization problem into estimating $\boldsymbol{\theta}$ from the expert dataset. Speficically, we aim to find $\boldsymbol{\theta}$ via inverse optimization such that for each expert pair $(\boldsymbol{x}_t, \boldsymbol{u}_t)$, the action $\boldsymbol{u}_t$ is (approximately) optimal for $\mathrm{FOP}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t)$ under some subgoal $\hat{\boldsymbol{x}}_t$. Specifically, let $\mathcal{U}^{\mathrm{opt}}(\boldsymbol{\theta}, \boldsymbol{x}_t, \hat{\boldsymbol{x}}_t)$ denote the optimal solution set corresponding to parameter $\boldsymbol{\theta}$, state $\boldsymbol{x}_t$, and subgoal $\hat{\boldsymbol{x}}_t$. In practice, since the lower-level optimization is convex, $\mathcal{U}^{\mathrm{opt}}(\boldsymbol{\theta}, \boldsymbol{x}_t, \hat{\boldsymbol{x}}_t)$ can be characterized by the Karush–Kuhn–Tucker (KKT) conditions. Then, the inverse optimization problem can be formulated as

$$\min_{\boldsymbol{\theta}} \left\{ \kappa h(\boldsymbol{\theta}) + \frac{1}{|\mathcal{T}_e|} \sum_{t \in \mathcal{T}_e} \ell\left(\boldsymbol{u}_t, \mathcal{U}^{\mathrm{opt}}(\boldsymbol{\theta}, \boldsymbol{x}_t, \hat{\boldsymbol{x}}_t)\right) \,\middle|\, \boldsymbol{\theta} \in \boldsymbol{\Theta} \right\}. \tag{6}$$

where the objective function is a weighted combination of two parts: (i) the sum of losses with each loss $\ell(\boldsymbol{u}_t, \mathcal{U}_t^{\mathrm{opt}}(\boldsymbol{\theta}))$ indicating the deviation of expert action $\boldsymbol{u}_t$ from the optimal solution set $\mathcal{U}_t^{\mathrm{opt}}(\boldsymbol{\theta})$, i.e., the sub-optimality of the expert action, and (ii) a user-defined, application-specific regularization term $h(\boldsymbol{\theta})$ representing prior information or user preference regarding $\boldsymbol{\theta}$, with γ as the weight. This formulation follows a standard data-driven inverse optimization paradigm (Chan et al., 2025).

Remark 3.4 (Inverse optimization for lower-level formulations). We make three remarks regarding the inverse optimization framework. First, we focus on learning the objective function of $\mathrm{FOP}_{\boldsymbol{\theta}}$ rather than its constraints, as constraints are typically dictated by physical requirements. Second, without loss of generality, we assume the constraint $\boldsymbol{a}_t + B_t\boldsymbol{u}_t = \hat{\boldsymbol{x}}_t$ is feasible. If this is not the case, we can relax it as $\boldsymbol{a}_t + B_t\boldsymbol{u}_t = \hat{\boldsymbol{x}}_t + \boldsymbol{\epsilon}_t$, and augment the action, corresponding matrix, and cost function as $\tilde{\boldsymbol{u}}_t = [\boldsymbol{u}_t; \boldsymbol{\epsilon}_t]$, $\tilde{B}_t = [B_t, I]$, and $d_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t, \tilde{\boldsymbol{u}}_t)$, respectively, which yields a lower-level optimizer of the same form.

Without loss of generality, we assume $\widetilde{\mathcal{U}}_t$ is a polytope represented by $\widetilde{\mathcal{U}}_t = \{u \mid H_t\boldsymbol{u} \leq \boldsymbol{b}\}$. Under this assumption, the learning problem for θ can be formulated as

$$\min_{\boldsymbol{\theta}, \{\boldsymbol{w}_t, \boldsymbol{\lambda}_t, \boldsymbol{\epsilon}_t\}_{t \in \mathcal{T}_e}} \sum_{t \in \mathcal{T}_e} \|\boldsymbol{\epsilon}_t\|_2^2 + \kappa h(\boldsymbol{\theta}) \tag{7a}$$

$$\text{s.t.} \mathbf{0} \in \partial_{\boldsymbol{u}}\left(\tilde{c}(\boldsymbol{x}_t, \boldsymbol{u}_t) + d_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t, \boldsymbol{u}_t)\right) + \boldsymbol{\lambda}_t^T H_t + \boldsymbol{w}_t^T B_t, \ \forall t \tag{7b}$$

$$\boldsymbol{a}_t + B_t\boldsymbol{u}_t = \hat{\boldsymbol{x}}_t, \quad H_t\boldsymbol{u} \leq \boldsymbol{b}, \ \forall t \tag{7c}$$

$$\boldsymbol{\lambda}_t \geq 0, \ \forall t \tag{7d}$$

$$\varphi(\boldsymbol{u}_t, \boldsymbol{\lambda}_t, \boldsymbol{w}_t) + \boldsymbol{\epsilon}_t = c(\boldsymbol{x}_t, \boldsymbol{u}_t) + d_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t, \boldsymbol{u}_t), \ \forall t \tag{7e}$$

where $\partial_{\boldsymbol{u}}$ denotes the subdifferential of the cost function of $\mathrm{FOP}_{\boldsymbol{\theta}}$, which generalizes the gradient to allow for non-smooth objectives. The vectors $\boldsymbol{w}_t$ and $\boldsymbol{\lambda}_t$ are dual variables. The objective function in (7a) follows the structure of Problem (6), penalizing the sum of squared duality gaps while regularizing $\boldsymbol{\theta}$ via $h(\cdot)$. Constraints (7b)-(7e) are the KKT conditions, where (7e) relaxes the strong duality condition with a duality gap $\boldsymbol{\epsilon}_t$.

Concretely, we minimize the duality gap in Problem (7) instead of the distance between given expert decisions and optimal solutions to the forward optimization problem. This aims to mitigate the computational burden associated with solving mathematically challenging problems, for instance, when Problem (7) is non-convex, which will be analyzed in more detail in the following sections.

3.3.2. Special Case with Theoretical Analysis

To illustrate how inverse optimization serves as a principled guideline for designing lower-level formulations, we consider a special class of decision-making problems with a linear stage cost function $\boldsymbol{c}^\top \boldsymbol{u}_t$ and state-independent function $g(\cdot)$ (i.e., $B_t = B, \forall t$). We show that under this setting, our proposed cost structure can achieve desirable properties such as inverse feasibility, forward stability, and computation efficiency. At the end of this section, we generalize our conclusions to a broader class of problems with quadratic stage cost functions.

We make the following remarks on this special setting. First, this setting captures a broad range of applications in resource allocation, logistics, and energy systems, where strict constraint satisfaction and fast computation are critical. Second, it is important to note that despite the linearity of the stage cost, the original multi-stage decision-making problem (1) can still be challenging to solve in real-world operations due to (i) the potentially nonlinear structures of $f(\cdot)$ and $g(\cdot)$ and (ii) potentially high state and action dimensions.

In this special setting, we aim to derive an FOP formulation such that the inverse optimization is always feasible. This is important, as it can allow us to utilize any type of expert

data. In this case, we formulate the estimation of $\boldsymbol{\theta}$ as

$$\min_{\boldsymbol{\theta}} \left\{ \varphi(\boldsymbol{\theta}) \;\middle|\; \boldsymbol{\theta} \in \boldsymbol{\theta}_t^{\text{inv}}(\boldsymbol{u}_t)\ \forall t \in \mathcal{T}_e,\ \boldsymbol{\theta} \in \boldsymbol{\Theta} \right\}. \tag{8}$$

where $\boldsymbol{\theta}_t^{\text{inv}}(\boldsymbol{u}_t) := \left\{ \boldsymbol{\theta} \;\middle|\; \boldsymbol{u}_t \in \mathcal{U}_t^{\text{opt}}(\boldsymbol{\theta}) \right\}$ is the inverse feasible set, i.e., the set of parameter values under which the expert action $\boldsymbol{u}_t$ belongs to the optimal solution set $\mathcal{U}_t^{\text{opt}}(\boldsymbol{\theta})$ of the lower-level problem $\text{FOP}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t)$. The function $\varphi(\boldsymbol{\theta})$ denotes a user-defined, application-specific objective that resolves indeterminacy by selecting among multiple feasible parameters.

As stated in Proposition 3.5, without a properly designed cost structure $d_{\boldsymbol{\theta}}(\cdot)$, the lower-level formulation lacks the expressivity to rationalize expert behaviors, which means there may exist no feasible upper-level subgoal $\hat{x}_t$ that renders the observed expert actions optimal.

Proposition 3.5. *The inverse feasible set for Problem (4)*

$$\hat{\boldsymbol{\mathcal{X}}}_t^{inv}(\boldsymbol{u}_t) := \left\{ \hat{\boldsymbol{x}}_{\boldsymbol{t}} \;\middle|\; \boldsymbol{u}_t \in \mathcal{U}_t^{opt}(\hat{\boldsymbol{x}}_{\boldsymbol{t}}) \right\}.$$

is not guaranteed to be non-empty, where $\mathcal{U}_t^{opt}(\hat{\boldsymbol{x}}_{\boldsymbol{t}})$ is defined by the KKT conditions of Problem (4).

To address the sub-optimality, we propose an inverse-optimization–guided design procedure that leverages criteria such as inverse feasibility and forward stability to inform the lower-level formulation.

Stage 1: Validating optimality of expert decisions. We first construct a preliminary formulation of $\text{FOP}_{\boldsymbol{\theta}}$ such that the historical decisions $\{\boldsymbol{u}_t\}_{t=1}^T$ are possible to be optimal under the observed states $\{\boldsymbol{x}_t\}_{t=1}^T$. We achieve this goal by validating the feasibility of the inverse problem.

We select the terminal cost $d_{\boldsymbol{\theta}}(\cdot)$ represented by the summation of ReLU-based regularization terms, which demonstrates high efficiency despite its simplicity. Propositions 3.6 and 3.7 show the motivation to design such cost structure from both algebraic and geometric perspectives respectively.

$$\begin{aligned}\text{FOP}(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t) := \arg\min_{\boldsymbol{u}_t} \{ &\boldsymbol{c}^\top \boldsymbol{u}_t + \sum_{k=1}^{K} \max\{\boldsymbol{\theta}_k^T \boldsymbol{u}_t - \nu_k, 0\} \\ &\big| \boldsymbol{a}_t + B\boldsymbol{u}_t = \hat{\boldsymbol{x}}_t, \boldsymbol{u}_t \in \widetilde{\mathcal{U}}_t\}\end{aligned} \tag{9}$$

Proposition 3.6. *The inverse feasible set for Problem (9), $\boldsymbol{\theta}_t^{*inv}(\boldsymbol{u}_t) := \left\{ \boldsymbol{\theta} \;\middle|\; \boldsymbol{u}_t \in \mathcal{U}_t^{*opt}(\boldsymbol{\theta}) \right\}$, is always non-empty, where $\mathcal{U}_t^{*opt}(\boldsymbol{\theta})$ is defined by KKT conditions of Problem (9).*

Proposition 3.7. *By appropriately enlarging the decision space with auxiliary variables in Problem (9), any given set of expert decisions can be made optimal in the lifted space.*

Stage 2: Ensuring forward stability. Although Problem (9) has proven to make expert data optimal, i.e., the inverse feasibility is guaranteed, another critical issue in inverse optimization, named forward stability, proposed by Shahmoradi & Lee (2022), is not guaranteed in the case of a linear program. The definition of forward stability is defined as follows.

Definition 3.8 (Forward Instability (Shahmoradi & Lee, 2022))**.** Given a set of expert observations $\hat{\mathcal{U}}$, the *forward instability* of an inverse solution $\hat{\boldsymbol{\theta}} \in \boldsymbol{\theta}^{*\text{inv}}(\hat{\mathcal{U}})$ is defined as

$$\max_{u \in \mathcal{U}^{*opt}(\hat{\boldsymbol{\theta}})} \left\{ d(\hat{\mathcal{U}}, u) \right\}, \tag{10}$$

where $\mathcal{U}^{*opt}(\hat{\boldsymbol{\theta}})$ denotes the set of forward optimal solutions corresponding to $\hat{\boldsymbol{\theta}}$. This value quantifies the worst-case distance between a forward solution u induced by $\hat{\boldsymbol{\theta}}$ and the expert data $\hat{\mathcal{U}}$, measuring how unstable the inverse solution $\hat{\boldsymbol{\theta}}$ can be.

To solve this issue, we include small quadratic terms into objective function to ensure strong convexity of objective function, such that forward stability is improved and the expert decisions are approximately optimal at the same time. Overall, we formulate the forward optimization problem as follows.

$$\begin{aligned}\text{FOP}(\hat{\boldsymbol{x}}_t, \boldsymbol{x}_t) := \arg\min_{\boldsymbol{u}_t} \{ &\boldsymbol{c}^\top \boldsymbol{u}_t + \sum_{k=1}^{K} \max\{\boldsymbol{\theta}_k^\top \boldsymbol{u}_t - \nu_k, 0\} \\ &+ l\|\boldsymbol{u}_t\|_2^2 \;\big|\; \boldsymbol{a}_t + B\boldsymbol{u}_t = \hat{\boldsymbol{x}}_t, \boldsymbol{u}_t \in \widetilde{\mathcal{U}}_t\}\end{aligned} \tag{11}$$

We propose the following inverse optimization formulation, where dual variables are denoted by $z_{kt}, y_{kt}, \boldsymbol{w}_t, \boldsymbol{\lambda}_t$ and τ_{kt} is the auxiliary variable. $g(\boldsymbol{\lambda}_t, \boldsymbol{z}_t, \boldsymbol{y}_t, \boldsymbol{w}_t)$ is the dual objective function with explicit form in this problem. To keep the formulation concise, we omit the explicit form of this function.

$$\begin{aligned}\min_{\substack{z_{kt}, y_{kt}, \boldsymbol{\lambda}_t, \boldsymbol{w}_t, \\ \boldsymbol{\epsilon}_t, \boldsymbol{\theta}_k, \nu_k}} \quad & \sum_{t \in \mathcal{T}_e} \|\boldsymbol{\epsilon}_t\|_2^2 + \rho \sum_k \|\boldsymbol{\theta}_k\|_2^2 \\ \text{s.t.} \quad & 1 - z_{kt} - y_{kt} = 0, \quad k = 1, \ldots, K,\ t \in \mathcal{T}_e \\ & \boldsymbol{c}^\top + \boldsymbol{\lambda}_t^\top H + \boldsymbol{w}_t^\top B \\ & \qquad + \sum_k z_{kt} \boldsymbol{\theta}_k^\top + 2l\boldsymbol{u}_t = 0, \quad t \in \mathcal{T}_e \\ & g(\boldsymbol{\lambda}_t, \boldsymbol{z}_t, \boldsymbol{y}_t, \boldsymbol{w}_t) + \epsilon_t = \boldsymbol{c}^\top \boldsymbol{u}_t \\ & \qquad + \sum_{k=1}^{K} \tau_{kt} + l\boldsymbol{u}_t^\top \boldsymbol{u}_t, \quad t \in \mathcal{T}_e \\ & \tau_{kt} \geq \boldsymbol{\theta}_k^\top \boldsymbol{u}_t - \nu_k, \quad k = 1, \ldots, K,\ t \in \mathcal{T}_e \\ & z_{kt}, y_{kt}, \boldsymbol{\lambda}_t, \tau_{kt} \geq 0, \quad k = 1, \ldots, K,\ t \in \mathcal{T}_e\end{aligned} \tag{12}$$

The problem is non-convex including bi-linear terms in constraints which poses significant challenges for computational tractability. We minimize duality gap together with

regularizing norm of $\boldsymbol{\theta}_k$ for robustness. For solving Problem (12), it can be reformulated as a Mixed Integer Program and solved using the spatial branch-and-bound algorithm within a reasonable time (Smith & Pantelides, 1999). By minimizing the duality gap instead of the distance between optimal solutions and given expert solutions, the number of bilinear terms is reduced from $\mathcal{O}(|\mathcal{A}|K + |\mathcal{A}|^2)$ to $\mathcal{O}(|\mathcal{A}|K)$. Thus, the number of constraints introduced by the construction of the McCormick envelope can be significantly reduced.

What's more, Proposition 3.9 demonstrates that the distance between the input and optimal solutions remains bounded. By introducing a sufficient number of ReLU-based terms, we can constrain the values of ϵ_0 within a small range, thereby controlling the upper bound of the distance.

Proposition 3.9. *Problem (12) yields optimal solutions $\boldsymbol{u}_t^*$ satisfying $|f(\boldsymbol{u}_t^*) - f(\boldsymbol{u}_t)| \leq \epsilon_0$, s.t.*

$$\|\boldsymbol{u}_t^* - \boldsymbol{u}_t\| \leq \sqrt{\frac{\epsilon_0}{l}} \tag{13}$$

where $\epsilon_0 = \max_t \boldsymbol{\epsilon}_t^*$ and $\boldsymbol{u}_t$ is the given expert decision in time step t.

For the broader class of control problems with quadratic objective functions, we can still infer parameters as shown in Problem (12). Proposition 3.9 can be extended as shown below.

Corollary 3.10. *When the objective function is given by $\boldsymbol{u}^T R \boldsymbol{u} + \boldsymbol{x}^T Q \boldsymbol{x}$ with the assumption $R \succ 0$, the upper bound will be achieved by $\sqrt{\frac{\lambda_{min}(R)}{l}}$, where $\lambda_{min}(R)$ denotes the minimum eigenvalue of R.*

4. Case Study

In this section, we evaluate the proposed framework through three case studies from different fields: (1) autonomous vehicle rebalancing; (2) inventory management in multi-warehouse multi-retailer supply chain, and (3) mobile robot navigation. The results highlight its ability to enhance decision quality and provide interpretable solutions in dynamic environments. The details of the three environments are given in Appendix B.

4.1. Experimental Setup

Data Collection: We generate expert demonstrations by simulating a full-horizon MPC controller over a finite horizon T. The dataset consists of state-action pairs collected offline, designed to cover diverse operating conditions for each distinct task. Crucially, we utilize only a handful of these demonstrations to infer the formulation of the lower-level optimization policy via inverse optimization

RL Implementation: We employ A2C with Graph Neural Networks in the hierarchical framework for the AV and Supply Chain tasks, and Soft Actor-Critic (SAC) for the Mobile Robot task. Detailed network architectures and hyperparameters are provided in Appendix C.1.

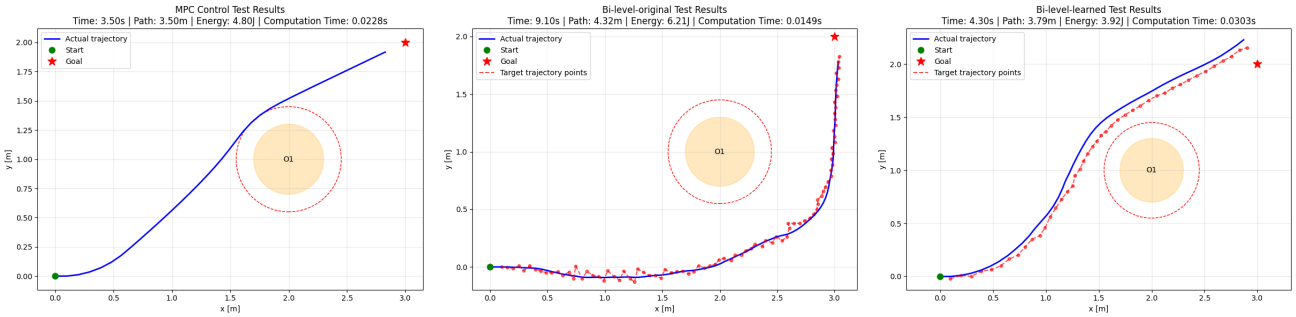
Oracle and Baselines: We use **MPC** with long-time horizon as an Oracle, and baselines include:

- **Bi-level-original**: RL-OC framework where lower-level optimization policy is unchanged with original single-step cost (Gammelli et al., 2023).
- **End-to-end RL**: End-to-end reinforcement learning that trains a single model to directly map inputs to the final control actions.
- **Inverse MPC**: Identify the optimal cost function that minimizes the discrepancy between the actual trajectory and expert demonstration (Zhang et al., 2024a).
- **S-type Heuristic**: Given the widespread use of established heuristics such as (S, s) and (Q, r) policies in inventory management, we specifically include a representative heuristic baseline to evaluate our framework's performance.
- **Value Approximation**: Quadratic terminal cost estimation via value function (Abdufattokhov et al., 2021), whose performance will be analyzed in the Appendix C.4.1.

4.2. Model Evaluation

We compare the RL-OC framework equipped with a learned lower-level policy against variant baselines. Table 1 and 2 summarize the comparative results. First, our hierarchical RL-OC framework achieves a marked improvement, which highlights the effectiveness applying RL-OC framework to mitigate the curse of dimensionality by compressing the original action space into lower-dimensional subgoals. Building on the hierarchical structure, our Bi-level-learned approach improves upon the Bi-level-original baseline by employing a refined lower-level formulation. Additionally, the framework significantly outperforms Inverse MPC baselines, confirming that our ReLU-based cost structure informed by inverse optimization provides a more principled design than the standard quadratic forms used in prior works.

The qualitative advantages of our framework are further illustrated in Figure 2 below and Figure 4 in Appendix C.2.2 , which compare the system trajectories across different tasks. By leveraging inverse optimization, the proposed framework effectively recovers the latent behavioral patterns in expert demonstrations. As a result, the trajectories generated by our Bi-level-learned method exhibit significantly higher alignment with the expert (MPC) benchmarks compared to the Bi-level-original baseline.



(a) Expert demonstrations (MPC) *(b)* Bi-level-original *(c)* Bi-level-learned

Figure 2. Trajectory comparison for the Mobile Robot Navigation task across different methods

Table 1. Performance Comparison on Supply Chain Inventory Management.

Method	**Reward**	**Served Demand**
MPC	11335 (±34.3)	1337 (±5.61)
End-to-end RL	2764 (±90.9)	476 (±18.4)
Inverse MPC	5608 (±31.8)	643 (±7.04)
S-type Heuristic	2709 (± 195.6)	601 (± 12.5)
Bi-level-original	7491 (±26.5)	778 (±2.62)
Bi-level-learned (Ours)	**9442** (±42.3)	**930** (±1.17)

Moreover, compared with MPC, the bi-level framework shortens the planning horizon from T steps to one step, which significantly reduces decision-making time. We compare the time spent to make decisions for one step in Table 3. It shows a significant reduction in runtime which indicates that our method can significantly improve computational efficiency without substantially compromising solution quality. These results highlight the effectiveness of optimizing the lower-level problem structure in enhancing overall system performance. Besides, scalability analysis, sensitivity analysis, interpretations of learned cost structure and motivations of proposed methods are provided and discussed in the Appendix C.

5. Limitations and Future Work

While the developed hierarchical RL-OC scheme provides an effective solution for real-time, safety-critical control tasks, it presents certain limitations that motivate our future research directions.

First, our methodology is primarily evaluated on tasks with structured state representations. To broaden its real-world applicability, we intend to scale the proposed framework to larger-scale systems with complex dynamics, and to expand its scope to applications with unstructured inputs. A key advantage of our hierarchical RL-OC architecture is its potential to bridge high-dimensional perception with structured control, where the upper-level RL learns to extract latent representations and generate low-dimensional subgoals, while the lower-level optimization then strictly enforces physical constraints while optimizing for consistency with these learned representations and long-term objectives. Second, the quality of the solutions to the inverse optimization depends on the quality of the provided expert demonstrations. Although our empirical sensitivity analysis demonstrates reasonable robustness to data noise, learning from highly sub-optimal data remains challenging. We aim to conduct a more rigorous investigation into handling noisy or sub-optimal expert demonstrations. A natural extension is to develop noise-aware inverse formulation.

6. Conclusion

In this work, we propose a hierarchical RL-OC framework that addresses key challenges in real-time, safety-critical decision-making scenarios, where the upper-level policy generates subgoals that abstract planning objectives, while the lower-level controller executes these subgoals through a structured optimization problem. A central contribution of this work is the principled construction of the lower-level optimization formulation via inverse optimization, rather than relying on heuristic design. By recovering the latent cost structure from expert demonstrations, we ensure that the lower-level controller is aligned with the overarching objectives. Empirical evaluations across diverse domains demonstrate that our framework not only outperforms strong baselines in decision quality but also provides interpretable control policy. This work highlights a promising direction for combining learning-based goal abstraction with structured optimization. In particular, our study positions inverse optimization as an effective bridge between reinforcement learning and optimal control.

Table 2. Performance Comparison on AV Rebalancing and Mobile Robot Navigation tasks

Method	AV Rebalancing		Mobile Robot Navigation		
	Reward ↑	Served Demand ↑	Time (s) ↓	Length (m) ↓	Energy (J) ↓
MPC (Oracle)	35725 ($\pm$41.6)	3203 ($\pm$3.07)	3.50	3.50	4.81
End-to-end RL	20989 ($\pm$213.5)	2334.8 ($\pm$14.9)	7.60 ± 0.11	4.29 ± 0.02	6.78 ± 0.23
Inverse MPC	32426 ($\pm$401.2)	2996 ($\pm$28.6)	*	*	*
Bi-level-original	33406 ($\pm$128.1)	2993 ($\pm$8.06)	9.14 ± 0.08	4.35 ± 0.02	6.20 ± 0.13
Bi-level-learned (Ours)	**35019** ($\pm$53.0)	**3141** ($\pm$6.14)	$\mathbf{4.40 \pm 0.10}$	$\mathbf{4.14 \pm 0.07}$	$\mathbf{1.52 \pm 0.06}$

Note: Upward (↑) and downward (↓) arrows indicate that higher or lower values are preferred, respectively. Asterisks (*) denote collisions where the agent fails to reach the goal. MPC serves as the Oracle.

Table 3. Comparison of Computational Time Across Scenarios

Scenarios	Runtime (s)	
	Our Method	MPC
AV Rebalancing	**0.025**	1.44
Inventory Management	**0.034**	0.082
Mobile Robot Navigation	**0.0187**	0.0275

Acknowledgements

This research was supported by the Singapore Ministry of Education (MOE) under its Academic Research Fund Tier 1 (A-8001183-00-00).

Impact Statement

This paper presents a methodological advance in integrating reinforcement learning and optimal control by leveraging inverse optimization to construct structured policies. There are many potential societal consequences of our work, particularly in improving the efficiency and safety of cyber-physical systems, including transportation, power grids, and robotics. Overall, we don't foresee any specific societal consequences of this work that necessitate a separate discussion.

References

Abdufattokhov, S., Zanon, M., and Bemporad, A. Learning convex terminal costs for complexity reduction in mpc. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 2163–2168. IEEE, 2021.

Alsmeier, H., Savchenko, A., and Findeisen, R. Neural horizon model predictive control-increasing computational efficiency with neural networks. In *2024 American Control Conference (ACC)*, pp. 1646–1651. IEEE, 2024.

Aubret, A., Matignon, L., and Hassas, S. A survey on intrinsic motivation in reinforcement learning. *arXiv preprint arXiv:1908.06976*, 2019.

Bertsimas, D., Gupta, V., and Paschalidis, I. C. Data-driven estimation in equilibrium using inverse optimization. *Mathematical Programming*, 153(2):595–633, 2015.

Chan, T. C., Mahmood, R., and Zhu, I. Y. Inverse optimization: Theory and applications. *Operations Research*, 73 (2):1046–1074, 2025.

Cheng, Z., Zeng, X., Fang, H., Wang, G., and Dou, L. Hierarchical mpc-based motion planning for automated vehicles in parallel autonomy. *Unmanned Systems*, 12 (05):927–938, 2024.

Coulson, J., Lygeros, J., and Dörfler, F. Data-enabled predictive control: In the shallows of the deepc. In *2019 18th European control conference (ECC)*, pp. 307–312. IEEE, 2019.

Dinkla, R., Oomen, T., Mulders, S. P., and van Wingerden, J.-W. Closed-loop data-enabled predictive control and its equivalence with closed-loop subspace predictive control. *Automatica*, 183:112556, 2026.

Dogan, I., Shen, Z.-J. M., and Aswani, A. Regret analysis of learning-based mpc with partially unknown cost function. *IEEE Transactions on Automatic Control*, 69(5):3246–3253, 2023.

Eppe, M., Gumbsch, C., Kerzel, M., Nguyen, P. D., Butz, M. V., and Wermter, S. Intelligent problem-solving as integrated hierarchical reinforcement learning. *Nature Machine Intelligence*, 4(1):11–20, 2022.

Gammelli, D., Harrison, J., Yang, K., Pavone, M., Rodrigues, F., and Pereira, F. C. Graph reinforcement learning for network control via bi-level optimization. *arXiv preprint arXiv:2305.09129*, 2023.

Gu, Z., Gao, L., Ma, H., Li, S. E., Zheng, S., Jing, W., and Chen, J. Safe-state enhancement method for autonomous driving via direct hierarchical reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 24(9):9966–9983, 2023.

Hewing, L., Wabersich, K. P., Menner, M., and Zeilinger, M. N. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):269–296, 2020.

Hirt, S., Pfefferkorn, M., Mesbah, A., and Findeisen, R. Stability-informed bayesian optimization for mpc cost function learning. *IFAC-PapersOnLine*, 58(18):208–213, 2024.

Huang, K., Wei, K., Li, F., Yang, C., and Gui, W. Lstm-mpc: A deep learning based predictive control method for multimode process control. *IEEE Transactions on Industrial Electronics*, 70(11):11544–11554, 2022.

Jendoubi, I. and Bouffard, F. Multi-agent hierarchical reinforcement learning for energy management. *Applied Energy*, 332:120500, 2023.

Karamanakos, P., Geyer, T., Oikonomou, N., Kieferndorf, F. D., and Manias, S. Direct model predictive control: A review of strategies that achieve long prediction intervals for power electronics. *IEEE Industrial Electronics Magazine*, 8(1):32–43, 2014.

Karnchanachari, N., Valls, M. I., Hoeller, D., and Hutter, M. Practical reinforcement learning for mpc: Learning from sparse objectives in under an hour on a real robot. In *Learning for Dynamics and Control*, pp. 211–224. PMLR, 2020.

Kazemian, K., Sattar, Y., and Dean, S. Random features approximation for control-affine systems. In *6th Annual Learning for Dynamics & Control Conference*, pp. 732–744. PMLR, 2024.

Krishnamoorthy, D., Biegler, L. T., and Jäschke, J. Adaptive horizon economic nonlinear model predictive control. *Journal of Process Control*, 92:108–118, 2020.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Landgraf, D., Völz, A., Kontes, G., Mutschler, C., and Graichen, K. Hierarchical learning for model predictive collision avoidance. *IFAC-PapersOnLine*, 55(20):355–360, 2022.

Lenz, I., Knepper, R. A., and Saxena, A. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*, volume 10, pp. 25. Rome, Italy, 2015.

Lew, T., Singh, S., Prats, M., Bingham, J., Weisz, J., Holson, B., Zhang, X., Sindhwani, V., Lu, Y., Xia, F., et al. Robotic table wiping via reinforcement learning and whole-body trajectory optimization. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7184–7190. IEEE, 2023.

Liang, J., Tan, C., Yan, L., Zhou, J., Yin, G., and Yang, K. Interaction-aware trajectory prediction for safe motion planning in autonomous driving: A transformer-transfer learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 2025.

Liu, C., Zhu, F., Liu, Q., and Fu, Y. Hierarchical reinforcement learning with automatic sub-goal identification. *IEEE/CAA journal of automatica sinica*, 8(10): 1686–1696, 2021.

Lowrey, K., Rajeswaran, A., Kakade, S., Todorov, E., and Mordatch, I. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848*, 2018.

Lu, Y., Li, Z., Zhou, Y., Li, N., and Mo, Y. Mpc-inspired reinforcement learning for verifiable model-free control. In *6th Annual Learning for Dynamics & Control Conference*, pp. 399–413. PMLR, 2024.

Luo, Y., Ji, T., Sun, F., Liu, H., Zhang, J., Jing, M., and Huang, W. Goal-conditioned hierarchical reinforcement learning with high-level model approximation. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

Ma, H., Vo, T. V., and Leong, T.-Y. Human-ai collaborative sub-goal optimization in hierarchical reinforcement learning. In *Proceedings of the AAAI symposium series*, volume 1, pp. 86–89, 2023.

Ma, Y., Hao, X., Hao, J., Lu, J., Liu, X., Xialiang, T., Yuan, M., Li, Z., Tang, J., and Meng, Z. A hierarchical reinforcement learning based optimization framework for large-scale dynamic pickup and delivery problems. *Advances in neural information processing systems*, 34: 23609–23620, 2021.

Mao, Y., Dueri, D., Szmuk, M., and Açıkmeşe, B. Successive convexification of non-convex optimal control problems with state constraints. *Ifac-PapersOnline*, 50 (1):4063–4069, 2017.

Mao, Z., Liu, Y., and Qu, X. Integrating big data analytics in autonomous driving: An unsupervised hierarchical reinforcement learning approach. *Transportation Research Part C: Emerging Technologies*, 162:104606, 2024.

Mutha, R. K., Cluett, W. R., and Penlidis, A. Nonlinear model-based predictive control of control nonaffine systems. *Automatica*, 33(5):907–913, 1997.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Naveed, K. B., Qiao, Z., and Dolan, J. M. Trajectory planning for autonomous vehicles using hierarchical reinforcement learning. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 601–606. IEEE, 2021.

Qi, C., Zhu, Y., Song, C., Yan, G., Xiao, F., Zhang, X., Cao, J., Song, S., et al. Hierarchical reinforcement learning based energy management strategy for hybrid electric vehicle. *Energy*, 238:121703, 2022.

Rawlings, J. B., Mayne, D. Q., Diehl, M., et al. *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, WI, 2020.

Schaefer, A. J. Inverse integer programming. *Optimization Letters*, 3(4):483–489, 2009.

Schmidt, C., Gammelli, D., Harrison, J., Pavone, M., and Rodrigues, F. Offline hierarchical reinforcement learning via inverse optimization. *arXiv preprint arXiv:2410.07933*, 2024.

Schmitz, P., Bold, L., Philipp, F. M., Rosenfelder, M., Eberhard, P., Ebel, H., and Worthmann, K. On excitation of control-affine systems and its use for data-driven koopman approximants. *arXiv preprint arXiv:2511.03734*, 2025.

Shahmoradi, Z. and Lee, T. Quantile inverse optimization: Improving stability in inverse linear programming. *Operations research*, 70(4):2538–2562, 2022.

Smith, E. M. and Pantelides, C. C. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex minlps. *Computers & chemical engineering*, 23(4-5):457–478, 1999.

Song, Y. and Scaramuzza, D. Learning high-level policies for model predictive control. in 2020 ieee. In *RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7629–7636.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. Feudal networks for hierarchical reinforcement learning. In *International conference on machine learning*, pp. 3540–3549. PMLR, 2017.

Wang, H.-n., Liu, N., Zhang, Y.-y., Feng, D.-w., Huang, F., Li, D.-s., and Zhang, Y.-m. Deep reinforcement learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 21(12):1726–1744, 2020.

Xie, R., Zhang, S., Wang, R., Xia, F., and Lin, L. Hierarchical reinforcement learning for integrated recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 4521–4528, 2021.

Yu-Geng, X., De-Wei, L., and Shu, L. Model predictive control—status and challenges. *Acta Automatica Sinica*, 39(3):222–236, 2013.

Zhang, F., Duan, J., Xu, H., Chen, H., Liu, H., Nie, S., and Li, S. E. Inverse model predictive control: Learning optimal control cost functions for mpc. *IEEE Transactions on Industrial Informatics*, 2024a.

Zhang, Z., Xiao, Y., Jiang, L., Yang, D., Yin, M., and Wang, P. Spatial-temporal interplay in human mobility: A hierarchical reinforcement learning approach with hypergraph representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 9396–9404, 2024b.

Zhao, W., He, T., Chen, R., Wei, T., and Liu, C. State-wise safe reinforcement learning: A survey. *arXiv preprint arXiv:2302.03122*, 2023.

Zhou, J., Yan, L., and Yang, K. Enhancing system-level safety in mixed-autonomy platoon via safe reinforcement learning. *IEEE Transactions on Intelligent Vehicles*, 2024.

A. Proof of Propositions

A.1. Proposition 3.5

The inverse feasible set for Problem (4)

$$\hat{\boldsymbol{\mathcal{X}}}_t^{\text{inv}}(\boldsymbol{u}_t) := \left\{ \hat{\boldsymbol{x}}_{\boldsymbol{t}} \mid \boldsymbol{u}_t \in \mathcal{U}_t^{\text{opt}}(\hat{\boldsymbol{x}}_{\boldsymbol{t}}) \right\}.$$

is not guaranteed to be non-empty, where $\mathcal{U}_t^{\text{opt}}(\hat{\boldsymbol{x}}_{\boldsymbol{t}})$ is defined by the KKT conditions of Problem (4).

Proof. We explore whether there exists $\boldsymbol{\theta}$ belonging to the inverse feasible set for Problem (4), which is making $\boldsymbol{u}_t$ to satisfy all KKT conditions.

- KKT conditions for Problem (4):
  1. Stationary Conditions:
  $$\boldsymbol{c}^T + \boldsymbol{\lambda}_t^T H + \boldsymbol{w}_t^T B = 0$$
  2. Complementary Slackness:
  $$\boldsymbol{\lambda}_t^T (H\boldsymbol{u}_t - h) = 0$$
  3. Dual Feasibility and Primal Feasibility:
  $$\boldsymbol{\lambda}_t \geq 0$$
  $$H\boldsymbol{u}_t \leq h$$

We assume $H\boldsymbol{u}_t \leq h$ is not active, which means $\boldsymbol{\lambda}_t = 0$. The system $B^T \boldsymbol{w}_t = -c$ has no feasible solution if the vector $-c$ does not lie in the column space of B^T, for example, when the system is overdetermined. Specifically, the overdetermined nature of the system arises from our design, where the sub-goal space is intentionally compressed relative to the action space. □

A.2. Proposition 3.6

The inverse feasible set for Problem (9)

$$\boldsymbol{\theta}_t^{*\text{inv}}(\boldsymbol{u}_t) := \left\{ \boldsymbol{\theta} \mid \boldsymbol{u}_t \in \mathcal{U}_t^{*\text{opt}}(\boldsymbol{\theta}) \right\}.$$

is always non-empty, where $\mathcal{U}_t^{*\text{opt}}(\boldsymbol{\theta})$ is defined by KKT optimality conditions of Problem (9).

Proof. We explore whether there exists $\boldsymbol{\theta}$ belonging to the inverse feasible set for Problem (9), which is making $\boldsymbol{u}_t$ to satisfy all KKT conditions.

- KKT conditions for Problem 9:
  1. Stationary Conditions:
  $$\boldsymbol{c}^T + \sum_{k=1}^{K} z_{kt} \boldsymbol{\theta}_k^T + \boldsymbol{\lambda}_t^T H + \boldsymbol{w}_t^T B = 0, \quad for\ t \in \mathcal{T}_e$$
  $$1 - z_{kt} - y_{kt} = 0, \quad for\ k = 1, ...K\ and\ t \in \mathcal{T}_e$$
  2. Complementary Slackness:
  $$z_{kt}(\tau_{kt} - \boldsymbol{\theta}_k^T \boldsymbol{u}_t + \nu_k) = 0, \quad for\ k = 1, ...K\ and\ t \in \mathcal{T}_e$$
  $$y_{kt}\tau_{kt} = 0, \quad for\ k = 1, ...K\ and\ t \in \mathcal{T}_e$$
  $$\boldsymbol{\lambda}_t^T (H\boldsymbol{u}_t - h) = 0, \quad for\ t \in \mathcal{T}_e$$

3. Dual Feasibility and Primal Feasibility:

$$z_{kt},\ y_{kt},\ \boldsymbol{\lambda}_t \geq 0, \quad for\ k = 1, ...K\ and\ t \in \mathcal{T}_e$$
$$H\boldsymbol{u}_t \leq h, \quad for\ t \in \mathcal{T}_e$$
$$\tau_{kt} \geq \boldsymbol{\theta}_k^T \boldsymbol{u}_t - \nu_k, \tau_{kt} \geq 0, \quad for\ k = 1, ...K\ and\ t \in \mathcal{T}_e$$

There exist trivial feasible solutions to satisfy all KKT conditions:
We assume $z_{kt} = 1$, $y_{kt} = 0$ and $\boldsymbol{\lambda}_t = \mathbf{0}$. Then the set of KKT conditions are transformed into

$$\begin{aligned} &\boldsymbol{c}^T + \sum_{k=1}^{K} \boldsymbol{\theta}_k^T + \boldsymbol{w}_t^T B = 0, \quad for\ t \in \mathcal{T}_e \\ &\tau_{kt} = \boldsymbol{\theta}_k^T \boldsymbol{u}_t - \nu_k,\ \tau_{kt} \geq 0, \quad for\ k = 1, ...K\ and\ t \in \mathcal{T}_e \end{aligned} \tag{14}$$

We must find $\{\boldsymbol{\theta}_k\}_{k=1}^K$ to satisfy $\boldsymbol{c}^T + \sum_{k=1}^K \boldsymbol{\theta}_k^T + \boldsymbol{w}_t^T B = 0$ and let $\nu_k = \min_{t=1}^T \{\boldsymbol{\theta}_k^T \boldsymbol{u}_t\}$ □

A.3. Proposition 3.7

By appropriately enlarging the decision space with auxiliary variables in Problem (9), any given set of expert decisions can be made optimal in the lifted space.

Proof. Consider a collection of expert decisions $\{u_i\}_{i=1}^N$ with $u_i \in \mathbb{R}^n$. Problem (9) can be reformulated as a linear program by introducing auxiliary variables $\{t_k\}_{k=1}^K$, which effectively enlarges the decision space from $\mathbb{R}^n$ to $\mathbb{R}^{n+K}$. For clarity, let us focus on the case $K = 1$. By adding the constraints

$$t_k \ \geq\ \boldsymbol{\theta}_k^\top u_i - \nu, \tag{15}$$

We can at least guarantee that all expert decisions $\{u_i\}_{i=1}^N$ lie on the same n-dimensional face of the feasible polytope in the augmented space by making the new added constraint active. This face can be made parallel to the hyperplane defined by the objective function by adjusting the new added constraint.

This situation, however, represents an extreme case. In practice, the set $\{u_i\}_{i=1}^N$ may lie within a face of dimension strictly smaller than n. To illustrate, suppose $u_i \in \mathbb{R}^2$. If all expert decisions lie on a one-dimensional face (an edge) of the feasible polytope, then they remain on a corresponding one-dimensional face in the lifted $\mathbb{R}^3$ space. Assume that this edge arises as the intersection of two adjacent facets, whose supporting hyperplanes have normal vectors $\vec{n}_1$ and $\vec{n}_2$, respectively. The direction vector of the edge is then given by

$$\vec{d}_{\text{edge}} = \vec{n}_1 \times \vec{n}_2, \tag{16}$$

since it is simultaneously orthogonal to both $\vec{n}_1$ and $\vec{n}_2$.

For the expert decisions to be optimal, it suffices that the objective hyperplane be parallel to this edge. Equivalently, the edge direction $\vec{d}_{\text{edge}}$ must be orthogonal to the objective normal $\vec{n}_{\text{obj}}$, i.e.,

$$\vec{d}_{\text{edge}} \cdot \vec{n}_{\text{obj}} = 0. \tag{17}$$

This condition establishes the required relationship between the optimal geometry and the objective orientation. Thus, we can solve the equation for feasible parameters θ in the terminal cost. □

A.4. Proposition 3.9

Problem (12) yields optimal solutions $\boldsymbol{u}_t^*$ satisfying $|f(\boldsymbol{u}_t^*) - f(\boldsymbol{u}_t)| \leq \epsilon_0$, s.t.

$$\|\boldsymbol{u}_t^* - \boldsymbol{u}_t\| \leq \sqrt{\frac{\epsilon_0}{l}} \tag{18}$$

Proof. First, we show that the feasible region of Problem (12) is non-empty. Since the strong duality constraint has been relaxed, the main task reduces to verifying the stationarity condition

$$\boldsymbol{c}^T + \boldsymbol{\lambda}_t^T H + \boldsymbol{w}_t^T B + \sum_k z_{kt}\boldsymbol{\theta}_k^T + 2l\boldsymbol{u}_t = 0, \qquad t = 1, \ldots, T. \tag{19}$$

For the i-th expert decision, suppose there are n_i active inequality constraints. Each such active constraint introduces one additional degree of freedom in the stationarity condition. Together with the N equality constraints, the total number of effective constraints across T time periods is

$$\sum_{i=1}^{T} n_i + NT. \tag{20}$$

Since the decision variable c lies in $\mathbb{R}^M$, the stationarity condition can be satisfied provided that the number of auxiliary terms is large enough to compensate for the remaining degrees of freedom. More precisely, the required number of ReLU terms is bounded above by

$$\max\left\{0,\, T - \frac{\sum_{i=1}^{T} n_i + NT}{M}\right\}. \tag{21}$$

This establishes the feasibility of the relaxed problem. Next, we derive the upper bound of distance.

Based on strong convexity of $f(u)$, we have

$$f(\boldsymbol{u}_t) \geq f(\boldsymbol{u}_t^*) + \nabla f(\boldsymbol{u}_t^*)^\top(\boldsymbol{u}_t - \boldsymbol{u}_t^*) + \frac{\mu}{2}\|\boldsymbol{u}_t - \boldsymbol{u}_t^*\|^2 \tag{22}$$

where $\nabla^2 f(\boldsymbol{u}_t) \geq \mu I > 0$.

By stationary condition $\nabla f(\boldsymbol{u}_t^*) + \lambda_t^{*T} H + w_t^{*T} B = 0$ and complementary slackness condition $\lambda_t^{*T}(Hu_t^* - h_t) = 0$,

$$\begin{aligned}\nabla f(\boldsymbol{u}_t^*)^\top(\boldsymbol{u}_t - \boldsymbol{u}_t^*) =& (-\boldsymbol{\lambda}_t^{*T} H - \boldsymbol{w}_{\boldsymbol{t}}^{*T} B)(\boldsymbol{u}_t - \boldsymbol{u}_t^*) = -\boldsymbol{\lambda}_{\boldsymbol{t}}^{*T} H(\boldsymbol{u}_t - \boldsymbol{u}_t^*)\\ =& \boldsymbol{\lambda}_t^{*T}(\boldsymbol{h_t} - H\boldsymbol{u}_t)\\ \geq& 0\end{aligned}$$

Thus,

$$f(\boldsymbol{u}_t) - f(\boldsymbol{u}_t^*) \geq \frac{\mu}{2}\|\boldsymbol{u}_t - \boldsymbol{u}_t^*\|^2 \tag{23}$$

The dual objective function $g(\boldsymbol{\lambda}_t, \boldsymbol{z}_t, \boldsymbol{y}_t, \boldsymbol{w}_t) \leq f(\boldsymbol{u}_t^*) \leq f(\boldsymbol{u}_t)$ and we have $f(\boldsymbol{u}_t) - g(\boldsymbol{\lambda}_t, \boldsymbol{z}_t, \boldsymbol{y}_t, \boldsymbol{w}_t) \leq \epsilon_0$ by constraining strong duality gap, s.t.

$$f(\boldsymbol{u}_t) - f(\boldsymbol{u}_t^*) \leq \epsilon_0 \tag{24}$$

Thus, we have

$$\|\boldsymbol{u}_t - \boldsymbol{u}_t^*\|^2 \leq \epsilon_0 * \frac{2}{\mu} = \frac{\epsilon_0}{l} \tag{25}$$

□

B. Details of Case Studies

We consider three simulation scenarios described as follows:

Autonomous Vehicle Rebalancing. Autonomous Mobility-on-Demand (AMoD) systems are an evolving mode of transportation in which a centrally coordinated fleet of self-driving vehicles dynamically serves travel requests. In real-world systems, the effectiveness of rebalancing strategies is central to the overall system performance, with sub-optimal strategies potentially exacerbating congestion through unnecessary trips or increased passenger waiting time. The control of these systems is typically formulated as a large network optimization problem. This framework comprises two stages: (1) determining the desired distribution of idle vehicles $\hat{\boldsymbol{q}}_t$ through the use of the learned policy $\pi_\phi(\hat{\boldsymbol{q}}_t|s_t)$ by reinforcement learning, (2) converting this distribution to a passenger flow g_{ij}^t and rebalancing flow f_{ij}^t by solving a linear control problem.

Supply Chain Inventory Management. In the supply chain inventory management task, we aim to determine the optimal ordering and distribution strategies across a network of interconnected multiple warehouses and multiple retail stores to satisfy customer demand while minimizing storage, production and transportation costs. We choose upper-level goals as (i) desired production in warehouse nodes $\hat{w}_i^t$ and (ii) desired inventory in store nodes $\hat{q}_i^t$. And the lower-level optimization module determines the amount of commodities $\boldsymbol{w}_i^t$ to order in each warehouse, and the shipping flow f_{ij}^t from warehouses to stores.

Mobile Robot Navigation. In the mobile robot navigation task, a ground robot moves from a start location to a target destination while avoiding static obstacles. The control problem is typically formulated in a hierarchical framework. At the high level, a learned policy $\pi_{\boldsymbol{\theta}}(\hat{p}_t \mid s_t)$ outputs intermediate waypoints, conditioned on the robot's state, the goal position, and obstacle information. At the low level, these actions are converted into executable control inputs (v_t, ω_t) for the unicycle dynamics, ensuring feasibility under kinematic and dynamic constraints.

The detailed formulations of the lower-level optimization problems for these three environments are given as follows.

B.1. Autonomous Vehicles Rebalancing

The lower-level optimization policy is represented in a matrix form as follows.

$$\begin{aligned}
\min_{\boldsymbol{f}_t, \boldsymbol{g}_t} \quad & \boldsymbol{c}^\top \boldsymbol{f}_t - \boldsymbol{p}^\top \boldsymbol{g}_t + \sum_{k=1}^{K} \max\{-\boldsymbol{\theta}_f^{(k)\top} \boldsymbol{f}_t + \boldsymbol{\theta}_g^{(k)\top} \boldsymbol{g}_t - \mu^{(k)}, 0\} \\
\text{s.t.} \quad & \boldsymbol{q}_t - A(\boldsymbol{f}_t + \boldsymbol{g}_t) \geq \hat{\boldsymbol{q}}_t \\
& \boldsymbol{q}_t - G(\boldsymbol{f}_t + \boldsymbol{g}_t) \geq \boldsymbol{0} \\
& \boldsymbol{f}_t \geq \boldsymbol{0}, \quad \boldsymbol{g}_t \geq \boldsymbol{0} \\
& \boldsymbol{g}_t \leq \boldsymbol{\lambda}_t
\end{aligned} \tag{26}$$

Inequation 26, the vectors $\boldsymbol{c}$ and $\boldsymbol{p}$ characterize the unit operational cost and unit profit, respectively. The constraints incorporate two critical matrices: the incidence matrix A, which governs the flow conservation and node-wise distribution updates, and the topology matrix G, which ensures the total outgoing flow does not exceed the available local supply. The term $\boldsymbol{\lambda}_t$ captures the real-time travel requests. Beyond the original stage cost, the parameters $\{\boldsymbol{\theta}_f^{(k)}, \boldsymbol{\theta}_g^{(k)}, \mu^{(k)}\}_{k=1}^K$ define a structured strategic cost learned via inverse optimization from expert demonstrations.

Proposition B.1. *The inverse feasible set for Problem 26*

$$\boldsymbol{\theta}_t^{*inv}(\mathbf{f}_t, \mathbf{g}_t) := \left\{ \boldsymbol{\theta} \;\middle|\; \mathbf{f}_t, \mathbf{g}_t \in \mathcal{X}_t^{*opt}(\boldsymbol{\theta}) \right\}. \tag{27}$$

is always non-empty, where $\mathcal{X}_t^{*opt}(\boldsymbol{\theta})$ is defined by KKT optimality conditions of Problem 26.

Next, we will prove this proposition and also show that the solved parameters have practical significance which has not been discussed further in the general setting.

Proof. The inverse optimization solving for unknown parameters in Problem 26 is formulated as follows

$$\begin{gathered}
\min_{\boldsymbol{x}_t,\boldsymbol{y}_t,\boldsymbol{w}_t,\boldsymbol{v}_{1t},\boldsymbol{v}_{2t},\boldsymbol{z}_{1t},\boldsymbol{z}_{2t},\boldsymbol{\tau}_t,\boldsymbol{\theta},\mu} \sum_k \|(\boldsymbol{\theta}^{(k)})\|^2 \\
1-\boldsymbol{z}_{1t}^{(k)}-\boldsymbol{z}_{2t}^{(k)}=0,\ k=1,...K,\ t=1,...T \\
\boldsymbol{c}^T+\boldsymbol{x}_t^T A+\boldsymbol{y}_t^T G-\boldsymbol{w}_t^T-\sum_k \boldsymbol{z}_{1t}^{(k)}\boldsymbol{\theta}_f^{(k)T}+2l_1\boldsymbol{f}_t=0,\ t=1,...T \\
-\boldsymbol{p}^T+\boldsymbol{x}_t^T A+\boldsymbol{y}_t^T G-\boldsymbol{v}_{1t}^T+\boldsymbol{v}_{2t}^T+\sum_k \boldsymbol{z}_{1t}^{(k)}\boldsymbol{\theta}_g^{(k)^T}+2l_2\boldsymbol{g}_t=0,\ t=1,...T \\
\boldsymbol{x}_t^T(\boldsymbol{q}_t-A(\boldsymbol{f}_t+\boldsymbol{g}_t)-\hat{\boldsymbol{q}}_t)=0,\ t=1,...T \\
\boldsymbol{y}_t^T(\boldsymbol{q}_t-G(\boldsymbol{f}_t+\boldsymbol{g}_t)=0,\ t=1,...T \\
\boldsymbol{w}_t^T\boldsymbol{f}_t=0,\ t=1,...T \\
\boldsymbol{v}_{1t}^T\boldsymbol{g}_t=0,\ t=1,...T \\
\boldsymbol{v}_{2t}^T(\boldsymbol{g}_t-\boldsymbol{\lambda}_t)=0,\ t=1,...T \\
\text{(other feasiblity constraints)}
\end{gathered} \tag{28}$$

Given historical data $\{\boldsymbol{f}_i,\boldsymbol{g}_i,\boldsymbol{q}_i,\boldsymbol{\lambda}_I\}_{i=1}^N$, we consider the worst-case scenario where

$$G\boldsymbol{f}_i+G\boldsymbol{g}_i\neq\boldsymbol{q}_i,\quad \boldsymbol{f}_i\neq 0,\quad \boldsymbol{g}_i\neq 0,\quad \boldsymbol{g}_i\neq\boldsymbol{\lambda}_I\quad \forall i=1,\dots,N.$$

This implies that all corresponding dual variables

$$\boldsymbol{y}_i=\boldsymbol{w}_i=\boldsymbol{v}_{1i}=\boldsymbol{v}_{2i}=0\quad \forall i=1,\dots,N.$$

We proceed with the analysis under this assumption. Moreover, We further set

$$\boldsymbol{z}_{1i}^{(k)}=1,\ \boldsymbol{z}_{2i}^{(k)}=0$$

which simplifies inverse problem corresponding to Problem 26 into the following form:

$$\begin{aligned}
\min_{\boldsymbol{\theta}^{(k)},\mu^{(k)},\boldsymbol{x}_i}\quad & \sum_k \|\boldsymbol{\theta}^{(k)}\|^2 \\
\text{s.t.}\quad & -\boldsymbol{\theta}_f^{(k)\top}\boldsymbol{f}_i+\boldsymbol{\theta}_g^{(k)\top}\boldsymbol{g}_i-\mu^{(k)}=\tau_i^{(k)},\quad \forall k=1,\dots,K, \\
& \boldsymbol{c}^\top+\boldsymbol{x}_i^\top A-\sum_k \boldsymbol{\theta}_f^{(k)\top}=0, \\
& -\boldsymbol{p}^\top+\boldsymbol{x}_i^\top A+\sum_k \boldsymbol{\theta}_g^{(k)\top}=0, \\
& \text{(other feasibility constraints).}
\end{aligned} \tag{29}$$

Now assume $\boldsymbol{x}_i\preceq\epsilon\mathbf{1}$ for some small $\epsilon>0$. Then we have

$$-\epsilon\mathbf{1}\preceq\boldsymbol{x}_i^\top A\preceq\epsilon\mathbf{1}.$$

Suppose $\mathbf{c},\mathbf{p}\in\mathbb{R}_{++}^n$. We can select ϵ such that

$$\min_{1\le j\le M} c_j\ge\epsilon,\qquad \min_{1\le j\le M} p_j\ge\epsilon.$$

Then the constraints in 29 reduce to

$$\begin{aligned}
-\boldsymbol{\theta}_f^{(k)\top}\boldsymbol{f}_i+\boldsymbol{\theta}_g^{(k)\top}\boldsymbol{g}_i-\mu^{(k)}&\ge 0,\quad \forall k=1,\dots,K, \\
\boldsymbol{c}^\top+\boldsymbol{x}_i^\top A-\sum_k \boldsymbol{\theta}_f^{(k)\top}&=0, \\
-\boldsymbol{p}^\top+\boldsymbol{x}_i^\top A+\sum_k \boldsymbol{\theta}_g^{(k)\top}&=0.
\end{aligned} \tag{30}$$

Assume further that $K = 1$ (for simplicity). Then the last two equalities yield:

$$\boldsymbol{\theta}_f = c + A^\top \boldsymbol{x}_i, \qquad \boldsymbol{\theta}_g = p - A^\top \boldsymbol{x}_i.$$

In the context of autonomous vehicle rebalancing, it is reasonable to assume that the total revenue exceeds the cost, i.e., $\boldsymbol{p}^\top \boldsymbol{g}_i > \boldsymbol{c}^\top \boldsymbol{f}_i$. Then we compute:

$$\begin{aligned} -\boldsymbol{\theta}_f^\top \boldsymbol{f}_i + \boldsymbol{\theta}_g^\top \boldsymbol{g}_i - \mu &= -(\boldsymbol{c}^\top + \boldsymbol{x}_i^\top A)\boldsymbol{f}_i + (\boldsymbol{p}^\top - \boldsymbol{x}_i^\top A)\boldsymbol{g}_i - \mu \\ &= (\boldsymbol{p}^\top \boldsymbol{g}_i - \boldsymbol{c}^\top \boldsymbol{f}_i) - \boldsymbol{x}_i^\top A(\boldsymbol{f}_i + \boldsymbol{g}_i) - \mu \\ &\geq (\boldsymbol{p}^\top \boldsymbol{g}_i - \boldsymbol{c}^\top \boldsymbol{f}_i) - \epsilon \mathbf{1}^\top (\boldsymbol{f}_i + \boldsymbol{g}_i) - \mu. \end{aligned} \tag{31}$$

Therefore, we can choose ϵ sufficiently small and μ accordingly, such that the right-hand side of 31 remains non-negative. This guarantees that the constraints in 30 are satisfied, hence the problem admits a feasible solution. □

Problem 26 is constructed so that the historical expert decisions $\{\boldsymbol{f}_i, \boldsymbol{g}_i\}_{i=1}^N$ are optimal solutions to the forward problem. There exist model parameters $\{\boldsymbol{\theta}_f^{(k)}, \boldsymbol{\theta}_g^{(k)}, \mu^{(k)}\}_{k=1}^K$ and desired next states $\{\hat{\boldsymbol{q}}_i\}_{i=1}^N$, which can be learned via reinforcement learning, such that the expert decisions become approximately optimal.

B.2. Supply Chain Inventory Management

The lower-level policy is formulated as follows

$$\min_{f_{ij}^t, \boldsymbol{w}_i^t} \quad \sum_{i \in V_W} m_i^O \cdot \boldsymbol{w}_i^t + \sum_{(i,j)\in\mathcal{E}} m_{ij}^T \cdot f_{ij}^t + \sum_{i \in V_S} \rho_i^f |\epsilon_i^f| + \sum_{i \in V_W} \rho_i^g |\epsilon_i^w| + \sum_{k=1}^K \max\{-\boldsymbol{\theta}_k^T \boldsymbol{f}^t + \mu_k, 0\} + \sum_{k=1}^K \max\{-\boldsymbol{\beta}_k^T \boldsymbol{w}^t + \nu_k, 0\} \tag{32a}$$

$$\text{s.t.} \quad \sum_{j \in N^-(i)} f_{ji}^t + \epsilon_i^f = \hat{q}_i^{t+1}, i \in V_S \tag{32b}$$

$$\boldsymbol{q}_i^t + \sum_{j \in N^-(i)} f_{ji}^t - d_i^t \leq c_i^t, i \in V_S \tag{32c}$$

$$\sum_{j \in N^+(i)} f_{ij}^t \leq \boldsymbol{q}_i^t, i \in V_W \tag{32d}$$

$$\boldsymbol{q}_i^t + w_i^t - \sum_{j \in N^+(i)} f_{ij}^t \leq c_i^t, i \in V_W \tag{32e}$$

$$w_i^t + \epsilon_i^w = w_i^t, i \in V_W \tag{32f}$$

$$f_{ij}^t \geq 0, (i,j) \in \mathcal{E}, w_i^t \geq 0, i \in V_W \tag{32g}$$

where m^o, m^T, d, c are production cost, transportation cost, customer demand and capacity respectively, w_i denotes order quantity in warehouse i and f_{ij} denotes the departing flow from warehouse i to store j, and q denotes the inventory level. Specifically, ϵ quantifies the magnitude of constraint violations, and the parameters β, μ, and ρ are estimated through the inverse optimization process to align the model with expert data.

B.3. Mobile Robot

$$\min_{\boldsymbol{u}_t=(v_t,\omega_t)} \quad c_p\|\boldsymbol{p}_{t+1}-\hat{\boldsymbol{p}}_t\|^2 + c_h(\theta_{t+1}-\hat{\theta}_t)^2 + c_u\|\boldsymbol{u}_t\|^2 + \sum_{k=1}^{K}\max\{\boldsymbol{\alpha}_k^\top \boldsymbol{u}_t - \beta_k, 0\} \tag{33a}$$

$$\text{s.t.} \quad x_{t+1} = x_t + T_s v_t \cos\theta_t \tag{33b}$$

$$y_{t+1} = y_t + T_s v_t \sin\theta_t \tag{33c}$$

$$\theta_{t+1} = \theta_t + T_s \omega_t \tag{33d}$$

$$v_{\min} \le v_t \le v_{\max} \tag{33e}$$

$$\omega_{\min} \le \omega_t \le \omega_{\max} \tag{33f}$$

$$\|\boldsymbol{p}_{t+1} - \boldsymbol{p}_j^{\text{obs}}\| \ge r_j + d_{\text{safe}}, \quad \forall j = 1,\dots,M \tag{33g}$$

Here, $v_t \in \mathbb{R}$ and $\omega_t \in \mathbb{R}$ denote the linear speed and angular velocity of the mobile robot, respectively. The vector $\boldsymbol{p}_t = (x_t, y_t)^\top \in \mathbb{R}^2$ represents the robot's position in the 2D Cartesian plane, while $\theta_t \in [-\pi, \pi)$ denotes its orientation. The term $\hat{\boldsymbol{p}}_t$ is the reference waypoint generated by the upper-level RL policy. In the objective function (33a), c_p, c_h, and c_u are weighting coefficients for position tracking, heading alignment, and control effort. The parameters $\{\alpha_k, \beta_k\}_{k=1}^K$ characterize the structured safety cost learned via inverse optimization from expert demonstrations. In the constraints, T_s is the sampling time, while $[v_{\min}, v_{\max}]$ and $[\omega_{\min}, \omega_{\max}]$ define the kinematic limits of the unicycle model. Lastly, $\boldsymbol{p}_j^{\text{obs}}$, r_j, and d_{safe} represent the position of the j-th obstacle, its radius, and the required safety margin, respectively.

C. Further Experimental Results and Discussions

C.1. RL Methodology and Experimental Setup

- **Autonomous Vehicle Rebalancing**: For the Autonomous Vehicle Rebalancing experiment, we train an A2C-GNN agent as in (Gammelli et al., 2023) using on-policy updates with a discount factor of 0.99, a maximum of 50 decision steps per episode, and 10000 training episodes. The actor and critic share the same EdgeConv-based GNN encoder and each head is implemented as a two-layer multilayer perceptron with 256 hidden units, and both networks are optimized with the Adam optimizer with a learning rate of 5×10^{-5}. At the end of each episode we normalize the discounted returns, perform a single A2C update of actor and critic parameters with gradient-norm clipping at 5 to improve stability, and we save the checkpoint achieving the highest cumulative training reward for later evaluation.
- **Inventory Management**: We adopt a graph neural network-based Advantage Actor–Critic (A2C) method as in (Gammelli et al., 2023) to learn the control policy. Training is performed with an on-policy A2C update, using a discount factor of 0.99, a maximum episode length of 30 steps, and up to 20,000 training episodes. Both actor and critic are optimized with Adam with learning rate 5e-5, and gradient norm clipping is applied to improve training stability.
- **Mobile Robot Navigation**: For the mobile robot, we use an off-policy Soft Actor–Critic (SAC) algorithm. SAC is trained with a replay buffer of size 10^6, a mini-batch size of 256, Adam optimizer with learning rate 3×10^{-4} for both actor and critic, discount factor $\gamma = 0.99$, soft target network updates with $\tau = 0.005$, and automatic entropy tuning with target entropy set to $-\dim(\mathcal{A})$. Gradient updates start once the replay buffer contains more than 400 transitions, and the agent is trained for 500 episodes, periodically saving the actor network during training.

C.2. Performance under Non-Convex True Cost

To address the performance of our framework under non-convex settings, we conducted a supplementary experiment in the mobile robot navigation domain. Specifically, we introduced a non-convex obstacle avoidance penalty into the overarching true cost function, modeled as a Gaussian repulsive field—a widely adopted formulation in the robotics literature to represent smooth, non-convex spatial constraints:

$$C_{obs}(p) = \sum_{i=1}^{M} A_i \exp\left(-\frac{\|p - o_i\|^2}{2\sigma_i^2}\right) \tag{34}$$

where p is the robot's position, o_i is the obstacle's position, A_i is the penalty amplitude, and σ_i controls the field's range.

Empirical results, as summarized in Table 4, demonstrate that the learned convex surrogate remains highly effective. Specifically, the subgoal-guided terms and the learned ReLU-based regularization terms successfully steer the robot to safely bypass obstacles and efficiently reach its destination. Notably, the baseline bi-level method lacking this ReLU-based term requires more manual effort to tune the hyperparameters within the cost function.

Table 4. Performance Comparison on Mobile Robot Navigation under Non-Convex True Cost

Method	**Average Nav Time (s)**	**Average Path Length (m)**	**Energy (J)**
End-to-end RL	7.45	4.27	6.80
Bi-level-original	8.10	3.82	4.93
Bi-level-learned (Ours)	**4.45**	**3.93**	**2.63**

C.3. Reward-Shaping Sensitivity

We have conducted an explicit reward-shaping sensitivity experiment on the mobile robot navigation task. By varying the weights in the reward function, we observe that our method still remains robust according to 5, 6 and 7.

Across all three case studies we employ standard RL methods (GNN-A2C and SAC) with conventional hyperparameters and reward designs that closely mirror the MPC stage costs. In the revised version, we will clarify that we use off-the-shelf RL implementations with fixed hyperparameters and reward shaping across all variants, so that the comparison cleanly isolates the impact of the learned lower-level FOP.

C.4. Motivations for Inverse Optimization Approach

We justify our choice of the proposed inverse optimization approach from three perspectives: the structural design of the cost function and the ability to extract and interpert the behavior pattern contained in expert demonstrations, and present the efficacy of the proposed inverse optimizaiton-guided RL-OC framework compared with purely imitation learning.

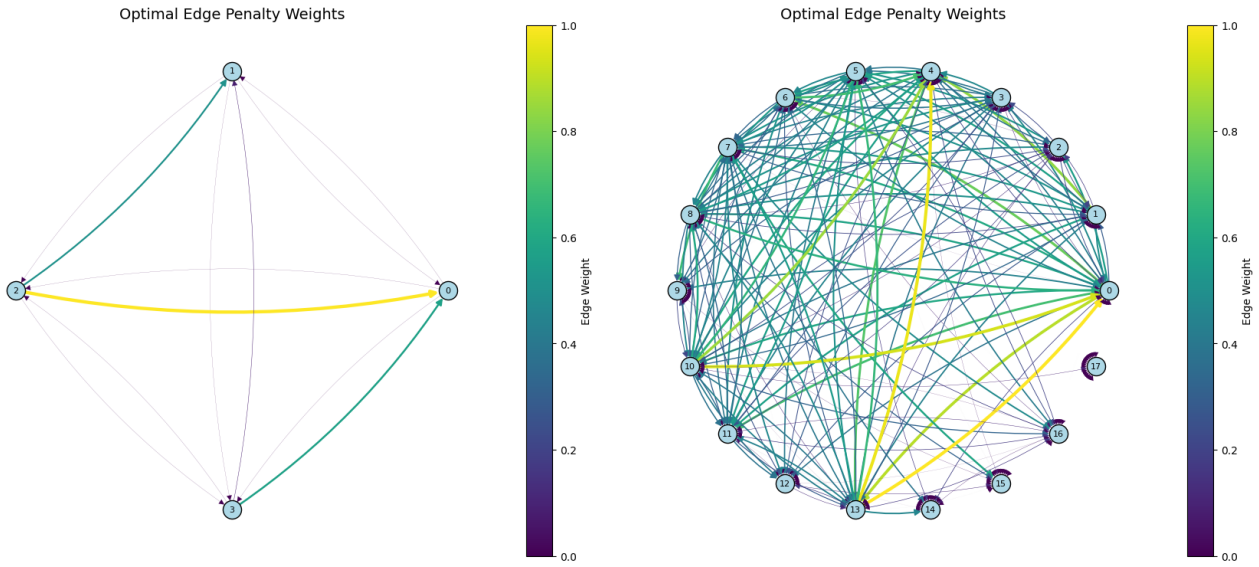
C.4.1. Structural Design Guided by Theoretical Properties

The primary motivation for adopting the inverse optimization framework lies in its ability to theoretically inform the structure of the lower-level cost function. Standard approaches often heuristically assume a fixed cost form (e.g., quadratic). We implement the value approximation method proposed by Abdufattokhov et al. (2021), where quadratic terminal cost is estimated via value functions and inverse MPC method proposed by (Zhang et al., 2024a) which assumes quadratic structure of terminal cost. To ensure a rigorous comparison, we further implement a baseline termed IMPC-bi-level, which integrates the terminal cost identified by Inverse MPC into the structured RL-OC framework. We compare the performance on autonomous vehicle rebalancing task. As table 8 suggests, our proposed framework outperforms all three baselines.

C.4.2. Interpretability

Interpretations of Optimal Parameters. The optimal parameters inferred by inverse optimization are visualized in the following figures, which show that different weights are assigned to different edges. For example, the district ”0” of higher value motivates the repositioning of idle vehicles towards this area, particularly along paths with low travel costs. Thus, a significant penalty is applied to insufficient rebalancing flows on the corresponding routes. The yellow line in the figure, which denotes high inferred penalty weights, achieves this motivation. In scenario 2, the situation becomes more complex, but we can still notice some ”important” routes and the ”not important” ones are not penalized.

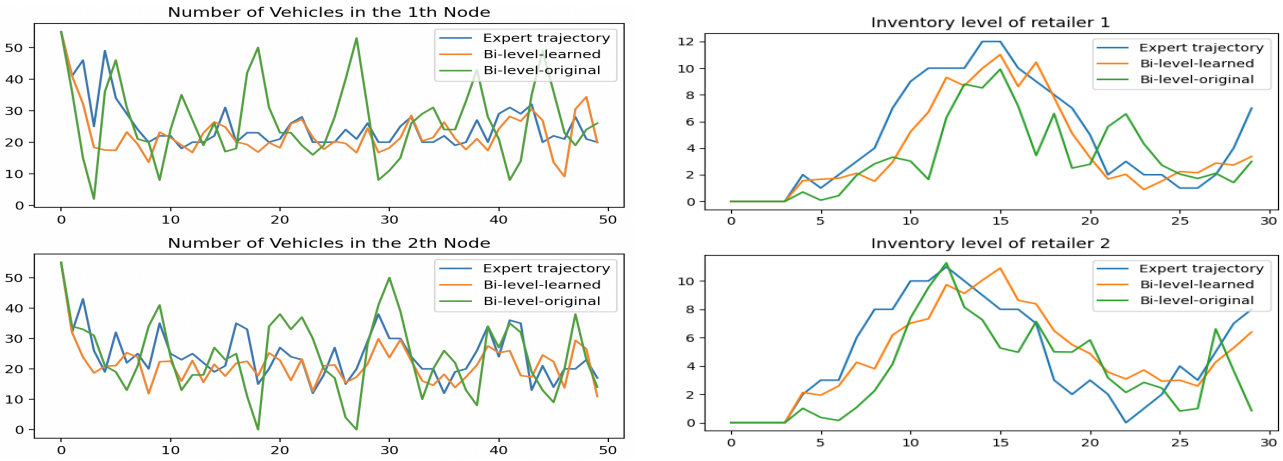
Interpretations of Outputs. To gain deeper insights into the effectiveness of the inverse method and explain model interpretability, we analyze how decisions made by two bi-level frameworks deviate from trajectories made by MPC. Table 9 shows that Bi-level-learned framework make decisions much closer to MPC trajectories, which is measured by cosine similarity and Manhattan Distance. And Figure 4 shows the number of vehicles in the AV network and the inventory level over time. Similarly, our framework produces a trajectory that is closer to the MPC solution. It indicates that the learned lower-level policy based on expert demonstrations extracts information efficiently compared with the original policy with short-sighted cost functions.



(a) Optimal Weights in Scenario 1

(b) Optimal Weights in Scenario 2

Figure 3. Visualizations of optimal parameters solved by inverse optimization



(a) Number of vehicles in nodes across different methods

(b) Inventory level of retailers in chosen nodes across different methods

Figure 4. Comparison of system state trajectories in AV Rebalancing and SCIM across different methods

Table 5. Performance Comparison on Mobile Robot Navigation ($c_u = 0.01$).

Method	Travel Time (s)	Path Length (m)	Energy (J)
MPC	3.50 ± 0.00	3.50 ± 0.00	4.81 ± 0.00
End-to-end RL	7.60 ± 0.11	4.29 ± 0.02	6.78 ± 0.23
Bi-level-unchanged	9.14 ± 0.08	4.35 ± 0.02	6.20 ± 0.13
Bi-level-learned (Ours)	$\mathbf{4.82 \pm 0.30}$	$\mathbf{4.16 \pm 0.19}$	$\mathbf{2.92 \pm 1.04}$

Table 6. Performance Comparison on Mobile Robot Navigation ($c_u = 0.05$).

Method	Travel Time (s)	Path Length (m)	Energy (J)
MPC	3.50 ± 0.00	3.50 ± 0.00	4.81 ± 0.00
End-to-end RL	7.60 ± 0.10	4.27 ± 0.00	6.84 ± 0.31
Bi-level-unchanged	9.10 ± 0.10	4.31 ± 0.02	6.41 ± 0.20
Bi-level-learned (Ours)	$\mathbf{4.40 \pm 0.10}$	$\mathbf{4.14 \pm 0.07}$	$\mathbf{1.52 \pm 0.06}$

C.4.3. Data efficiency

We further benchmark against imitation learning (IL) to demonstrate the superior sample efficiency of our approach. We implemented a set of imitation-learning baselines trained on varying numbers of expert demonstrations, and considered several supervised learning architectures as candidate policy models, including multilayer perceptrons (MLPs), Random Forest regressors, and k-Nearest Neighbors (kNN). Across the configurations we tested, these methods achieved broadly similar performance. As shown in the table 10, our bi-level framework consistently outperforms these imitation-learning baselines and maintains strong performance even when the number of expert demonstrations is relatively small, indicating favorable sample efficiency.

Table 10. Performance comparison with imitation learning

Data Size	**MPC**	**Inv-Bi-level**	**Imitation Learning**
Small ($N = 100$)	$35725(\pm 41.6)$	$35019(\pm 53.0)$	$33606(\pm 199.6)$
Medium ($N = 500$)	*	*	$33709(\pm 216.3)$
Large ($N = 2000$)	*	*	$33285(\pm 203.6)$

Note: The asterisks (*) indicate that experiments were not required for these configurations. The MPC baseline is independent of the training dataset size. For our Inv-Bi-level approach, it effectively recovers the underlying cost function with minimal data, rendering larger datasets redundant.

C.5. Sensitivity Analysis

In this subsection, we analyze how the number of ReLU terms included in the cost function and the noise in expert data influence the performance of inverse optimization. We use cosine similarity and Manhattan distance between the noise-free expert decisions and decisions made by solving the lower-level optimization problem with different settings.

- Sensitivity to Number of ReLU Terms

Based on Figure 5, we notice that the quality of solutions doesn't vary a lot as number of ReLU terms varies. We can infer well-performed parameters via inverse optimization utilizing only a handful of expert decisions and very few ReLU terms.

- Sensitivity to Data Noise

Sometimes given demonstrations are not optimal, thus we figure out how the noise affect inverse optimization. As figure 6 shows, the solution is much closer to optimal expert demonstrations although some noise exists in offline data. Moreover, when we utilize noisy expert demonstrations to derive lower-level problem formulation and apply it in the bi-level framework,

Table 7. Performance Comparison on Mobile Robot Navigation ($c_u = 0.1$).

Method	Travel Time (s)	Path Length (m)	Energy (J)
MPC	3.50	3.50	4.81
End-to-end RL	7.53 ± 0.06	4.28 ± 0.01	6.85 ± 0.19
Bi-level-unchanged	9.07 ± 0.06	4.33 ± 0.02	6.02 ± 0.03
Bi-level-learned (Ours)	$\mathbf{4.00 \pm 0.00}$	$\mathbf{3.93 \pm 0.01}$	$\mathbf{2.84 \pm 0.06}$

Table 8. Performance Comparison on Autonomous Vehicle (AV) Rebalancing Task.

Method	**Reward**	**Served Demand**
MPC (Oracle)	35725 ($\pm$41.6)	3203 ($\pm$3.07)
Value Approximation	24774 ($\pm$213.5)	2395 ($\pm$14.9)
IMPC	32426 ($\pm$401.2)	2996 ($\pm$28.6)
IMPC-bi-level	33431 ($\pm$320.4)	3044 ($\pm$25.3)
Bi-level-learned (Ours)	**35019** ($\pm$53.0)	**3141** ($\pm$6.14)

Table 9. Comparison of the cosine similarity and Manhattan distance between the rebalancing flow f obtained from different bi-level models and the flow generated by the MPC.

Formulation	**Cosine Similarity**	**Manhattan Distance**
Bi-level-original	0.477	44
Bi-level-learned	**0.976**	**7**

the framework improves the reward compared to the one achieved by original noisy expert demonstrations by 1.7%. Although the improvement is not so significant, it suggests a promising direction that we can employ our inverse optimization-guided bi-level framework to extract insights from sub-optimal offline data and make better decisions.

C.6. Scalability Discussions

C.6.1. Solve-time Scaling for Inverse Optimization

We systematically vary the problem dimension, the number of ReLU terms in the objective, and the number of expert demonstrations, and plot the resulting wall-clock solve times as shown in Figure 7.

We have observed that solve-time increases predictably with problem dimensions and the number of demonstrations, remaining tractable for typical system sizes without imposing prohibitive bottlenecks. The primary driver of complexity is the number of ReLU terms (K). While $K = 1$ is computationally lightweight, increasing K introduces significant combinatorial complexity. To address this, we reformulate the inverse problem as a Mixed-Integer Program (MIP) by treating the auxiliary dual variables associated with ReLU activation as binary. This reformulation is mathematically rigorous—aligning with the inherent discrete logic of ReLU functions—and leverages efficient branch-and-bound solvers to drastically accelerate convergence without compromising solution quality. We also note that solve times can vary even for instances of identical size, driven by the specific geometry of the expert demonstrations and the complexity of the active constraint set required to rationalize the data.

C.6.2. Learning Structured Costs at Scale

We also evaluate a gradient-based alternative that keeps the same ReLU-based terminal-cost structure but learns its parameters by embedding the lower-level problem as a differentiable convex program via CVXPYLayers. The learned cost d_θ is then used in our bi-level RL-OC framework. Table 11 and 12 reports results on autonomous vehicle rebalancing, supply chain inventory management and mobile robot navigation. Compared with solving the inverse problem to high precision—which can be challenging at larger scales—this approach enables efficient differentiating through the optimization layer. Empirically, CVXPYLayers performs slightly worse than inverse optimization but remains comparable in AV rabalancing and SCIM tasks, which highlights that our structured cost design is effective and compatible with different solution pipelines. However,

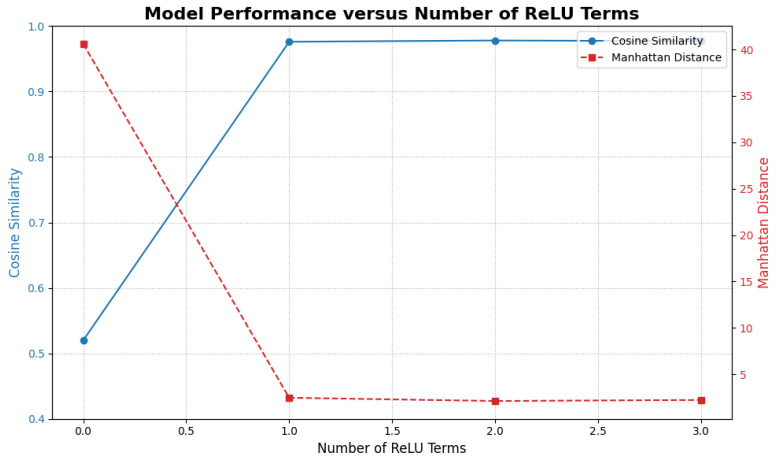


Figure 5. Sensitivity to the Number of ReLU Terms

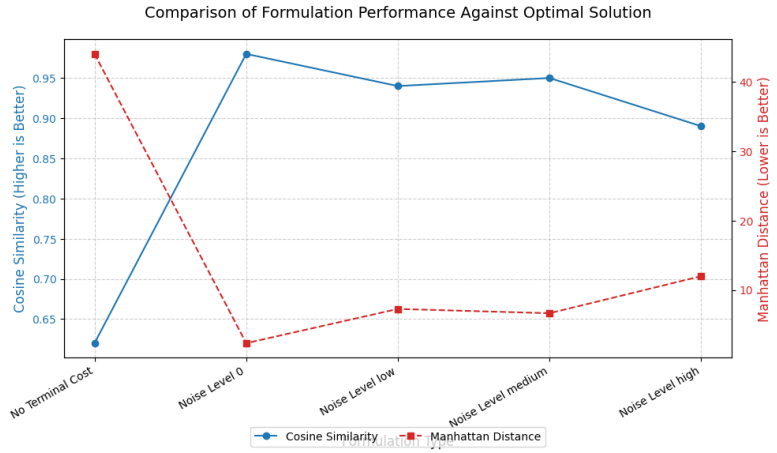


Figure 6. Sensitivity to Noise in Offline Data

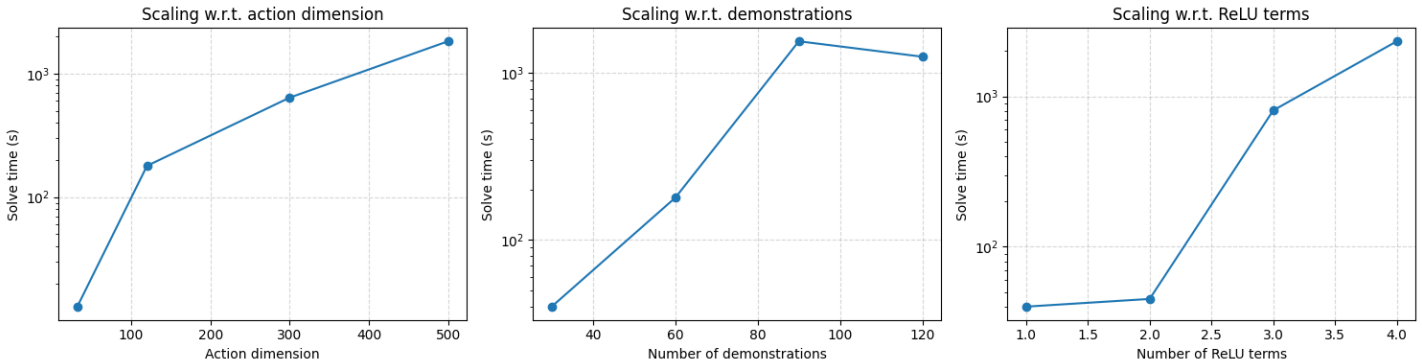


Figure 7. Solve-time Scaling Curve

Table 11. Performance comparison on two tasks

	AV Rebalancing		Supply Chain Inventory Management	
Method	**Reward**	**Served Demand**	**Reward**	**Served Demand**
MPC	35725 ($\pm$41.6)	3203 ($\pm$3.07)	11335 ($\pm$34.3)	1337 ($\pm$5.61)
Bi-level-cvxpy	34403 ($\pm$108.7)	3086 ($\pm$7.06)	9268 ($\pm$32.1)	912 ($\pm$2.20)
Bi-level-learned	**35019** ($\pm$53.0)	**3141**	**9442** ($\pm$42.3)	**930** ($\pm$1.17)

performance of such gradient-based methods degrades in mobile robot navigation. This outcome reveals a distinct trade-off: while it hints at the potential of differentiable layers to solve for parameters at scale, it also implies that gradient-based methods might fail to identify underlying structures that yield generalizable insights.

Table 12. Performance Comparison on Mobile Robot Navigation.

Method	**Travel Time (s)**	**Path Length (m)**	**Energy (J)**
MPC	3.50	3.50	4.81
End-to-end RL	7.60 ± 0.11	4.29 ± 0.02	6.78 ± 0.23
Bi-level-cvxpy	11.2 ± 0.20	4.57 ± 0.03	7.00 ± 0.02
Bi-level-learned (Ours)	$\mathbf{4.82} \pm 0.30$	$\mathbf{4.16} \pm 0.19$	$\mathbf{2.92} \pm 1.04$

An asterisk (*) indicates that no valid result was obtained for this method.