Highlights

Model-Based Reinforcement Learning for Heterogeneous Multi-Robot Task Assignment Under Distribution Shifts

Daniel Garces, Sara Castro, Adrian Haimovich, Byron Crowe, Stephanie Gil

- Online task assignment for heterogeneous robots with service windows.
- Prediction-aware rollout uses future requests without committing to them.
- Adaptive weights reduce the impact of unreliable demand forecasts.
- Selective re-optimization revises assigned but unstarted requests.
- Hospital case study shows lower wait times and near-complete service.

Model-Based Reinforcement Learning for Heterogeneous Multi-Robot Task Assignment Under Distribution Shifts

Daniel Garces[a,*], Sara Castro[b], Adrian Haimovich[b,c], Byron Crowe[d] and Stephanie Gil[a]

[a]John A. Paulson School Of Engineering And Applied Sciences, Harvard University, 150 Western Avenue, Boston, 02134, MA, USA
[b]Harvard Medical School, 77 Avenue Louis Pasteur, Boston, 02215, MA, USA
[c]Beth Israel Deaconess Medical Center, 330 Brookline Avenue, Boston, 02215, MA, USA
[d]Stanford University School of Medicine, 300 Pasteur Drive, Stanford, 94305, CA, USA

ARTICLE INFO

Keywords:
Heterogeneous Task Allocation
Multi-Agent Model-Based Reinforcement Learning
Sequential Decision Making
Planning Under Uncertainty
Online Route Optimization
Event Prediction
Autonomous Robots

ABSTRACT

Heterogeneous multi-robot service systems must assign requests to compatible robots, construct feasible schedules, and adapt as new tasks arrive online. Historical data can help anticipate future demand, but relying too heavily on inaccurate predictions can degrade performance under distribution shifts. We develop a prediction-aware adaptive rollout framework for heterogeneous multi-robot task assignment with scheduled and real-time requests. The problem is formulated as a finite-horizon stochastic dynamic program incorporating robot-task compatibility, ordered service requirements, routing constraints, service windows, and end-of-horizon return requirements. The proposed policy evaluates current assignments using sampled future request scenarios while restricting immediate commitments to requests already observed. To enable online use, the framework combines pruned candidate controls, wait actions, and an interaction-aware base policy for efficient future-cost estimation. Robustness to forecast error is provided by adaptively reweighting predicted requests based on recent prediction mismatch and selectively re-optimizing assigned but unstarted requests. We also introduce a historical-data-driven procedure for selecting the heterogeneous fleet composition before deployment. In a case study using real nursing-task requests from hospital inpatient floors, the proposed approach achieves near-complete service and reduces serviced-request wait times relative to reactive, token-passing, prediction-positioning, and myopic greedy baselines, with the largest improvements in tail-delay metrics.

1. Introduction

Autonomous robot teams are increasingly being deployed in service environments where spatially distributed requests must be completed under timing, routing, and compatibility constraints. Examples include monitoring and maintenance [75, 114, 15, 95], warehouse and logistics operations [39, 92], last-mile delivery and transportation services [64, 84, 50, 17, 44, 78, 86, 48], UAV resupply and freight management [2, 102], and service coordination during emergency or pandemic responses [32]. In these settings, robot teams rarely execute a fixed set of tasks known completely in advance. Instead, they must repeatedly adapt assignments and schedules as new requests arrive, robot availability changes, and operating conditions deviate from historical expectations.

This paper studies online task assignment and scheduling for heterogeneous multi-robot teams operating over a finite horizon. The team must serve both scheduled requests, known before execution, and real-time requests, revealed sequentially during operation. Requests may require ordered visits to one or more service locations and must satisfy timing, routing, and compatibility constraints. Robots may differ in their service capabilities, travel times, operational status, and maintenance requirements. The resulting planning problem is to assign observed requests to robots while preserving enough scheduling flexibility to respond effectively to uncertain future demand.

A key challenge is that online assignment decisions have delayed consequences. Assigning a robot to a currently observed request changes the robot's future location, availability, and ability to serve requests that may arrive later. Predictive models can help by using historical data to generate possible future request scenarios, allowing the planner to estimate the opportunity cost of current commitments. However, operating conditions may change during deployment: the number, timing, location, or type of real-time requests may differ from the historical patterns used to generate predictions. When this occurs, a planner that continues to rely heavily on inaccurate forecasts may reserve robots unnecessarily, delay urgent observed requests, or overcommit resources to schedules that are poorly matched to the demand actually being realized. This paper addresses the question of how an online heterogeneous multi-robot planner should use predictive demand information when that information is useful on average but may become unreliable during operation.

Existing approaches only partially address this issue. Reactive online assignment methods [96, 52, 113, 26, 8, 7, 31, 100, 45] can respond to newly observed requests, but they are typically myopic and do not explicitly reason about the future consequences of current commitments. Prediction-based methods [99, 34, 80, 90, 109, 25] can anticipate likely future demand, but they often lack mechanisms for adjusting the influence of predictions when recent observations

*Corresponding author
dgarces@g.harvard.edu (D. Garces)
ORCID(s): 0000-0002-4161-0265 (D. Garces)

indicate forecast mismatch. Offline deterministic scheduling methods [60, 115, 98] assume that the relevant requests are known in advance, while exact stochastic dynamic programming formulations [10, 58] are computationally intractable for the heterogeneous online setting considered here if no approximations are used. These limitations motivate an approximate online planning method that can use lookahead to anticipate future demand when predictions are reliable, reduce the influence of predictions when recent observations indicate forecast mismatch, and avoid overcommitting robots when future demand remains uncertain.

Rollout-based lookahead [11, 12] provides a natural way to construct such an approximation. Because current assignments affect future robot availability, sampled future request scenarios can be used to estimate the opportunity cost of assigning a robot to a currently observed request, and candidate decisions can be compared according to their immediate and simulated future costs. However, deploying rollout in this setting introduces significant computational challenges. Each candidate decision is itself a schedule update: the planner must decide not only which robot should serve each known request, but also where that request should be inserted in the robot's route and when each service should occur. As a result, the feasible schedule space grows combinatorially with the number of robots, requests, compatibility relations, and insertion positions, making exhaustive action enumeration impractical. In addition, each candidate schedule must be evaluated by simulating future decisions across sampled request scenarios. These simulations must be efficient enough to run repeatedly online, yet detailed enough to preserve the dominant interactions among robot compatibility, graph-based routing, service timing, and robot availability. Beyond the rollout computation itself, the usefulness of these simulated policies also depends on the initial heterogeneous robot composition: if the team lacks sufficient compatible capacity for bottleneck request types, then even an effective online scheduling policy may be unable to maintain feasible service.

Accurate future-cost estimation is also difficult because the request scenarios used in rollout are only approximations of future operating conditions. Predictions generated from historical data may be informative on average, but the realized number, timing, location, or type of real-time requests can change during deployment. When this occurs, simulated future costs based on outdated or inaccurate forecasts may misrepresent the true opportunity cost of current assignments. Moreover, forecast errors may have already influenced the current schedule before they are detected. For this reason, future-cost estimates should reflect the planner's current confidence in the predicted request scenarios, rather than treating all predictions as equally reliable throughout the operating horizon. The planner must also be able to revise unexecuted assignments when new observations indicate that the current schedule was shaped by unreliable forecasts, while preserving commitments to services that have already begun.

We address these challenges with a prediction-aware adaptive rollout framework for heterogeneous multi-robot task assignment and scheduling. The central idea is to treat predictions as uncertain evidence whose influence should vary over time, rather than as fixed information that should be trusted uniformly throughout the operating horizon. At each decision time, sampled future request scenarios are used to estimate the downstream cost of candidate scheduling decisions. Their contribution to these estimates is modulated by confidence weights derived from recent discrepancies between predicted and observed demand. In this way, the planner can use predictions to make anticipatory decisions when they are informative, while reducing their influence when realized operating conditions suggest that the forecast is unreliable.

The framework combines this adaptive use of predictions with computational mechanisms needed for online deployment. A pruned action-generation procedure constructs a tractable set of candidate schedule modifications instead of enumerating the full feasible schedule space. Explicit wait actions allow the planner to preserve robot capacity when committing immediately to an observed request may be undesirable under uncertain future demand. An interaction-aware base policy is used inside rollout simulations to approximate future assignment and routing decisions efficiently while preserving the main coupling effects among timing, routing, compatibility, and robot availability. A selective re-optimization mechanism can return eligible assigned but unstarted requests to the pending set when new observations indicate that the current schedule has become undesirable, while already-started services remain fixed. As a supporting pre-deployment component, we also develop a historical-data-driven fleet-sizing procedure that selects an initial heterogeneous robot composition for which the assignment routines achieve a target empirical feasibility level on representative operating days.

An overview of the proposed adaptive rollout method is shown in Figure 1. The framework integrates prediction, adaptive forecast reweighting, selective re-optimization, and rollout-based schedule evaluation. Together, these components produce an online planner that is anticipatory when predictions are useful, adaptive when operating conditions change, computationally tractable for repeated online decisions, and less prone to overcommitting robot capacity when future demand is uncertain.

The main contributions of this paper are as follows:

1. We formulate online heterogeneous multi-robot task assignment and scheduling with scheduled and real-time requests as a finite-horizon stochastic dynamic program that captures robot-request compatibility, ordered service requirements, timing constraints, graph-based routing, and end-of-horizon feasibility.
2. We develop a prediction-aware adaptive rollout framework for online scheduling under uncertain demand. The framework uses sampled future request scenarios to estimate the opportunity cost of current decisions, dynamically adjusts their influence based on

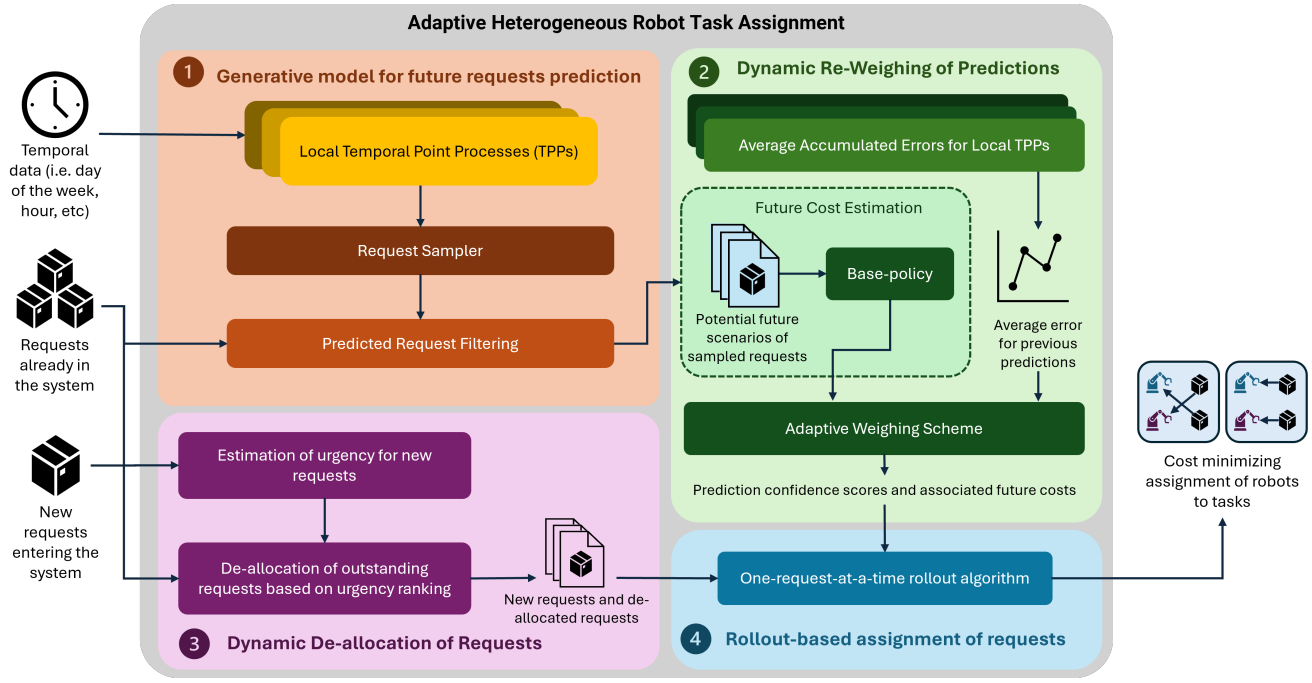

Figure 1: Overview of the proposed prediction-aware adaptive rollout framework for heterogeneous multi-robot task assignment. Local temporal point process models generate sampled future request scenarios, which are filtered to remove requests that are already observed or scheduled. Recent forecast errors are used to dynamically reweight predicted requests, producing context-dependent prediction-confidence scores and weighted future-cost estimates. Newly observed urgent requests can trigger re-optimization by returning eligible assigned but unstarted requests to the pending set. The rollout-based assignment module evaluates candidate assignments for observed and re-optimized requests using the weighted future-cost estimates and selects schedules for the heterogeneous robot team.

recent forecast accuracy, and selectively re-optimizes assigned but unstarted requests when operating conditions deviate from predictions. Pruned action generation, explicit wait actions, and an interaction-aware base policy make the resulting rollout computation practical for repeated online use.

3. We develop a historical-data-driven fleet-sizing procedure for selecting an initial heterogeneous robot composition that achieves a target empirical feasibility level on representative operating days while accounting for robot-request compatibility and bottleneck service requirements.
4. We evaluate the framework using real nursing-task requests from hospital inpatient floors at Beth Israel Deaconess Medical Center. Compared with reactive dispatching, token-passing, prediction-positioning, and myopic greedy baselines, the proposed method achieves near-complete service while reducing both average and upper-tail request waiting times. Ablation studies further demonstrate the complementary roles of adaptive prediction reweighting and selective re-optimization.

The remainder of the paper is organized as follows. Section 2 reviews related work on heterogeneous online task allocation, fleet sizing, request-sequence prediction, rollout-based replanning, and robustness under distribution shift. Section 3 presents the stochastic dynamic programming formulation. Section 4 introduces the prediction-aware adaptive rollout framework and the historical-data-driven fleet-sizing procedure. Section 5 presents the case study and empirical results. Finally, Section 6 concludes the paper and discusses limitations and future directions.

2. Related Work

This work lies at the intersection of heterogeneous multi-robot task allocation, online scheduling under uncertain demand, request-sequence prediction, and robust planning under forecast mismatch. Prior work has developed expressive models for heterogeneous robot coordination, scalable online allocation mechanisms, predictive models for irregular event sequences, and robustness techniques for model-based decision-making. However, these components are often studied separately. Many online allocation methods react to observed requests but do not explicitly reason about the downstream opportunity cost of current commitments. Prediction-aware methods can anticipate future demand, but often assume that forecasts remain reliable during deployment. Robust planning methods account for uncertainty, but typically do not specify how concrete sampled future requests should be weighted inside an online heterogeneous scheduling policy. Our work addresses this gap by integrating future-request prediction into a rollout-based online

planner, adapting the influence of predicted requests using recent forecast errors, and supporting the planner with a historical-data-driven procedure for selecting an initial heterogeneous fleet composition.

2.1. Heterogeneous Online Task Allocation, Scheduling, and Fleet Composition

Multi-robot task allocation has been studied through combinatorial optimization, task-and-motion planning, market-based coordination, decentralized algorithms, learning-based policies, and uncertainty-aware planning. These approaches differ in how they model robot heterogeneity, temporal constraints, routing constraints, and online task arrivals. The setting considered in this paper combines several of these difficulties: robots have type-dependent capabilities and travel times, requests have service windows and ordered endpoint sequences, and new real-time requests arrive during execution. The resulting online scheduling problem requires decisions that are computationally tractable, compatible with heterogeneous robot capabilities, and sensitive to future fleet availability.

A first line of work develops expressive models for heterogeneous allocation, scheduling, and integrated task-and-motion planning. Complexity-theoretic results show that multi-robot task allocation is difficult in general [6], and dynamic variants remain challenging when tasks must be inserted, deleted, or rescheduled under heterogeneous capabilities and temporal constraints [73, 14]. Integrated task-and-motion planning methods further couple assignment with sequencing, routing, energy constraints, coalition formation, or learned subteam performance [24, 74, 73, 76, 16, 69, 4, 37, 9]. These methods provide high modeling fidelity, but repeatedly solving detailed coupled planning problems can be impractical in service settings where requests arrive continuously and schedules must be revised frequently.

A second line of work emphasizes scalability and responsiveness through heuristic, auction-based, market-based, decentralized, or metaheuristic mechanisms [96, 52, 113, 26, 8, 7, 31, 100, 45]. These methods can often respond quickly to newly observed tasks without solving a large centralized optimization problem. However, purely reactive allocation can be myopic when current assignments affect future service quality. For example, assigning a scarce compatible robot to a low-urgency request may prevent that robot from serving a more urgent request that is likely to arrive later. Our framework addresses this limitation by using sampled future request scenarios to estimate the downstream opportunity cost of current commitments, while still restricting executable assignments to requests that have actually entered the system.

Learning-based allocation methods provide another route to fast online decision-making. Recent work has explored graph neural network schedulers [99], imitation-learning-based collaborative schedulers [34], reinforcement learning for cooperative task allocation [80, 90, 109, 25], and LLM-based planning or coordination frameworks [23, 56, 46, 38, 107]. These approaches can amortize computation through offline training and produce rapid decisions during deployment. However, learned allocation policies can degrade when request arrival patterns, task frequencies, robot availability, travel conditions, or operating policies shift away from the training distribution. In contrast, our approach does not require retraining the allocation policy when prediction quality changes. Instead, it adjusts the planning objective online by reducing the influence of predicted future requests when recent observations indicate forecast mismatch.

Several works incorporate uncertainty directly into heterogeneous allocation, including robust schedules under uncertain robot capabilities [33], stochastic trait models with probabilistic task satisfaction [81], hindsight optimization for uncertain task outcomes [27], and proactive allocation under spatiotemporal uncertainty [93]. These methods show the importance of modeling uncertainty in robot capabilities, task outcomes, and future demand. Our focus is complementary: we consider uncertainty in the future request stream and, more specifically, in the reliability of the predictive distribution used by the online planner. While prior uncertainty-aware allocation methods often model uncertainty in the planning problem itself, they do not typically adapt the influence of predicted future requests based on forecast errors observed during deployment. Our method addresses this gap by estimating prediction reliability online and using it to reweight predicted demand during rollout-based replanning.

The effectiveness of an online assignment policy also depends on the initial heterogeneous robot composition. In heterogeneous service systems, adding robots does not necessarily increase capacity for all requests: a fleet may still fail if it lacks robots compatible with bottleneck request types. Many allocation and scheduling approaches assume that the robot team is fixed before planning begins, while related work on team formation, coalition selection, and subteam performance typically decides which available robots should cooperate on tasks during execution [74, 73, 37, 9]. These problems are distinct from selecting the initial fleet composition before deployment, where the goal is to ensure that the online assignment policy has sufficient compatible capacity for the demand patterns it is likely to encounter. Our work treats initial heterogeneous fleet sizing as a supporting design decision for online scheduling. We develop a historical-data-driven procedure that evaluates candidate fleet compositions on representative operating days and selects a composition for which the assignment routines used in simulation achieve a target empirical feasibility level.

2.2. Request-Sequence Prediction and Scenario Generation

Future service demand in online task-allocation problems can be represented as an irregular sequence of marked events. Each event has an arrival time and a mark encoding task-relevant information such as service type, location, priority, workload, or task family. Temporal point processes are well suited to this setting because they model event timing and marks directly, without requiring demand to be aggregated into fixed time bins. This is useful in sparse

or bursty service environments, where discretization can obscure timing structure or introduce many artificial zero-count observations [5, 19]. For heterogeneous robot allocation, the mark structure is especially important because downstream feasibility depends not only on when a request arrives, but also on which robot types can serve it and which service locations must be visited.

Recent work has extended temporal point process models beyond classical formulations by combining continuous-time event modeling with neural sequence architectures. Neural marked temporal point processes have been developed for multivariate and set-valued event data [20], invertible intensity models improve likelihood evaluation and sampling for multivariate processes [5], and state-space point processes combine deep state-space models with continuous-time marked event dynamics [19]. Transformer-based event models provide another direction by using attention mechanisms to model dependencies across event histories while preserving continuous and discrete event attributes [28]. These developments are relevant to robot service systems because they enable predictive models to condition on irregular request histories and generate heterogeneous future request samples.

In our framework, temporal point process predictions are not used as fixed schedules or mandatory tasks. Instead, the prediction module generates sampled future request scenarios from an estimated distribution. Each sampled event is converted into the same request representation used by the planner, including task type, service-window parameters, and graph-grounded endpoint sequence. This representation allows predicted future requests to be evaluated using the same compatibility, routing, and timing constraints. However, because deployment demand may differ from the historical data used to train the predictor, the planner must decide how much influence these sampled requests should have on current decisions. Our contribution is therefore not a new temporal point process architecture, but a planning framework that uses such predictions in a confidence-aware manner. As real requests are observed, recent forecast errors are used to update context-dependent prediction weights, and those weights determine how strongly predicted requests affect rollout values.

2.3. Robust Planning Under Forecast Error and Distribution Shift

The value of rollout-based lookahead [11, 12] depends on the quality of the future request scenarios used in simulation. Predictive models are typically trained on historical data, but deployment conditions may change because of shifts in request arrival rates, task mix, robot availability, travel times, patient or customer behavior, or operating policies. In online allocation, this mismatch is especially consequential because forecast-driven decisions affect future robot availability and may continue to shape the schedule even after prediction errors become apparent. If forecast errors persist, repeated replanning can continue to reintroduce misleading predicted demand unless the planner explicitly reassesses forecast reliability.

Prior work addresses related issues through multi-step prediction, self-correction, adaptation, and retraining. Multi-step prediction and self-correcting dynamics models reduce error accumulation by training over longer rollouts or exposing models to their own past prediction errors [3, 94]. Other methods adapt learned models after detecting mismatch between real and simulated data through feature alignment, instance reweighting, event-triggered updates, or concept-drift pipelines [89, 42, 112]. These approaches can improve the predictive model itself, but they may require sufficient post-shift data and additional computation before the adapted model becomes useful. Our method instead acts directly at the planning layer: when recent forecast errors increase, the rollout objective immediately reduces the influence of predicted future requests from unreliable contexts.

Other work reduces reliance on model predictions when uncertainty or error is high. A common strategy is to shorten the effective model rollout horizon as model accuracy deteriorates [43], as in short-horizon model-based reinforcement learning [40], or to use uncertainty-aware masking schemes that only use model rollouts when they are sufficiently reliable [79]. Related approaches reweight simulated data, Bellman targets, or rewards according to model quality or uncertainty [105, 108], with similar ideas appearing in adaptive learning and inference-time reweighting [57, 51]. Our framework shares the principle that unreliable predictions should have less influence, but differs in how this influence is applied. Rather than discarding predictions beyond a fixed horizon or retraining the predictor, we compute context-dependent confidence weights from observed forecast errors and use them to continuously modulate predicted-request costs during rollout.

Distributionally robust optimization and conformal prediction provide more conservative forms of robustness. Distributionally robust optimization optimizes against a family of plausible distributions and has been applied to stochastic control, routing, vehicle balancing, and other risk-sensitive planning problems [70, 21, 106, 68, 111, 41, 1, 85]. Conformal prediction provides calibrated prediction sets or intervals and has been used in robotics, planning, control, online prediction, and safety assurance [97, 54, 55, 36, 35, 22, 13, 53, 59]. These methods provide principled uncertainty quantification, but they do not directly specify how an online multi-robot allocation policy should weight concrete sampled future requests during replanning. Our approach is operational: it converts recent forecast errors into prediction-confidence weights that directly affect the rollout value of candidate scheduling decisions.

Re-optimization provides a complementary mechanism for robustness. Model predictive control and online allocation methods repeatedly replan as new state information becomes available, which helps limit the effect of disturbances and modeling errors [65]. However, replanning alone does not necessarily repair assignments that were made

when predictions appeared reliable, especially if those assignments have not yet started but continue to occupy robot capacity. Our framework therefore combines adaptive prediction weighting with selective re-optimization of unstarted assignments. When new observations indicate that the current schedule has become undesirable, eligible assigned but unstarted requests are returned to the pending set and reconsidered under the updated state and prediction-confidence weights. This allows the planner to recover from forecast-driven allocation errors while preserving commitments to services that have already begun.

3. Problem Formulation

We consider a heterogeneous multi-robot task-assignment and scheduling problem over a finite operating horizon. A fixed robot team must serve scheduled requests known before execution and real-time requests revealed sequentially during operation. Each request has a task type, an ordered sequence of service locations, and timing requirements. Robots are heterogeneous in their task compatibility, travel times, operational status, and assigned maintenance stations / home stations. The online planning problem is to update assignments and schedules as new requests arrive so as to minimize service delay while satisfying compatibility, routing, timing, and end-of-horizon return constraints.

Future real-time requests are uncertain, and the deployment request distribution may differ from the historical distribution used to train the prediction model. We therefore formulate the online scheduling problem as a finite-horizon stochastic dynamic program. The robot team composition is treated as fixed in this formulation; the historical-data-driven procedure used to select this composition before deployment is introduced in Section 4.

3.1. Operating Horizon and Environment

We consider one operating day at a time and index the finite horizon by time steps $t = 0, 1, \dots, T$, where T is the horizon length. A control selected at decision time $t \in \{0, \dots, T-1\}$ determines the evolution of the system over the interval $[t, t+1]$.

The physical environment is represented by a directed traversal graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the finite set of traversable nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of directed traversal edges. Each node $v \in \mathcal{V}$ corresponds to a collision-free robot configuration, waypoint, or location in the environment. Each edge $(v, v') \in \mathcal{E}$ corresponds to a physically realizable local trajectory that moves a robot from node v to node v' without collision. Obstacles and inaccessible regions are therefore represented implicitly by the absence of corresponding nodes or edges in $\mathcal{G}$.

Only a subset of traversal nodes can appear as request service locations. We denote this subset by $\mathcal{V}^{\text{svc}} \subseteq \mathcal{V}$. Nodes in $\mathcal{V}^{\text{svc}}$ correspond to pickup, drop-off, or task-service locations. Nodes in $\mathcal{V} \setminus \mathcal{V}^{\text{svc}}$ are traversal nodes used only for motion planning and do not accept requests.

Let $\mathcal{Z}$ denote the finite set of robot types. Robot types represent different robot capabilities with distinct available actions. For a robot of type $z \in \mathcal{Z}$, the time it takes to traverse an edge (v, v') is denoted as $\tau_z(v, v') > 0$.

3.2. Robot Team and Task Compatibility

The heterogeneous team composition is $\mathbf{M} = (M_z)_{z \in \mathcal{Z}}$, where M_z is the number of robots of type z. The total number of robots is $M = \sum_{z \in \mathcal{Z}} M_z$.

The robot identifiers are indexed by $\mathcal{L} = \{1, \dots, M\}$. Each robot $\ell \in \mathcal{L}$ has type $z_\ell \in \mathcal{Z}$.

The finite set of task types is denoted by $\mathcal{K}$. Robot-task compatibility is encoded by $\mathcal{K}(z) \subseteq \mathcal{K}$, where $k \in \mathcal{K}(z)$ means that a robot of type z can serve a request of task type k.

The environment contains a set of maintenance stations $\mathcal{V}^{\text{st}} \subseteq \mathcal{V}$. Each robot $\ell \in \mathcal{L}$ is assigned a maintenance station $v^{\text{st}}_\ell \in \mathcal{V}^{\text{st}}$. Robots start the horizon at their assigned maintenance stations and must return to their assigned maintenance stations no later than time T.

The online planning problem is defined for a fixed team composition $\mathbf{M}$.

3.3. Requests

A request r_j is described by

$$r_j = \left(k_j, \boldsymbol{\rho}_j, t_j^{\text{entry}}, t_j^{\text{start}}, t_j^{\text{des}}, t_j^{\text{comp}}\right). \tag{1}$$

Here, $k_j \in \mathcal{K}$ is the task type. Because the completion of a task may involve a robot visiting a sequence of locations in a specific order, we model the ordered sequence of service nodes that must be visited to complete request r_j as

$$\boldsymbol{\rho}_j = \left(\rho_{j,1}, \rho_{j,2}, \dots, \rho_{j,n_j^{\text{svc}}}\right), \qquad \rho_{j,h} \in \mathcal{V}^{\text{svc}}, \tag{2}$$

The restriction $\rho_{j,h} \in \mathcal{V}^{\text{svc}}$ means that requests can only originate, terminate, or require service at designated service nodes; other nodes in $\mathcal{V}$ are used only for traversal.

The time t_j^{entry} is the time at which request r_j becomes known to the system, t_j^{start} is the earliest allowable service start time, t_j^{des} is the desired completion time, and t_j^{comp} is the latest allowable completion time. We assume

$$t_j^{\text{entry}} \le t_j^{\text{start}} \le t_j^{\text{des}} \le t_j^{\text{comp}} \le T.$$

Each task type $k \in \mathcal{K}$ has an execution time F_k. Thus, a feasible schedule for request r_j must visit the service nodes in the order specified by $\boldsymbol{\rho}_j$, respect the earliest-start time t_j^{start}, allocate execution time F_{k_j}, and complete the request no later than t_j^{comp}.

Requests are divided into scheduled and real-time requests. The scheduled request set is

$$\mathcal{R}^{\text{sched}} = \left\{ r_j : t_j^{\text{entry}} < 0,\ 0 \le t_j^{\text{start}} \le t_j^{\text{des}} \le t_j^{\text{comp}} \le T \right\}. \tag{3}$$

These requests are known before the operating horizon begins. The real-time request set revealed at time t is

$$\mathcal{R}_t^{\text{real}} = \left\{ r_j : t_j^{\text{entry}} = t,\ t \le t_j^{\text{start}} \le t_j^{\text{des}} \le t_j^{\text{comp}} \le T \right\}.$$

(4)

The set of requests known after the arrivals at time t have been observed is

$$\mathcal{R}_t = \mathcal{R}^{\text{sched}} \cup \bigcup_{\tau=0}^{t} \mathcal{R}_\tau^{\text{real}}. \tag{5}$$

3.4. Standing Assumptions

We impose the following assumptions throughout the paper.

Assumption 1 (Well-formed environment and request process)**.** *The traversal graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and the request process are well formed in the following sense:*

1. *The scheduled request set $\mathcal{R}^{\text{sched}}$ is finite, and the real-time request set $\mathcal{R}_t^{\text{real}}$ entering the system at each decision time $t = 0, \dots, T-1$ is finite.*
2. *For every ordered pair of service nodes $v, v' \in \mathcal{V}^{\text{svc}}$ that may appear as request endpoints, there exists a directed path $P(v, v') = (v_0, v_1, \dots, v_K)$ on $\mathcal{G}$ such that $v_0 = v$, $v_K = v'$, $(v_i, v_{i+1}) \in \mathcal{E}$ for all $i = 0, \dots, N-1$, and $v_i \notin \mathcal{V}^{\text{svc}}$ for $i = 1, \dots, N-1$. Thus, any two request endpoints can be connected by a traversal path whose internal nodes are not themselves service locations.*

Assumption 2 (Reliable robot execution)**.** *Robots execute assigned motion and service actions without failures. In particular, if a robot is assigned a feasible schedule, then it follows the corresponding paths on $\mathcal{G}$, performs the required service actions, and completes them at the planned times. Thus, uncertainty in the model enters through the real-time request process rather than through robot failures, localization failures, or stochastic task execution.*

Assumption 1 ensures that the request process is finite over the planning horizon and that service locations are not unavoidable transit bottlenecks in the traversal graph. This allows a motion planner to connect request endpoints without requiring robots to pass through other service nodes as intermediate waypoints. Assumption 2 focuses the stochastic component of the model on future request arrivals, rather than on failures in robot motion, localization, or task execution.

3.5. Stochastic Request Process and Distribution Shift

Let $\omega_t = \mathcal{R}_t^{\text{real}}$ denote the random real-time requests revealed at decision time t. For notational convenience, let $\omega_T = \emptyset$, so that no new requests enter after the final decision time. The request sequence for an operating day is $\mathcal{W} = (\omega_0, \omega_1, \dots, \omega_T)$, where the last element is deterministic and empty. The true deployment distribution of $\mathcal{W}$ is denoted by $\mathbb{P}$. This distribution governs the number of real-time requests, their task types, entry times, service windows, desired completion times, and spatial endpoints.

Historical operating days provide request-sequence samples that are used to train a predictive model. The model induces an estimated distribution $\widehat{\mathbb{P}}$, from which sampled future request scenarios are generated during rollout. We do not assume that $\widehat{\mathbb{P}} = \mathbb{P}$. The mismatch between $\widehat{\mathbb{P}}$ and $\mathbb{P}$ represents the distribution shift considered in this paper.

3.6. Schedules and Feasible Controls

At time t, each robot has a current node, operational status, assignment, and planned schedule. Let y_t^ℓ denote the planned schedule for robot ℓ. A schedule specifies the future traversal nodes and service actions of robot ℓ from time t until its return to its maintenance station.

For a given state x_t, let $\mathcal{Y}_t^\ell(x_t)$ denote the set of feasible schedules for robot ℓ. A schedule $y_t^\ell \in \mathcal{Y}_t^\ell(x_t)$ is feasible if it satisfies the following conditions:

1. it starts from the current node of robot ℓ at time t;
2. it follows valid directed paths on the traversal graph $\mathcal{G}$;
3. it assigns only compatible requests to robot ℓ, meaning $k_j \in \mathcal{K}(z_\ell)$;
4. it visits the service nodes of each assigned request r_j in the order specified by $\boldsymbol{\rho}_j$;
5. it respects each assigned request's earliest-start time and deadline;
6. it preserves the already executed portion of the current schedule;
7. it does not reassign a request to a different robot after service has started; and
8. it returns robot ℓ to its maintenance station v_ℓ^{st} no later than time T.

The joint schedule is $\mathbf{y}_t = (y_t^1, y_t^2, \dots, y_t^M)$. The corresponding feasible joint schedule set is $\mathcal{Y}_t(x_t) = \mathcal{Y}_t^1(x_t) \times \cdots \times \mathcal{Y}_t^M(x_t)$.

A control at time t is an updated assignment and schedule, $u_t = (\mathbf{a}_t^u, \mathbf{y}_t^u)$, where $a_{j,t}^u \in \mathcal{L} \cup \{-1, \emptyset\}$ records whether each known request $r_j \in \mathcal{R}_t$ is assigned to a robot, rejected, or left pending, and $\mathbf{y}_t^u$ is the resulting team schedule. The feasible control set is denoted by $\mathcal{U}_t(x_t)$. Thus, $u_t \in \mathcal{U}_t(x_t)$ if the updated assignments and schedules satisfy the compatibility, routing, time-window, no-reassignment, and return-to-station constraints described above.

3.7. State Dynamics

The state at time t, after observing the real-time requests revealed at time t, is $x_t = (\mathbf{p}_t, \boldsymbol{\theta}_t, \mathbf{y}_t, \mathbf{a}_t, \boldsymbol{\sigma}_t, \mathcal{R}_t)$. Here, $\mathbf{p}_t$ is the vector of nodes currently occupied by the robots, $\boldsymbol{\theta}_t$ is the vector of robot operational statuses, $\mathbf{y}_t$ is the current team schedule, $\mathbf{a}_t$ is the current request assignment vector, $\boldsymbol{\sigma}_t$ is the vector of request service statuses, and $\mathcal{R}_t$ is the set of requests known by time t.

For each robot ℓ the status θ_t^ℓ records whether robot ℓ is available, executing an assigned schedule, or returning to its maintenance station. For each request $r_j \in \mathcal{R}_t$, the assignment status satisfies $a_{j,t} \in \mathcal{L} \cup \{-1, \emptyset\}$, where $a_{j,t} = \ell$ means that request r_j is assigned to robot ℓ, $a_{j,t} = \emptyset$ means that it is known but not currently assigned, and $a_{j,t} = -1$ means that it has been rejected. The service status satisfies

$$\sigma_{j,t} \in \{\text{pending}, \text{started}, \text{completed}, \text{rejected}\}. \tag{6}$$

After selecting a feasible control u_t, robots execute the first step of their updated schedules. The system then observes the next real-time request set ω_{t+1}, and the state evolves according to $x_{t+1} = f_t\left(x_t, u_t, \omega_{t+1}\right)$. For notational convenience, let $\omega_T = \emptyset$. For $t = 0, \ldots, T-2$, the transition $x_{t+1} = f_t(x_t, u_t, \omega_{t+1})$ incorporates the real-time requests revealed at the next decision time. At the terminal transition, $x_T = f_{T-1}(x_{T-1}, u_{T-1}, \omega_T)$ with $\omega_T = \emptyset$, so no new requests enter after the final decision time.

3.8. Service Cost and Planning Objective

The planner evaluates assignments according to the service delay and feasibility of the resulting schedules. For a known request $r_j \in \mathcal{R}_t$ that is assigned to a feasible schedule, let $\widehat{C}_{j,t}(x_t)$ denote its planned completion time under the current assignment and schedule. The planned delay of request r_j is $h_j(x_t) = \max\left\{0, \widehat{C}_{j,t}(x_t) - t_j^{\text{des}}\right\}$.

A known request is said to be still serviceable at state x_t if there exists at least one compatible robot schedule, consistent with the constraints in $\mathcal{Y}_t(x_t)$, that can complete the request no later than t_j^{comp}. Pending serviceable requests are not assigned the infeasibility penalty, since a wait action may intentionally leave such requests unassigned to preserve future capacity.

To compare candidate schedules during online planning, we use a finite penalized request cost that distinguishes between assigned requests, pending requests that can still be served, and requests that have been rejected or have become infeasible. Let $\Phi < \infty$ be a large penalty for rejected or infeasible requests. Define

$$H_j(x_t) = \begin{cases} h_j(x_t), & \text{if } r_j \text{ is assigned to a feasible schedule or completed,} \\ 0, & \text{if } r_j \text{ is pending and can still be feasibly served,} \\ \Phi, & \text{if } r_j \text{ is rejected or can no longer be feasibly served.} \end{cases} \tag{7}$$

Pending serviceable requests incur no immediate cost, but they remain in the known request set and are evaluated in subsequent decision times. Any request that is not completed or assigned to a feasible schedule by the terminal time is treated as no longer serviceable and receives the penalty Φ.

The aggregate schedule score is

$$H(x_t) = \sum_{r_j \in \mathcal{R}_t} H_j(x_t), \tag{8}$$

with the convention that $H(x_t) = 0$ if no request has entered the system. The penalty Φ is chosen large enough to discourage rejection or loss of serviceability, while pending serviceable requests incur no immediate penalty. Thus, a wait action may intentionally leave a request unassigned when preserving capacity is valuable, but the request must eventually be assigned to a feasible schedule or incur the penalty if it is rejected or becomes no longer serviceable.

The one-step cost is defined as the change in aggregate schedule score: $g_t(x_t, u_t, \omega_{t+1}) = H(x_{t+1}) - H(x_t)$, where $x_{t+1} = f_t(x_t, u_t, \omega_{t+1})$. This incremental cost measures how the selected assignment and schedule update changes planned service quality after the next request arrivals are incorporated.

A policy is a sequence $\pi = \left\{\mu_0, \mu_1, \ldots, \mu_{T-1}\right\}$, where each decision rule maps the current state to a feasible control: $\mu_t(x_t) \in \mathcal{U}_t(x_t)$.

The finite-horizon cost-to-go of policy π from state x_t is

$$J_t^{\pi}(x_t) = \mathbb{E}_{\mathbb{P}}\left[\sum_{\tau=t}^{T-1} g_\tau\left(x_\tau, \mu_\tau(x_\tau), \omega_{\tau+1}\right) \,\middle|\, x_t\right], \tag{9}$$

where the expectation is taken over future real-time request arrivals governed by the deployment distribution $\mathbb{P}$. Because g_t is defined as the change in aggregate schedule score, the finite-horizon objective evaluates the expected net change in planned service quality over the horizon. Equivalently, up to the fixed baseline $H(x_t)$, the objective minimizes the expected terminal aggregate schedule score.

The ideal planning objective is to find an admissible policy that minimizes this expected cost-to-go:

$$J_t^{\pi^*}(x_t) = \min_{\pi \in \Pi} J_t^{\pi}(x_t), \tag{10}$$

where Π is the set of admissible policies.

The stochastic dynamic program in (10) provides a formal description of the online planning objective, but it cannot be solved exactly in the setting considered here. The control space grows combinatorially because each decision requires selecting assignments, insertion positions, service times, and routes for heterogeneous robots. In addition, evaluating a candidate control requires reasoning about future request arrivals whose distribution is only estimated from historical data and may change during deployment. These difficulties motivate the prediction-aware adaptive rollout framework developed in Section 4, which approximates the feasible control set, estimates downstream cost through efficient base-policy simulation, adapts the influence of predicted requests using recent forecast errors, and re-optimizes eligible unstarted assignments when the current schedule becomes undesirable.

4. Our Approach

We now develop the prediction-aware adaptive rollout framework motivated by the stochastic dynamic program in Section 3. The framework defines an online policy $\tilde{\pi} = \tilde{\mu}_0, \tilde{\mu}_1, \ldots, \tilde{\mu}_{T-1}$, where each decision rule selects a feasible control from a tractable candidate set, $\tilde{\mu}_t(x_t) \in \widetilde{\mathcal{U}}_t(x_t)$, where $\widetilde{\mathcal{U}}_t(x_t) \subseteq \mathcal{U}_t(x_t)$. The candidate set contains schedule modifications for requests that have actually entered the system. Predicted future requests are not eligible for immediate assignment; they are used only inside rollout simulations to estimate the downstream cost of current decisions.

4.1. Overview of the Prediction-Aware Adaptive Rollout Framework

At each decision time, the online policy combines computationally tractable candidate generation, sampled future request scenarios, interaction aware base-policy simulation, adaptive prediction confidence weights, and selective re-optimization. After newly observed requests have been incorporated into the state, the planner may return eligible assigned but unstarted requests to the pending set when the current schedule has become undesirable. It then generates a small set of feasible schedule modifications for currently pending requests. Each candidate is evaluated by simulating future decisions over sampled request scenarios from the estimated distribution $\widehat{\mathbb{P}}$, with predicted requests weighted according to recent forecast accuracy. The selected control updates the real robot schedules, while predicted requests remain hypothetical and influence the decision only through their weighted contribution to simulated future cost.

The rollout policy is implemented using a one-robot-at-a-time decision rule. Instead of optimizing over the full joint control set $\mathcal{U}_t(x_t)$, the planner processes robots sequentially. When a robot is considered, controls already selected for earlier robots are held fixed, a robot-local candidate set is generated, and each local candidate is evaluated by completing the remaining simulated decisions with an efficient base policy. This decomposition replaces one large joint scheduling problem with a sequence of smaller robot-level rollout decisions, making online evaluation practical while still accounting for interactions through the simulated state, service-node reservations, and future-cost base policy.

The online rollout policy is supported by a pre-deployment fleet-sizing procedure. This procedure uses historical request sequences to select a heterogeneous robot composition for which the assignment routines used inside rollout achieve a target empirical feasibility level on representative operating days. The resulting composition is then treated as fixed during online deployment, as assumed in Section 3.

The remainder of this section presents the components of the framework. Section 4.2 gives the one-robot-at-a-time rollout decision rule. Section 4.3 describes candidate-control generation and wait actions. Section 4.4 describes future-request scenario generation. Section 4.5 introduces the interaction-aware base policy used for future-cost estimation. Section 4.6 presents adaptive prediction-confidence weighting. Section 4.7 describes selective re-optimization of unstarted assignments. Finally, Section 4.8 presents the supporting historical fleet-sizing procedure.

4.2. One-Robot-at-a-Time Rollout Decision Rule

We first define the online rollout decision rule. The remaining subsections describe the components used by this rule: candidate-control generation, future-request scenario generation, the interaction-aware base policy, adaptive prediction-confidence weights, selective re-optimization, and the supporting fleet-sizing procedure.

At decision time t, the planner has observed state x_t and must select a feasible control from a tractable subset of the full feasible control set. Directly optimizing over the joint candidate set for all robots is expensive because a joint control specifies assignments, insertion positions, service times, and routes for the entire team. We therefore use a one-robot-at-a-time rollout scheme, following the one-agent-at-a-time rollout principle in [11, 12]. This replaces one large joint optimization with a sequence of smaller robot-local optimizations, while using simulation to account for the effect of each local decision on the rest of the team.

Let $\mathcal{O}_t = (\ell_1, \ell_2, \dots, \ell_M)$ denote the order in which robots are processed at decision time t. This order may be fixed or chosen from the current state, for example by prioritizing robots that become available earlier. When robot ℓ_i is considered, the local controls already selected for robots $\ell_1, \dots, \ell_{i-1}$ are held fixed. The planner then generates a robot-local candidate set $\widetilde{\mathcal{U}}_t^{\ell_i}\left(x_t; \widetilde{u}_t^{\ell_1}, \dots, \widetilde{u}_t^{\ell_{i-1}}\right)$, where each candidate modifies only the schedule of robot ℓ_i and only uses requests that have entered the system by time t. This set includes feasible assignment actions and a wait action, as described in Section 4.3. Predicted future requests are not included in the immediate candidate set.

For a candidate local control

$$u_t^{\ell_i} \in \widetilde{\mathcal{U}}_t^{\ell_i}\left(x_t; \widetilde{u}_t^{\ell_1}, \dots, \widetilde{u}_t^{\ell_{i-1}}\right),$$

we construct a completed joint control by fixing the controls already selected for earlier robots, applying $u_t^{\ell_i}$ to robot ℓ_i, and using the base policy to complete the simulated decisions for robots that have not yet been processed:

$$u_t^{\mathrm{comp},i}\left(u_t^{\ell_i}\right) = \mathrm{Complete}_{\bar{\pi}}\left(x_t; \widetilde{u}_t^{\ell_1}, \dots, \widetilde{u}_t^{\ell_{i-1}}, u_t^{\ell_i}\right).$$

Here, $\bar{\pi}$ is the interaction-aware base policy introduced in Section 4.5. The completion step is used only to evaluate candidate local controls inside rollout; the executable control applied to the real system is obtained after all robots have been processed.

The candidate $u_t^{\ell_i}$ is evaluated using sampled future request scenarios. Let $\widehat{\mathcal{W}}_t^s = \left(\widehat{\omega}_{t+1}^s, \widehat{\omega}_{t+2}^s, \dots, \widehat{\omega}_{\bar{T}}^s\right)$ be the s-th sampled scenario generated from the estimated distribution $\widehat{\mathbb{P}}$, where $\bar{T} = \min\{t + D + 1, T\}$ and D denotes the rollout depth. Starting from x_t, the simulator applies the completed joint control $u_t^{\mathrm{comp},i}(u_t^{\ell_i})$, incorporates the sampled future requests in $\widehat{\mathcal{W}}_t^s$, and then applies the base policy $\bar{\pi}$ until the truncated horizon $\bar{T}$. Let $x_{\bar{T}}^{s,i,u}$ denote the resulting simulated state.

Because pending serviceable requests have zero immediate cost in the formulation of H_j, the rollout value is not computed by scoring an arbitrary partially unresolved state. Instead, before evaluating the terminal score, the base policy is used to resolve pending requests that remain relevant within the truncated rollout horizon. Let $\mathrm{Resolve}_{\bar{\pi}}(x)$ denote the simulated state obtained by applying the base policy to the pending requests in state x, assigning them if feasible and marking them rejected or no longer serviceable if they

cannot be completed before their deadlines. We define the resolved weighted terminal score

$$H^{\lambda}_{\mathrm{res}}(x) = H^{\lambda}\left(\mathrm{Resolve}_{\bar{\pi}}(x)\right), \tag{11}$$

where H^{λ} is the weighted aggregate schedule score defined in Section 4.6. Observed and scheduled requests receive unit weight, while predicted requests are weighted according to the current prediction-confidence values. This resolved score ensures that a request left pending by a wait action is not treated as cost-free indefinitely: during simulation it is either assigned and contributes its planned delay, remains available for assignment at a later simulated decision time, or is eventually marked rejected or no longer serviceable and receives the penalty Φ.

The prediction-aware rollout value of candidate $u_t^{\ell_i}$ is estimated by

$$\widehat{Q}^{\lambda,i}_t\left(x_t, u_t^{\ell_i}\right) = \frac{1}{S}\sum_{s=1}^{S} H^{\lambda}_{\mathrm{res}}\left(x^{s,i,u}_{\bar{T}}\right). \tag{12}$$

Because the one-step cost in Section 3.8 is defined as a change in aggregate schedule score, this terminal-score form is equivalent, up to a fixed baseline, to summing weighted incremental costs over the truncated rollout horizon.

The selected local control for robot ℓ_i is

$$\widetilde{u}_t^{\ell_i} \in \underset{v_t^{\ell_i} \in \widetilde{\mathcal{U}}_t^{\ell_i}\left(x_t; \widetilde{u}_t^{\ell_1}, \ldots, \widetilde{u}_t^{\ell_{i-1}}\right)}{\operatorname{argmin}} \widehat{Q}^{\lambda,i}_t\left(x_t, v_t^{\ell_i}\right). \tag{13}$$

After the selected control is fixed, the planner proceeds to the next robot in the order $\mathcal{O}_t$. Once all robots have been processed, the one-robot-at-a-time rollout control is $\widetilde{u}_t = \left(\widetilde{u}_t^{\ell_1}, \widetilde{u}_t^{\ell_2}, \ldots, \widetilde{u}_t^{\ell_M}\right)$. This decision rule is prediction-aware because candidate controls are evaluated using sampled future request scenarios. It is adaptive because the contribution of predicted requests to H^{λ} depends on online confidence weights computed from recent forecast errors. It avoids premature commitment because predicted requests are never included in the immediate candidate sets; they influence the decision only through their effect on simulated downstream schedule quality.

4.3. Candidate-Control Generation and Wait Actions

The one-robot-at-a-time rollout rule in Section 4.2 requires a finite robot-local candidate set. This subsection describes how that set is constructed. The goal is not to enumerate all feasible controls in $\mathcal{U}_t(x_t)$, but to generate a small set of feasible and promising schedule modifications for currently known requests. Predicted future requests are deliberately excluded from this immediate candidate set; they influence candidate selection only later, through rollout simulation.

At decision time t, let $\mathcal{D}_t$ denote the set of known requests that are eligible for assignment by the candidate generator. After any selective re-optimization step has been applied, we let

$$\mathcal{D}_t = \left\{r_j \in \mathcal{R}_t : \sigma_{j,t} = \text{pending},\ a_{j,t} = \emptyset\right\}. \tag{14}$$

Thus, $\mathcal{D}_t$ contains requests that are known, not rejected, not completed, and not currently fixed in an executing service. Requests whose service has already started are excluded because feasible controls must preserve started assignments. Assigned but unstarted requests enter $\mathcal{D}_t$ only if the re-optimization mechanism in Section 4.7 explicitly releases them back to the pending set.

For each request $r_j \in \mathcal{D}_t$, the compatible robot set is

$$\mathcal{L}_j = \left\{\ell \in \mathcal{L} : k_j \in \mathcal{K}(z_\ell)\right\}. \tag{15}$$

When robot ℓ is processed by the one-robot-at-a-time rollout rule, only requests for which ℓ is a compatible robot (i.e. $\ell \in \mathcal{L}_j$) are considered as assignment candidates for that robot.

A robot-local assignment action assigns a request $r_j \in \mathcal{D}_t$ to a compatible robot $\ell \in \mathcal{L}_j$ and appends the service-node sequence $\boldsymbol{\rho}_j$ at the end of the remaining schedule of robot ℓ. We denote such an action by $a = (r_j, \ell)$. The action is retained only if the resulting schedule is feasible with respect to the constraints in Section 3.6: it must follow valid directed paths on $\mathcal{G}$, visit the service nodes in the order specified by $\boldsymbol{\rho}_j$, satisfy release-time and deadline constraints, preserve the already executed portion of the schedule, and return the robot to its maintenance station by time T.

For each action $a = (r_j, \ell)$, we compute two heuristic values used only to rank candidate actions before rollout evaluation. First, we compute a latest feasible service-start time. Let $F^{\mathrm{travel}}_{\ell,j}$ denote the estimated time required for robot ℓ to traverse the service-node sequence $\boldsymbol{\rho}_j$, ignoring congestion and conflicts with other robots, but using the type-dependent graph travel-time estimates. The latest feasible service-start time is

$$F^{\mathrm{start}}_{\ell,j} = t^{\mathrm{comp}}_j - F^{\mathrm{travel}}_{\ell,j} - F_{k_j}. \tag{16}$$

This value estimates how urgent the request is: smaller values indicate less remaining scheduling flexibility.

Second, we compute a robot-specific heuristic completion delay. Let $F^{\mathrm{reach}}_{\ell,j}$ denote the estimated time at which robot ℓ reaches the first service node of r_j if the request is appended to its remaining schedule. The corresponding heuristic completion time is

$$\widehat{C}_{\ell,j,t} = \max\left\{F^{\mathrm{reach}}_{\ell,j}, t^{\mathrm{start}}_j\right\} + F^{\mathrm{travel}}_{\ell j} + F_{k_j}. \tag{17}$$

If $\widehat{C}_{\ell,j,t} > t^{\mathrm{comp}}_j$, the action is discarded. Otherwise, the heuristic delay is

$$F^{\mathrm{delay}}_{\ell,j,t} = \max\left\{0, \widehat{C}_{\ell,j,t} - t^{\mathrm{des}}_j\right\}. \tag{18}$$

Candidate assignment actions are ranked lexicographically by urgency and estimated delay:

$$\mathrm{rank}_t(\ell, r_j) = \left(F^{\mathrm{start}}_{\ell,j}, F^{\mathrm{delay}}_{\ell,j,t}\right). \tag{19}$$

Actions with earlier latest feasible service-start times are expanded first, and ties are broken by the robot-specific heuristic delay. After sorting by (19), the generator keeps only the first L feasible assignment actions for robot ℓ, where L is the candidate limit.

The ranking in (19) is used only for pruning. It is not the rollout objective. After the robot-local candidate set has been generated, each retained assignment action and associated scheduled is evaluated by the rollout estimator in Section 4.2, which accounts for sampled future requests, the base policy, and prediction-confidence weights.

The candidate set also includes a wait action, denoted a_ℓ^{wait}. This action assigns no request to robot ℓ during the current local decision. The wait action is always included, even when no feasible assignment action is available, so that the robot-local candidate set is never empty. If the wait action is selected, robot ℓ remains unassigned in the real system for the current decision cycle and may be reconsidered at a later decision time.

When a wait action is evaluated inside rollout, robot ℓ is temporarily treated as unavailable for the simulated completion of the current local decision. This prevents the base policy used inside the rollout simulation from immediately undoing the wait action by assigning the same robot to another pending request in the same simulated decision. The robot is not removed from the real system; it is simply left available for future decision times. We denote the set of robots that must be treated as unavailable for future cost estimation as $\mathcal{L}^{\text{block}}$.

This procedure reduces the action space used by rollout from the full feasible control set $\mathcal{U}_t(x_t)$ to a tractable robot-local candidate set $\widetilde{\mathcal{U}}_t^{\ell,i}$. The retained assignment actions are feasible for the current robot, urgent according to their latest feasible service-start times, and promising according to their heuristic delay estimates. The wait action preserves the planner's ability to defer assignment when the downstream rollout estimate indicates that keeping capacity available is preferable.

4.4. Future-Request Scenario Generation

The rollout decision rule in Section 4.2 evaluates current candidate controls by simulating possible future request arrivals. This subsection describes how those future request scenarios are generated. The scenario generator provides the interface between the learned request-prediction model and the scheduling model in Section 3: it samples future events from the estimated distribution $\widehat{\mathbb{P}}$, converts those events into request tuples with graph-grounded service locations and service windows, and returns sampled disturbance sequences that can be used directly inside rollout simulation.

Historical operating data are used to train temporal point process models for future request generation. Rather than predicting a single global request process over the entire environment, we decompose demand into local request processes. In the hospital case study, each local process corresponds to a patient encounter, but the same construction applies to any setting in which future demand can be associated

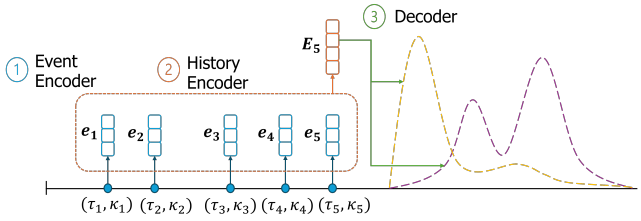

Figure 2: Schematic of the neural temporal point process used for future-request generation. Past request events are represented as a sequence of timed marks and encoded into a history-dependent representation. This representation parameterizes conditional distributions over future event times and request marks. Samples from these distributions are then converted into graph-grounded request tuples for rollout simulation.

with local entities, regions, or service contexts. Let $\mathcal{P}_t$ denote the set of local processes active at decision time t. For each process $p \in \mathcal{P}_t$, let

$$\mathcal{H}_{p,t} = \left((\tau_{p,1}, \kappa_{p,1}), \ldots, (\tau_{p,N_{p,t}}, \kappa_{p,N_{p,t}})\right) \tag{20}$$

denote the observed timed-mark history available at time t. Here, $\tau_{p,q}$ is the time of previously observed request event q and $\kappa_{p,q}$ is its mark. The mark encodes the task family or request type, together with any discrete attributes used by the trained prediction model. The number of timed marks considered in the history is denoted by $N_{p,t}$.

For each active process p, the trained temporal point process defines a conditional distribution over future timed marks. For rollout sample s, we draw

$$\left\{(\widehat{\tau}_{p,q}^s, \widehat{\kappa}_{p,q}^s)\right\}_{q\geq 1} \sim \widehat{\mathbb{P}}_p\left(\cdot \mid \mathcal{H}_{p,t}\right) \tag{21}$$

up to the truncated rollout horizon. Figure 2 illustrates this prediction module. The observed timed-mark prefix is encoded by a neural temporal point process, whose history-dependent representation parameterizes conditional distributions over future event times and marks. Samples from these conditional distributions provide the raw future demand events used by the scenario generator.

The temporal point process samples timed marks, but the rollout simulator requires request tuples of the form defined in Section 3.3. We therefore apply a deterministic request-construction map to each sampled timed mark. Given a sampled event $(\widehat{\tau}_{p,q}^s, \widehat{\kappa}_{p,q}^s)$, the mark is decoded into a task type $\widehat{k}_j$ and any task-specific attributes. The sampled time is converted into the desired service time $\widehat{t}_j^{\text{des}}$, while the entry time, earliest allowable start time, and deadline are constructed from task-specific service-window parameters. We write this conversion abstractly as

$$\widehat{r}_j = \Gamma\left(p, \widehat{\tau}_{p,q}^s, \widehat{\kappa}_{p,q}^s\right) = \left(\widehat{k}_j, \widehat{\boldsymbol{\rho}}_j, \widehat{t}_j^{\text{entry}}, \widehat{t}_j^{\text{start}}, \widehat{t}_j^{\text{des}}, \widehat{t}_j^{\text{comp}}\right). \tag{22}$$

Where Γ is a request constructor operator. The endpoint sequence $\hat{\rho}_j$ is obtained from the local process context and the task type. In the hospital case study, the process context identifies the patient location at the sampled time. For single-location service requests, this location is the service endpoint. For multi-location requests, such as delivery tasks, the endpoint sequence also includes the relevant supply or pickup location before the destination endpoint. Thus, the learned model predicts irregular timed marks, while the planner converts those marks into graph-grounded service requests using known location and task mappings.

For rollout sample s, the generator aggregates all predicted requests into a sampled future disturbance sequence

$$\widehat{\mathcal{W}}_t^s = \left(\widehat{\omega}_{t+1}^s, \widehat{\omega}_{t+2}^s, \dots, \widehat{\omega}_{\bar{T}}^s\right), \qquad \bar{T} = \min\{t+D+1, T\}, \tag{23}$$

where D is the rollout depth. Each $\widehat{\omega}_\tau^s$ is a set of predicted requests that enter the simulated system at future decision time τ. Let $\psi(\hat{r}_j)$ denote the decision time associated with the predicted entry time $\hat{t}_j^{\text{entry}}$. Then $\hat{r}_j$ is added to the bucket $\widehat{\omega}_{t'}^s$, where $t' = \psi(\hat{r}_j)$ and $\psi(\hat{r}_j) \in t+1, \dots, \bar{T}$. Algorithm 1 summarizes the scenario-generation procedure.

The generator also filters predicted requests that match requests already known to the planner. This prevents the rollout estimator from double-counting scheduled requests or real-time requests already observed in $\mathcal{R}_t$. A predicted request is treated as matching a known request if it has the same task type, compatible service-node context, and a predicted service time within a tolerance window Δ. Matched predictions are removed before the sampled disturbance sequence is passed to the rollout simulator.

Algorithm 1 Future-request scenario generation

Input: State x_t, active local processes $\mathcal{P}_t$, local histories $\mathcal{H}_{p,t}$, local TPP models $\widehat{\mathbb{P}}_p$, request-construction map Γ, rollout depth D, sample count S, match tolerance Δ.
Output: Sampled future request sequences $\widehat{\mathcal{W}}_t^1, \dots, \widehat{\mathcal{W}}_t^S$.

Set $\bar{T} \leftarrow \min\{t + D + 1, T\}$.
for $s = 1, \dots, S$ **do**
  Initialize $\widehat{\omega}_\tau^s \leftarrow \emptyset$ for $\tau = t+1, \dots, \bar{T}$.
  for each active local process $p \in \mathcal{P}_t$ **do**
    Sample future timed marks from $\widehat{\mathbb{P}}_p(\cdot \mid \mathcal{H}_{p,t})$ up to the truncated horizon.
    for each sampled timed mark $(\hat{\tau}_{p,q}^s, \hat{\kappa}_{p,q}^s)$ **do**
      Construct the predicted request $\hat{r}_j = \Gamma(p, \hat{\tau}_{p,q}^s, \hat{\kappa}_{p,q}^s)$.
      if $\hat{r}_j$ does not match a scheduled or already observed request within tolerance Δ **then**
        Add $\hat{r}_j$ to the future disturbance bucket $\widehat{\omega}_{\psi(\hat{r}_j)}^s$, if $\psi(\hat{r}_j) \in \{t+1, \dots, \bar{T}\}$.
  Form $\widehat{\mathcal{W}}_t^s = (\widehat{\omega}_{t+1}^s, \dots, \widehat{\omega}_{\bar{T}}^s)$.
return $\widehat{\mathcal{W}}_t^1, \dots, \widehat{\mathcal{W}}_t^S$.

The resulting scenarios have the same request structure as the observed request process in Section 3. Consequently, future demand is evaluated inside rollout using the same compatibility, routing, service-window, and scheduling constraints as real requests, while remaining excluded from the immediate candidate-control set.

4.5. Interaction-Aware Base Policy for Future-Cost Estimation

The one-robot-at-a-time rollout rule evaluates each candidate local control by simulating future system evolution. Because this simulation is repeated across candidate actions, robots, and sampled future request scenarios, the policy used inside rollout must be much cheaper than solving the full online scheduling problem. We therefore use an interaction-aware base policy, denoted $\bar{\pi}$, to approximate downstream assignment cost after a candidate control has been applied.

The base policy is used in two places. First, in the completion operator Complete$_{\bar{\pi}}$, it fills in simulated assignments for robots that have not yet been processed by the one-robot-at-a-time rollout rule. Second, in the resolution operator Resolve$_{\bar{\pi}}$, it attempts to resolve pending requests before the truncated terminal score is evaluated. In both cases, the base policy is used only inside rollout simulation. It is not required to produce the final executable paths applied to the real robots; instead, it provides a fast estimate of how difficult the remaining requests are likely to be after the current candidate decision.

Given a simulated state $\tilde{x}$ and a batch of requests $\mathcal{R}$, the base policy processes requests in priority order. The priority is based on the latest feasible service-start time used in Section 4.3: requests with less remaining scheduling flexibility are considered first. The batch may contain newly sampled future requests, requests left pending by earlier simulated decisions, or requests released for reconsideration in the simulated state. Predicted and observed requests are processed by the same assignment routine; prediction-confidence weights are applied later when the resulting simulated state is scored.

For each queued request r_j, the base policy considers compatible robots $\ell \in \mathcal{L}_j$. Robots that have been temporarily blocked by a wait action in the current rollout evaluation are excluded, so that the base policy cannot immediately undo the candidate wait decision being evaluated. For each remaining compatible robot, the policy estimates the completion time that would result from appending r_j to the robot's simulated schedule. This estimate is calculated using the type-dependent graph travel times through the service-node sequence ρ_j, the request service durations, release-time constraints, deadlines, and a service-node reservation table.

The service-node reservation table is the main interaction-aware component of the base policy. It records time intervals during which simulated robots are expected to occupy task service nodes. When the tentative service interval for a new request overlaps an existing reservation at the same service node, the interval is shifted to the earliest available time that satisfies the request's timing constraints. Figure 3 illustrates

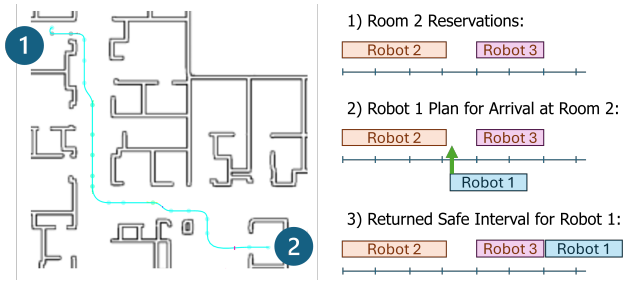

Figure 3: Reservation-table check used by the interaction-aware base policy. A candidate service interval is computed from the robot's estimated arrival time and the request execution duration. If the interval overlaps an existing reservation at the same service node, it is shifted to the earliest available interval that satisfies the request's timing constraints. The resulting interval determines the estimated completion time used to compare compatible robot assignments.

this reservation-table check. This approximation captures service-location blocking and competition for shared service locations without constructing full collision-free paths for every hypothetical future request.

After estimating feasible completion times for all compatible robots, the request is assigned to the robot with the smallest heuristic delay. If no compatible, unblocked robot can complete the request before its deadline, the request is marked as rejected or no longer serviceable in the simulated state. The resulting simulated assignments, planned completion times, rejected-request indicators, and service-node reservations are then used by the rollout estimator when computing the resolved weighted terminal score. Algorithm 2 summarizes the base-policy update.

This base policy preserves the constraints that most strongly affect downstream assignment cost: robot-task compatibility, endpoint order, release times, deadlines, service durations, robot availability, and service-location blocking. At the same time, it avoids the cost of constructing full executable schedules for every hypothetical future request in every rollout sample. Full feasibility of the immediate real candidate action is checked before rollout evaluation; the base policy approximation is used only to estimate downstream cost and to rank current candidate controls.

4.6. Adaptive Prediction Confidence and Weighted Rollout Costs

The rollout policy uses sampled future requests to estimate downstream cost, but the predictive distribution $\widehat{\mathbb{P}}$ may differ from the deployment distribution $\mathbb{P}$. If inaccurate predictions are trusted too strongly, the planner may preserve capacity for requests that do not occur or make current assignments that reflect an outdated demand pattern. We therefore assign confidence weights to predicted requests and update those weights online from recent forecast errors.

Algorithm 2 Interaction-aware base policy used inside rollout

Input: Simulated state $\widetilde{x}$, request batch $\mathcal{R}$, compatible robot sets $\mathcal{L}_j$, service-node reservation table, blocked robot set $\mathcal{L}^{\text{block}}$.

Output: Updated simulated state $\widetilde{x}'$.

Add requests in $\mathcal{R}$ to the simulated state if they are not already present.
Build a priority queue from unresolved requests, ordered by latest feasible service-start time.
while the priority queue is not empty **do**
  Remove the highest-priority request r_j.
  Initialize best robot $\ell^* \leftarrow \emptyset$, best delay $c^* \leftarrow +\infty$, and best service intervals $\mathcal{I}^* \leftarrow \emptyset$.
  for each robot $\ell \in \mathcal{L}_j \setminus \mathcal{L}^{\text{block}}$ **do**
    Estimate the earliest feasible service intervals for r_j if appended to robot ℓ's simulated schedule.
    Enforce endpoint order, release time, deadline, service duration, and service-node reservation constraints.
    if all service nodes of r_j can be served before the deadline **then**
      Compute the resulting heuristic delay c.
      if $c < c^*$ **then**
        Set $\ell^* \leftarrow \ell$, $c^* \leftarrow c$, and store the corresponding service intervals in $\mathcal{I}^*$.
  if $\ell^* = \emptyset$ **then**
    Mark r_j as rejected or no longer serviceable in the simulated state.
  else
    Assign r_j to ℓ^*, record its planned completion time, and add $\mathcal{I}^*$ to the reservation table.
return the updated simulated state $\widetilde{x}'$.

The confidence mechanism affects only hypothetical requests used inside rollout simulation. Observed real-time requests and scheduled requests always receive unit weight because they correspond to demand that is known to the planner. Predicted requests receive context-dependent weights. Let n_j denote the prediction context of a predicted request $\hat{r}_j$. A context may encode task family, request type, spatial region, floor, time of day, or any other grouping used to estimate prediction reliability. Let b_j denote the confidence-update bin associated with the predicted request $\hat{r}_j$. The confidence weight for context n in bin b is denoted by $\lambda_{n,b} \in [\lambda_{\min}, 1]$, where $\lambda_{\min} \in [0, 1]$ is the smallest prediction weight allowed by the planner. A value near one means that predictions in the context are trusted, while smaller values reduce their effect on rollout evaluation.

For a simulated state x, let $\mathcal{R}^{\text{obs}}(x)$ denote the observed or scheduled requests in the state, and let $\widehat{\mathcal{R}}(x)$ denote the predicted requests present in the simulated state. The weighted aggregate schedule score is

$$H^{\lambda}(x) = \sum_{r_j \in \mathcal{R}^{\text{obs}}(x)} H_j(x) + \sum_{\hat{r}_j \in \widehat{\mathcal{R}}(x)} \lambda_{n_j,b_j} H_{\hat{r}_j}(x). \quad (24)$$

This score is used only inside rollout simulation. It does not alter the real system state, the feasibility constraints, or the commitments made to observed requests. In the one-robot-at-a-time rollout rule, H^{λ} is used through the resolved terminal score $H^{\lambda}_{\mathrm{res}}$, so that pending predicted requests are resolved by the base policy before their weighted contribution is evaluated.

The weights are updated by comparing recent observations with prediction snapshots generated before those observations were available. Prediction confidence is evaluated on fixed bins of duration Δ_{λ}. Let ξ index the confidence-update bin, and let $\mathcal{T}_{\xi}$ denote the short set of bins used for the comparison. In the implementation, $\mathcal{T}_{\xi}$includes the target bin and adjacent bins used to tolerate small timing shifts. Let $O_{n,\iota}$ be the observed request count in context n and comparison bin $\iota \in \mathcal{T}_{\xi}$, and let $\widehat{O}^{s}_{n,\iota}$ be the corresponding predicted count in rollout sample s. The update distinguishes two forecast errors: overprediction, where predicted requests fail to occur, and timing error, where predicted demand appears near the predicted time but in an adjacent bin.

The sampled predictions define the mean predicted count

$$\mu_{n,\iota} = \frac{1}{S}\sum_{s=1}^{S} \widehat{O}^{s}_{n,\iota}. \tag{25}$$

We also compute empirical surprise scores from the sampled predictive counts. With add-one smoothing,

$$\varepsilon^{\mathrm{over}}_{n,\iota} = \frac{1 + \sum_{s=1}^{S} \mathbb{1}_{\{\widehat{O}^{s}_{n,\iota} \le O_{n,\iota}\}}}{S+1}, \tag{26}$$

$$E^{\mathrm{over}}_{n,\iota} = -\log \max\{\varepsilon^{\mathrm{over}}_{n,\iota}, \epsilon\} \tag{27}$$

$$\varepsilon^{\mathrm{under}}_{n,\iota} = \frac{1 + \sum_{s=1}^{S} \mathbb{1}_{\{\widehat{O}^{s}_{n,\iota} \ge O_{n,\iota}\}}}{S+1}, \tag{28}$$

$$E^{\mathrm{under}}_{n,\iota} = -\log \max\{\varepsilon^{\mathrm{under}}_{n,\iota}, \epsilon\}, \tag{29}$$

where $\epsilon > 0$ prevents undefined logarithms. The overprediction surprise $E^{\mathrm{over}}_{n,\iota}$ is large when the realized count is unusually low under the predictive samples, while the underprediction surprise $E^{\mathrm{under}}_{n,\iota}$ is large when the realized count is unusually high.

To separate spurious predictions from small temporal shifts, we first compute predicted and observed excess counts,

$$\varrho^{+}_{n,\iota} = [\mu_{n,\iota} - O_{n,\iota}]_{+}, \qquad \varphi^{+}_{n,\iota} = [O_{n,\iota} - \mu_{n,\iota}]_{+}, \tag{30}$$

where $[z]_{+} = \max\{z, 0\}$. Predicted excess mass in bin ι may be matched to observed excess mass in an adjacent bin ι', with $|\iota - \iota'| = 1$. Let $m_{n,\iota,\iota'}$ denote the amount of predicted excess mass in bin ι matched to observed excess mass in bin ι'. This matched mass is treated as timing error rather than as a completely spurious prediction. The remaining unmatched predicted excess is

$$P^{\mathrm{over}}_{n,\iota} = \varrho^{+}_{n,\iota} - \sum_{\iota' \in \mathcal{T}_{\xi}:\ |\iota - \iota'|=1} m_{n,\iota,\iota'}. \tag{31}$$

The instantaneous error scores are normalized by the predicted mass in the evaluated window,

$$B_{n,\xi} = \epsilon + \sum_{\iota \in \mathcal{T}_{\xi}} \mu_{n,\iota}. \tag{32}$$

The overprediction score is

$$A^{\mathrm{over}}_{n,\xi} = \frac{1}{B_{n,\xi}} \sum_{\iota \in \mathcal{T}_{\xi}} P^{\mathrm{over}}_{n,\iota} E^{\mathrm{over}}_{n,\iota}, \tag{33}$$

and the timing-error score is

$$A^{\mathrm{time}}_{n,\xi} = \frac{1}{B_{n,\xi}} \sum_{\iota \in \mathcal{T}_{\xi}} \sum_{\iota' \in \mathcal{T}_{\xi}:\ |\iota - \iota'|=1} m_{n,\iota,\iota'} \frac{E^{\mathrm{over}}_{n,\iota} + E^{\mathrm{under}}_{n,\iota'}}{2}. \tag{34}$$

Thus, $A^{\mathrm{over}}_{n,\xi}$ penalizes predicted mass that does not materialize even after adjacent-bin tolerance, while $A^{\mathrm{time}}_{n,\xi}$ penalizes predicted mass that appears shifted to a neighboring bin. Unmatched observed excess corresponds to underprediction. We record this diagnostically, but do not use it to decrease prediction confidence, because newly observed requests enter the real state with unit weight and can be handled directly by the online planner and the re-optimization mechanism.

The instantaneous scores are smoothed over time:

$$S^{\mathrm{over}}_{n,\xi} = (1 - \alpha_{\mathrm{over}}) S^{\mathrm{over}}_{n,\xi-1} + \alpha_{\mathrm{over}} A^{\mathrm{over}}_{n,\xi}, \tag{35}$$

and

$$S^{\mathrm{time}}_{n,\xi} = (1 - \alpha_{\mathrm{time}}) S^{\mathrm{time}}_{n,\xi-1} + \alpha_{\mathrm{time}} A^{\mathrm{time}}_{n,\xi}, \tag{36}$$

where $\alpha_{\mathrm{over}}, \alpha_{\mathrm{time}} \in [0, 1]$ control the responsiveness of the update. The confidence weight is then

$$\lambda_{n,\xi} = \lambda_{\min} + (1 - \lambda_{\min}) \exp\left(-\beta_{\mathrm{over}} S^{\mathrm{over}}_{n,\xi} - \beta_{\mathrm{time}} S^{\mathrm{time}}_{n,\xi}\right), \tag{37}$$

where $\beta_{\mathrm{over}}, \beta_{\mathrm{time}} \ge 0$ control how strongly the smoothed errors reduce confidence. Persistent overprediction or timing mismatch therefore lowers the weight assigned to future predicted requests in the same context. In our implementation, we typically choose $\beta_{\mathrm{over}} \ge \beta_{\mathrm{time}}$, because persistent overprediction can cause rollout to reserve capacity for demand that does not occur, whereas small timing shifts may still correspond to nearby real demand. If a prediction snapshot contains no positive predicted mass for a context, the update for that context is skipped and the previous smoothed error states are retained.

For sparse local processes, the implementation uses a hierarchical fallback to avoid overreacting to limited evidence. Let $\lambda^{\mathrm{loc}}_{n,\xi}$ be the local context weight, let $\lambda^{\mathrm{fb}}_{n,\xi}$ be a broader fallback-context weight, and let $c_{n,\xi}$ be the number of positive-prediction windows observed so far for context n. The local credibility coefficient is

$$\chi_{n,\xi} = \frac{c_{n,\xi}}{c_{n,\xi} + \zeta}, \tag{38}$$

where $\zeta \geq 0$ is a credibility prior. The effective weight applied to a predicted request is

$$\lambda^{\text{eff}}_{n,\xi} = \chi_{n,\xi}\lambda^{\text{loc}}_{n,\xi} + (1 - \chi_{n,\xi})\lambda^{\text{fb}}_{n,\xi}. \tag{39}$$

When hierarchical fallback is enabled, the symbol $\lambda_{n,\xi}$ in (24) denotes this effective weight. Thus, new or sparse contexts initially rely more heavily on a broader reliability estimate, while local context weights dominate after sufficient prediction evidence has accumulated. Algorithm 3 summarizes the confidence-update procedure.

Algorithm 3 Online prediction-confidence update

Input: Prediction snapshot $\{\widehat{O}^{s}_{n,\iota} : \iota \in \mathcal{T}_\xi\}_{s=1}^{S}$, observed counts $\{O_{n,\iota} : \iota \in \mathcal{T}_\xi\}$, previous states $S^{\text{over}}_{n,\xi-1}$ and $S^{\text{time}}_{n,\xi-1}$, hyperparameters α_{over}, α_{time}, β_{over}, β_{time}, $\lambda_{\min}$, and ϵ.

Output: Updated confidence weight $\lambda_{n,\xi}$.

1: Compute mean predicted counts $\mu_{n,\iota}$ and surprise scores $E^{\text{over}}_{n,\iota}$, $E^{\text{under}}_{n,\iota}$.
2: Compute predicted excess $\varrho^{+}_{n,\iota}$ and observed excess $\varphi^{+}_{n,\iota}$.
3: Match predicted excess to adjacent-bin observed excess, yielding timing matches $m_{n,\iota,\iota'}$.
4: Compute residual overprediction mass $P^{\text{over}}_{n,\iota}$.
5: Compute $A^{\text{over}}_{n,\xi}$ and $A^{\text{time}}_{n,\xi}$.
6: Update $S^{\text{over}}_{n,\xi}$ and $S^{\text{time}}_{n,\xi}$ using (35) and (36).
7: Compute $\lambda_{n,\xi}$ using (37).
8: Apply hierarchical fallback, if enabled, using (39).
9: **return** the effective confidence weight.

Adaptive prediction confidence provides prospective correction. It does not change real assignments already made by the planner; rather, it changes how strongly future predicted requests influence subsequent rollout evaluations. Retrospective correction of already assigned but unstarted requests is handled by the selective re-optimization mechanism in Section 4.7.

4.7. Selective Re-optimization of Unstarted Assignments

Adaptive prediction confidence provides prospective correction: it changes how strongly predicted future requests influence subsequent rollout evaluations. However, changing the prediction weights does not automatically repair assignments that were already made when those predictions appeared reliable. The planner may therefore remain committed to a schedule that is no longer desirable after new requests arrive or after recent observations reveal forecast mismatch. To address this issue, we include a selective re-optimization step for requests that have been assigned but whose service has not yet started.

Let

$$\mathcal{D}^{\text{unstarted}}_t = \left\{ r_j \in \mathcal{R}_t : a_{j,t} \in \mathcal{L}, \sigma_{j,t} = \text{pending} \right\} \tag{40}$$

denote the set of assigned but unstarted requests at decision time t. These requests may be returned to the pending set because changing their assignment does not violate the no-reassignment constraint for started requests. Requests whose service has already begun remain fixed until completion.

To decide when re-optimization is useful, we compare the urgency of newly observed requests with the urgency of assigned but unstarted requests. Let $\underline{C}_{j,t}$ be a lower bound on the earliest possible completion time of request r_j from state x_t, obtained using shortest-path travel-time estimates and the request execution duration. We define the slack index

$$\varsigma_{j,t} = t^{\text{comp}}_j - \underline{C}_{j,t}. \tag{41}$$

Smaller values of $\varsigma_{j,t}$ indicate less remaining flexibility before the request deadline. Re-optimization is considered only when both newly observed requests and assigned but unstarted requests are present.The re-optimization trigger is

$$\min_{r_j \in R^{\text{real}}_t} \varsigma_{j,t} < \min_{r_i \in \mathcal{D}^{\text{unstarted}}_t} \varsigma_{i,t}. \tag{42}$$

When (42) holds, at least one newly observed request is more urgent than every currently assigned but unstarted request. In this case, the planner releases lower-priority unstarted assignments back to the pending set. In particular, an assigned but unstarted request $r_i \in \mathcal{D}^{\text{unstarted}}_t$ is eligible for release if its slack is larger than the slack of the most urgent newly observed request: $\varsigma_{i,t} > \min_{r_j \in R^{\text{real}}_t} \varsigma_{j,t}$.

Released requests have their assignment reset to $a_{i,t} = \emptyset$ and are added to $\mathcal{D}_t$. The affected robot schedules are then repaired by removing the released requests while preserving all started or completed service commitments.

After this release step, the one-robot-at-a-time rollout policy is applied to the updated state. Newly observed requests and released unstarted requests are considered together by the candidate generator, and candidate controls are evaluated using the current prediction-confidence weights. Thus, re-optimization allows the planner to revise eligible commitments when new information makes the previous schedule undesirable.

This step provides retrospective correction. Prediction-confidence weighting reduces the influence of unreliable future predictions in subsequent rollout evaluations, whereas selective re-optimization allows the planner to recover from assignments made before those forecast errors were detected.

4.8. Supporting Pre-Deployment Fleet Sizing

The online planning problem in Section 3 is defined for a fixed heterogeneous team composition $\mathbf{M} = (M_z)_{z \in \mathcal{Z}}$. The rollout policy developed above assumes that this composition is given during deployment. However, the quality of rollout estimates depends on whether the team has enough capacity of the right robot types to serve the request patterns likely to occur in operation. If the assignment routines used inside rollout frequently reject requests or miss deadlines, then simulated future costs become dominated by infeasibility rather than by meaningful comparisons among current candidate controls. We therefore use a supporting

pre-deployment procedure to select a heterogeneous team composition from historical request sequences.

Let $\mathcal{H}^{\text{day}}$ denote a set of historical operating days. Each day $d \in \mathcal{H}^{\text{day}}$ provides a realized request sequence $\mathcal{W}^d = \left(\omega_0^d, \omega_1^d, \dots, \omega_{T-1}^d\right)$. The goal is to choose a fixed composition $\mathbf{M}$ for which the assignment routine used in simulation achieves a target empirical feasibility level $\gamma \in (0, 1]$ on these representative days. A historical day is considered feasible for composition $\mathbf{M}$ if all requests are assigned to compatible robots, completed no later than their deadlines, and all robots return to their assigned maintenance stations by time T.

Our proposed team-sizing procedure has two phases. The first phase computes day-wise feasible compositions and uses them to initialize the team size. For each historical day d, the procedure starts from a small baseline composition and simulates the assignment routine on $\mathcal{W}^d$. If a request is rejected or violates its service window, the procedure increases the count of a robot type compatible with the failed request and repeats the simulation. Once all requests on day d are served feasibly, the resulting day-wise composition is recorded as $\mathbf{M}^d = (M_z^d)_{\{z\in\mathcal{Z}\}}$. These day-wise counts estimate the amount of type-specific capacity required by individual historical operating days.

For each robot type z, we form the empirical distribution of the day-wise feasible counts $\{M_z^d : d \in \mathcal{H}^{\text{day}}\}$. Let

$$\widehat{p}_z(m) = \frac{1}{|\mathcal{H}^{\text{day}}|} \sum_{d\in\mathcal{H}^{\text{day}}} \mathbb{1}_{\{M_z^d \le m\}} \tag{43}$$

be the empirical distribution function for type z, where $\mathbb{1}_{\{M_z^d\le m\}}$ is an indicator variable. The initial team size for type z is chosen as the γ-quantile of this empirical distribution:

$$M_z^{(0)} = \min\left\{m \in \mathbb{Z}_{\ge 0} : \widehat{p}_z(m) \ge \gamma\right\}. \tag{44}$$

The resulting initialization is $\mathbf{M}^{(0)} = (M_z^{(0)})_{\{z\in\mathcal{Z}\}}$.

For example, if $\gamma = 0.95$, then $M_z^{(0)}$ is the smallest count such that at least 95% of the historical single-day sizing runs required no more than $M_z^{(0)}$ robots of type z. Algorithm 4 summarizes this initialization phase.

The second phase verifies the joint composition across all historical days. This step is necessary because the percentile initialization treats robot types marginally: choosing a high percentile for each type separately does not guarantee that the combined heterogeneous team achieves the desired feasibility rate when task interactions, service windows, routing, and robot-type substitution are evaluated jointly.

For a candidate composition $\mathbf{M}$, define the empirical feasibility rate

$$\widehat{p}_{\text{feas}}(\mathbf{M}) = \frac{1}{|\mathcal{H}^{\text{day}}|} \sum_{d\in\mathcal{H}^{\text{day}}} \mathbb{1}_{\{\text{day } d \text{ is feasible under } \mathbf{M}\}}. \tag{45}$$

where $\mathbb{1}$ is an indicator variable. The verification phase starts from $\mathbf{M}^{(0)}$ and repeatedly simulates the historical days. If $\widehat{p}_{\text{feas}}(\mathbf{M}) \ge \gamma$, the current composition is accepted.

Algorithm 4 Histogram-based initialization of the heterogeneous team composition

Input: Historical days $\mathcal{H}^{\text{day}}$, target level γ, baseline composition $\mathbf{M}^{\text{base}}$.
Output: Initial team composition $\mathbf{M}^{(0)}$.

for each historical day $d \in \mathcal{H}^{\text{day}}$ **do**
  Set $\mathbf{M}^d \leftarrow \mathbf{M}^{\text{base}}$.
  repeat
    Simulate the assignment routine on $\mathcal{W}^d$ using team $\mathbf{M}^d$.
    if all requests are completed feasibly and robots return by T **then**
      Mark day d feasible under $\mathbf{M}^d$.
    else
      Select a rejected or deadline-violating request r_j.
      Select a robot type z such that $k_j \in \mathcal{K}(z)$.
      Increase $M_z^d \leftarrow M_z^d + 1$.
  until day d is feasible
  Record $\mathbf{M}^d = (M_z^d)_{\{z\in\mathcal{Z}\}}$.
for each robot type $z \in \mathcal{Z}$ **do**
  Compute $\widehat{p}_z$ from $\{M_z^d : d \in \mathcal{H}^{\text{day}}\}$.
  Set $M_z^{(0)} \leftarrow \min\{m \in \mathbb{Z}_{\ge 0} : \widehat{p}_z(m) \ge \gamma\}$.
return $\mathbf{M}^{(0)}$.

Otherwise, the procedure identifies which robot types are associated with failed requests and increments the type with the largest failure score.

Let $\mathcal{R}_{\text{fail}}^d(\mathbf{M})$ denote the set of requests that are rejected or violate their deadlines on historical day d when using composition $\mathbf{M}$. The failure score of robot type z is

$$B_z(\mathbf{M}) = \sum_{d\in\mathcal{H}^{\text{day}}} \sum_{r_j\in\mathcal{R}_{\text{fail}}^d(\mathbf{M})} \mathbb{1}_{\{k_j\in\mathcal{K}(z)\}}. \tag{46}$$

This score counts how often robots of type z could have served failed requests. The search increases the count of a type with the largest failure score and repeats the historical verification. Algorithm 5 summarizes this phase.

This procedure does not prove feasibility under the deployment distribution $\mathbb{P}$. It selects a fixed heterogeneous team composition for which the assignment routine is empirically feasible on representative historical operating days. The selected composition is then used as the fixed team $\mathbf{M}$ assumed by the online problem formulation and by the prediction-aware rollout policy.

5. Case Study: Task Assignment in Hospital Floors

We evaluate the proposed framework on a hospital-floor task-assignment problem constructed from historical service requests on inpatient floors. This setting captures the main features of the formulation in Section 3: requests arrive over time, different task families require different robot capabilities, requests have service windows and deadlines,

Algorithm 5 Historical verification of the heterogeneous team composition

Input: Historical days $\mathcal{H}^{\text{day}}$, target feasibility level γ, initial composition $\mathbf{M}^{(0)}$.
Output: Verified team composition $\mathbf{M}$.

Set $\mathbf{M} \leftarrow \mathbf{M}^{(0)}$.
repeat
  Set $N_{\text{feas}} \leftarrow 0$ and $B_z \leftarrow 0$ for all $z \in \mathcal{Z}$.
  for each historical day $d \in \mathcal{H}^{\text{day}}$ **do**
    Simulate the assignment routine on $\mathcal{W}^d$ using team $\mathbf{M}$ and base policy $\bar{\pi}$.
    if all requests are completed feasibly and robots return by T **then**
      Set $N_{\text{feas}} \leftarrow N_{\text{feas}} + 1$.
    else
      Let $\mathcal{R}^d_{\text{fail}}(\mathbf{M})$ be the rejected or deadline-violating requests.
      for each $r_j \in \mathcal{R}^d_{\text{fail}}(\mathbf{M})$ **do**
        for each type $z \in \mathcal{Z}$ such that $k_j \in \mathcal{K}(z)$ **do**
          Set $B_z \leftarrow B_z + 1$.
  Compute $\widehat{p}_{\text{feas}}(\mathbf{M}) = N_{\text{feas}}/|\mathcal{H}^{\text{day}}|$.
  if $\widehat{p}_{\text{feas}}(\mathbf{M}) < \gamma$ **then**
    Select $z^* \in \operatorname{argmax}_{z \in \mathcal{Z}} B_z$, using a deterministic tie-breaker.
    Increase $M_z \leftarrow M_{z^*} + 1$.
until $\widehat{p}_{\text{feas}}(\mathbf{M}) \geq \gamma$
return $\mathbf{M}$.

and current assignment decisions affect the robot capacity available for future demand. It also provides a realistic setting in which the predictive request model may be useful on average but unreliable in particular time periods, floors, or task contexts.

The case study has three goals. First, we evaluate whether the temporal point process models used for scenario generation produce useful sampled future request sequences. Second, we evaluate the pre-deployment fleet-sizing procedure from Section 4.8, which selects the fixed heterogeneous team composition used by the online policies. Third, we compare the full prediction-aware adaptive rollout policy against baseline assignment policies and against ablated rollout variants. These comparisons isolate the contribution of sampled future-demand information, adaptive prediction reweighting, and selective re-optimization of unstarted assignments.

The results are presented in the same order as the components of the proposed framework. We first evaluate the temporal point process predictions used to generate future scenarios. We then report the fleet-sizing results that determine the fixed monitoring and delivery robot teams. Next, we compare the full adaptive rollout policy against baseline online assignment policies over held-out floor-days. Finally, we present ablation studies that separate the effects of prediction reweighting and re-optimization, followed by runtime results that assess whether the rollout computation is suitable for online use.

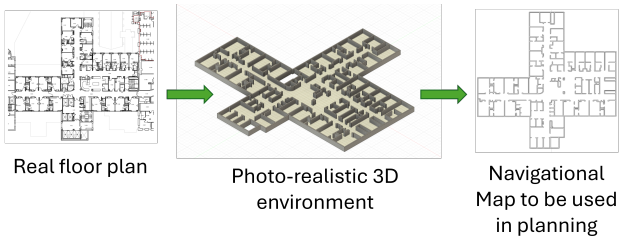

Figure 4: Environment-generation pipeline used in the case study. Starting from an architectural floor plan, we construct a simplified three-dimensional representation of the hospital floor and then extract a two-dimensional navigation map for robot simulation and planning. The resulting map defines the traversable workspace used to build the graph representation, compute travel-time heuristics, and assign service locations for scheduled, observed, and predicted requests.

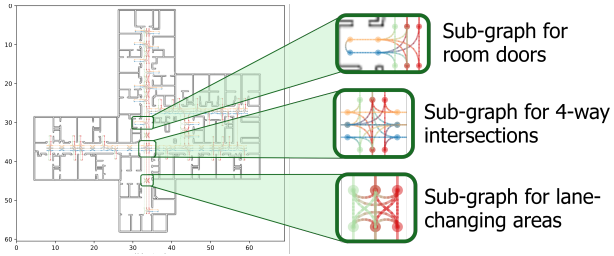

Figure 5: Example traversal graph used in the case-study simulations. Waypoint nodes and feasible local motion edges are overlaid on the hospital navigation map, with zoomed regions showing graph connectivity in representative corridor and intersection areas. The resulting graph supports shortest-path travel-time estimation and graph-constrained routing for task assignment and scheduling.

5.1. Experimental Setup

The simulated environment consists of inpatient hospital floors represented as traversal graphs. An example of how the hospital floor environments are generated is shown in Figure 4, while the extracted traversal graph for robot motion is shown in Figure 5. Nodes in the traversal graph correspond to traversable waypoints, patient rooms, supply rooms, and robot parking locations, while edges encode feasible robot motion through the floor. Each experiment uses the same graph representation and motion-planning interface for all policies. Shortest paths between any pair of nodes are cached for fast retrieval and fast heuristic travel-time estimation. Robots start from parking locations and must complete assigned service tasks while respecting routing, compatibility, timing, and return-to-station constraints.

We consider two robot families. Monitoring robots serve vital-sign requests, including blood pressure, heart rate, respiratory rate, temperature, and oxygen saturation tasks. Delivery robots serve medication requests, which require visiting a supply location before delivering to the patient-room endpoint. These two robot families define the heterogeneous team composition $\mathbf{M} = (M_z)_{\{z \in \mathcal{Z}\}}$ used in the problem

formulation: monitoring robots and delivery robots provide different capabilities and cannot generally substitute for one another.

All policies are evaluated on four inpatient floors from the west wing of BIDMC. Each floor contains approximately 25 to 30 patient beds. The historical data span June 24, 2024 through June 29, 2025. For each floor, we select six ISO weeks for evaluation: two high-demand weeks, two medium-demand weeks, and two low-demand weeks. This produces 42 test days per floor and 168 floor-day evaluations per policy. Each floor-day is simulated over a 12 hour window starting at $6am$ and ending at $6pm$, with one simulator time step corresponding to one second. The rest of the historical data is left as part of the training and validation sets. Request arrivals, service windows, patient-room endpoints, supply-room endpoints, and scheduled medication times are taken from the processed hospital historical data.

5.1.1. Constructing the Test Set

The held-out evaluation set is designed to test the policies under different demand regimes on each floor. We identify high-, medium-, and low-demand weeks using Laney u-charts [71], a statistical process-control method for identifying unusual count rates when the number of observational units may vary over time. In this setting, the unit is a floor-day, so the weekly rate is the average number of requests per floor-day for a given floor and task type. The Laney u chart adjusts the standard Poisson control limits using an estimated dispersion factor, making the demand-regime labels less sensitive to natural overdispersion or underdispersion in hospital request data [49].

For each floor and task type, we construct a weekly Laney u chart over the historical data. A week is flagged as high demand for a task type if its request rate exceeds the upper control limit, and as low demand if its request rate falls below the lower control limit. An example u-chart with flagged weeks for delivery requests is shown in Figure 6. We then aggregate flags across task types at the floor level. High-demand and low-demand weeks are selected as those with the largest number of corresponding task-type flags, while medium-demand weeks provide reference operating conditions between these extremes. This selection rule favors weeks in which the demand regime reflects a broad change in floor activity rather than an isolated anomaly in a single task type.

The selected weeks are listed in Table 1. Because the split is defined at the week level, every policy is evaluated on the same contiguous request sequences for each floor and demand regime. This preserves temporal correlations in the request stream and avoids evaluating policies on isolated requests removed from their operational context.

5.2. Implementation Details

This subsection specifies the implementation choices and parameter values used to instantiate the proposed framework in the hospital-floor case study. The goal is not to restate the algorithms in Section 4, but to describe how their inputs, hyperparameters, heuristics, and motion-planning components are configured for this domain.

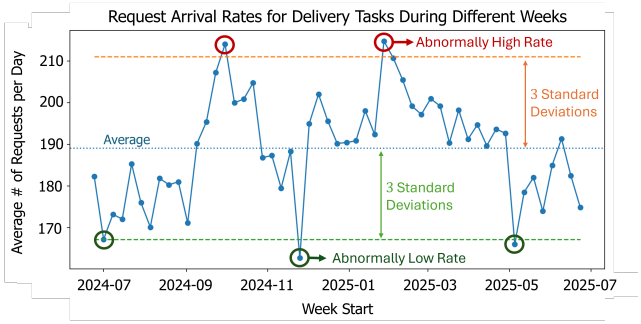

Figure 6: Example weekly delivery-request arrival rates used to select evaluation periods. The empirical mean is shown by the dotted blue line, and the dashed lines mark the Laney-adjusted upper and lower control limits. Weeks exceeding these limits are identified as abnormal high- or low-demand periods and are used to test the robustness of the proposed planner under demand shifts.

Both monitoring and delivery robots are modeled as disk robots with diameter $0.40,\mathrm{m}$. The two robot families have different capabilities: monitoring robots can service vital-sign requests, while delivery robots can service medication-delivery requests. Unless otherwise stated, both robot families use a nominal speed of $0.30,\mathrm{m/s}$. The traversal graph is constructed so that its edges correspond to physically realizable motions under the robot footprint and speed assumptions. All committed robot trajectories are planned using a variant of Safe Interval Path Planning with reservation tables, denoted SIPPwRT [61], which extends SIPP [82] time-interval reasoning with reservations induced by previously planned robot trajectories [91]. This choice is well suited to prioritized multi-robot routing because each robot can be planned sequentially while treating earlier planned trajectories as dynamic reservations.

5.2.1. One-Robot-at-a-Time Rollout Configuration

The one-robot-at-a-time rollout rule in Section 4.2 requires an ordering of robots at each decision time. In the case study, the ordering set $\mathcal{O}_t$ is constructed from the current simulator state by sorting robots according to the time at which they are expected to become available. Robots that are already available and unassigned are considered first, followed by robots whose currently assigned request is expected to finish within two minutes. Robots with longer remaining service times are not included in $\mathcal{O}_t$ at that decision time. If $\mathcal{O}_t = \emptyset$, no new assignments are made and the current schedules continue to execute.

The rollout depth is set to one hour of simulated operation. Since the simulator advances in one-second time steps, this corresponds to $D = 3600$. Thus, each rollout evaluation accounts for predicted and scheduled requests that may enter during the next hour, while avoiding the computational cost of simulating to the end of the full operating window.

Table 1
Summary of the hospital-floor test set. The table reports the selected high-, medium-, and low-demand weeks for each floor. The final row gives the total number of requests in each demand level aggregated across all task types, selected weeks, and floors.

Floor	High-demand weeks	Medium-demand weeks	Low-demand weeks
2	2025-W06, 2025-W08	2024-W44, 2025-W14	2024-W27, 2024-W28
3	2025-W05, 2025-W10	2024-W44, 2025-W14	2024-W46, 2025-W02
7	2024-W39, 2024-W40	2024-W44, 2025-W14	2024-W46, 2025-W26
9	2024-W43, 2025-W06	2024-W44, 2025-W14	2025-W19, 2025-W21
Total requests	**36,948**	**30,824**	**25,300**

5.2.2. Candidate-Control Generation

The candidate generator in Section 4.3 retains at most $L = 20$ assignment actions per processed robot. The wait action is then added separately, so the robot-local candidate set contains at most $L+1$ controls. The value $L = 20$ was chosen to preserve the most urgent and promising assignments while keeping the number of rollout evaluations manageable.

The candidate-ranking heuristics are computed as follows. The travel heuristic $F^{\text{travel}}_{\ell,j}$ is obtained from cached shortest-path distances on the traversal graph. Specifically, the shortest-path distance needed to traverse the service-node sequence $\boldsymbol{\rho}_j$ is divided by the nominal speed of robot ℓ. This computation ignores dynamic robot-robot interactions and service-node conflicts; those are handled by SIPPwRT for committed paths and by the reservation-table approximation in the rollout base policy.

The reach estimate $F^{\text{reach}}_{\ell,j,t}$ used in the heuristic completion calculation is computed with the motion planner. Candidate actions that cannot satisfy the request deadline under this heuristic estimate are discarded before rollout evaluation.

5.2.3. Future-Request Scenario Generation

Future request scenarios are generated from the temporal point process models described in Section 4.4. In the hospital case study, we instantiate one local TPP model per patient process. At each rollout decision, the planner draws $S = 20$ future-request samples. This value was chosen to represent variation in the predictive distribution while keeping online rollout runtimes reasonable.

Because each TPP prediction is associated with a patient process, the graph-grounded service nodes can be derived directly from the patient's room information at the sampled time. For vital-sign requests, the patient room is the service endpoint. For medication requests, the service-node sequence includes the relevant medication supply location followed by the patient-room endpoint.

We evaluate two mark representations for the temporal point process models. In the standard mark representation, vital-sign events are marked only by the request type, such as blood pressure, temperature, oxygen saturation, heart rate, or respiratory rate. In the enhanced mark representation, vital-sign events include both the request type and a discretized representation of the measured value, so that the model can condition on the magnitude of recent patient measurements.

Medication names are first mapped to RxNorm identifiers [72]. The medication mark is then constructed from Anatomical Therapeutic Chemical (ATC) codes [101]. In the standard medication mark representation, we use the ATC level-3 code. In the enhanced medication mark representation, we use the first three ATC hierarchy components, which provide a richer representation of the medication class while avoiding excessive sparsity at the individual-drug level.

Predicted requests that match scheduled or already observed requests are removed before rollout evaluation. The matching tolerance is set to $\Delta = 10$ minutes. Thus, a sampled prediction is removed if it corresponds to a request of the same task context that is already known to the simulator within a (10)-minute time window. This prevents the rollout estimator from double-counting demand that is already represented in the state.

5.2.4. Interaction-Aware Base Policy

The base policy used inside rollout is the interaction-aware future-cost estimator described in Section 4.5. Its reservation table is a simplified version of the full SIPPwRT reservation table used for committed motion planning. Instead of reserving all vertices and edges along complete robot trajectories, the base-policy reservation table only records service intervals at patient rooms and medication supply rooms. This approximation captures the dominant service-location interactions while avoiding the cost of planning full collision-free paths for every hypothetical future request in every rollout sample.

For each simulated assignment considered by the base policy, estimated service times are computed using cached graph shortest paths, the nominal robot speed, request execution durations, release times, deadlines, and the simplified service-node reservation table. If a tentative service interval overlaps a reservation at the same patient room or supply room, the interval is shifted to the earliest available time that satisfies the request's service-window constraints. If no feasible interval exists before the request deadline, the request is marked as rejected or no longer serviceable in the simulated state.

5.2.5. Adaptive Prediction Confidence

The adaptive prediction-confidence mechanism in Section 4.6 is enabled for rollout variants with prediction reweighting. Prediction weights are updated in bins of length $\Delta_\lambda = 5$ minutes. The minimum prediction weight is $\lambda_{\min} = 0.05$, so predicted requests can be strongly downweighed

but are not completely ignored. The exponential smoothing parameters are $\alpha_{\text{over}} = 0.3$ and $\alpha_{\text{time}} = 0.2$. Thus, the overprediction state reacts somewhat faster than the timing-error state. The confidence-decay parameters are $\beta_{\text{over}} = 4.0$ and $\beta_{\text{time}} = 1.0$, which makes persistent overprediction reduce confidence more strongly than one-bin timing shifts. The numerical constant used for normalization and logarithmic clipping is $\epsilon = 10^{-6}$. Hierarchical fallback is enabled in all rollout variants that use adaptive prediction confidence. The local-context credibility prior is $\zeta = 3.0$. Therefore, patient-specific confidence weights are initially blended strongly with the broader fallback context, and they dominate only after enough positive-prediction windows have been observed. Confidence updates are skipped for zero-prediction windows, so contexts are not penalized or rewarded when the prediction snapshot contains no positive predicted mass.

5.2.6. Scheduled Requests, Re-optimization, and Decision Triggers

Known future scheduled requests are included in the rollout future-cost estimate for the proposed rollout variants. They are treated as known commitments and receive unit weight in the weighted schedule score. However, they are not added to the immediately assignable request set until they enter the simulator state. In contrast, stochastic TPP predictions are never eligible as immediate assignment actions; they affect decisions only through sampled rollout simulations and confidence-weighted future costs.

Selective re-optimization is enabled only in rollout variants that include the mechanism described in Section 4.7. When enabled, assigned but unstarted requests may be released back to the pending set if newly observed requests have smaller slack. Started requests are never released or reassigned.

5.3. Temporal-Point-Process Prediction Ablation

Before evaluating the online assignment policies, we evaluate the temporal point process models used to generate future request scenarios in Section 4.4. This ablation serves two purposes. First, it identifies the prediction model configuration used by the rollout policies in the remaining experiments. Second, it tests whether richer timed-mark representations and previous-day context improve the quality of sampled future request sequences in the hospital setting.

The ablation compares the candidate temporal point process models used to sample future request sequences. The model set includes: RMTPP [29], NHP [66], FullyNN [77], SAHP [110], THP [116], IntesityFree [87], AttnNHP [67], ANHN [104], WSM-THP [18], S2P2 [19], TriTPP [88], inhomogeneous Poisson [88], Renewal [88] , Modulated Renewal [88], Spline Transformer [88], and FlexTPP-based variants [28]. The implementation of the first 8 methods was derived from [103], while the implementation for the rest of the methods was derived from their respective papers. The only modifications applied to all architectures were dimensionality adjustments needed to process the enhanced marks. All models under consideration were tested for both enahnced and standard marks. The standard mark representation uses only the request type for vital-sign requests and a coarser medication-code representation for medication requests. The enhanced mark representation augments vital-sign marks with discretized measurement information and uses a richer medication-code representation, as described in Section 5.2.

For the FlexTPP variants, we also compare models trained with and without previous-day summary conditioning. Previous-day conditioning summarizes the recent patient-level history available before the prediction prefix. For vital-sign requests, this includes whether previous-day context is available, the time since the last previous-day request, total previous-day request count, task-specific request counts, task-specific last request times, and summary statistics of previous-day measurements. For medication requests, the conditioning vector summarizes previous-day vital-sign and medication-administration activity for the same local patient process.

We train all architectures for 300 epochs using the training set. We select the log likelihood as the optimization objective for each temporal point process, and we use Adam [47] as the optimizer. The learning rate for each architecture is chosen using a hyper-parameter search. We choose a batch size of 32, and we shuffle the batches at every epoch during training. The number of layers and the width of each layer for each architecture are chosen following the recommendations given in their respective papers. All training was done using two NVIDIA RTX A6000 ADA.

Prediction quality is evaluated by comparing sampled future event sequences against the realized future sequence. For a local process p and prediction prefix ending at time t, let

$$\widehat{W}^{s}_{p,t} = \left(\widehat{e}^{s}_{1}, \dots, \widehat{e}^{s}_{\widehat{N}_s}\right), \qquad W_{p,t} = \left(e_1, \dots, e_N\right) \quad (47)$$

denote the s-th sampled predicted suffix and the corresponding realized suffix, where $\widehat{N}_s$ is the number of predicted events and N is the actual number of observed events. Each event is a timed mark $e = (\tau, k, \kappa)$, where τ is the event time, k is the request type, and κ is the model-specific mark representation. We evaluate each sampled suffix using a marked Optimal Transport Distance (OTD), implemented as an ordered edit-alignment distance between $\widehat{W}^{s}_{p,t}$ and $W_{p,t}$.

Matching a predicted event $\widehat{e}^{s}_{i}$ to a realized event e_j incurs the substitution cost

$$d_{\text{sub}}\left(\widehat{e}^{s}_{i}, e_j\right) = \alpha_{\text{OTD}} \frac{\left|\widehat{\tau}^{s}_{i} - \tau_j\right|}{\tau^{\text{scale}}_{k_j}} + \beta_{\text{OTD}} \mathbb{1}_{\left\{\widehat{k}^{s}_{i} \neq k_j\right\}}, \quad (48)$$

where $\tau^{\text{scale}}_{k_j}$ is a task-specific time scale used to normalize timing errors. Predicted events that cannot be matched to realized events incur deletion cost c_{del}, and realized events that are missing from the prediction incur insertion cost c_{ins}.

The sequence-level OTD is

$$\mathrm{OTD}\left(\widehat{W}^{s}_{p,t}, W_{p,t}\right) = \min_{\mathcal{A}}\Bigg[\sum_{(i,j)\in\mathcal{A}_{\mathrm{match}}} d_{\mathrm{sub}}\left(\hat{e}^{s}_{i}, e_{j}\right) + c_{\mathrm{del}}|\mathcal{A}_{\mathrm{del}}| + c_{\mathrm{ins}}|\mathcal{A}_{\mathrm{ins}}|\Bigg], \quad (49)$$

where $\mathcal{A}$ ranges over all ordered edit alignments between the sampled and realized suffixes. Lower OTD values indicate better agreement in event timing, task type, and sequence length. For each prediction prefix, we estimate the expected OTD over sampled futures by

$$\widehat{\mathrm{OTD}}_{p,t} = \frac{1}{S_{\mathrm{eval}}}\sum_{s=1}^{S_{\mathrm{eval}}} \mathrm{OTD}\left(\widehat{W}^{s}_{p,t}, W_{p,t}\right), \quad (50)$$

where S_{eval} is the number of sampled futures used for the prediction evaluation.

For all OTD results reported below, we use $S_{\mathrm{eval}} = 20$, $\alpha_{\mathrm{OTD}} = 1.0$, $c_{\mathrm{del}} = 1.0$, and $c_{\mathrm{ins}} = 1.0$. The timing weight α_{OTD} controls the contribution of normalized timing error for matched events. The deletion penalty c_{del} penalizes spurious predicted events, and the insertion penalty c_{ins} penalizes realized events that were missed by the prediction. The normalizing constant $\tau^{\mathrm{scale}}_{k_j}$ is set to the mean inter-event time for request type k_j in the training data.

For monitoring requests, we use a finite request-type substitution penalty $\beta_{\mathrm{OTD}} = 0.25$. This allows the alignment to match different vital-sign request types at a small cost, reflecting that the same monitoring robot can service all vital-sign tasks. The resulting OTD decomposes into timing, request-type mismatch, deletion, and insertion components. For medication-delivery requests, we use hard type matching: substitutions between different medications are disallowed. This reflects that different medication requests should not be treated as interchangeable. Therefore, the delivery OTD decomposes into timing, deletion, and insertion components, without a finite request-type mismatch component.

The OTD evaluation is stratified by the same high-, medium-, and low-demand weeks used in the assignment-policy experiments. Because the qualitative conclusions were similar across load regimes, Figures 7 and 8 report the high-demand results, where prediction errors are most likely to affect downstream assignment decisions. Figure 7 summarizes monitoring-request prediction, and Figure 8 summarizes medication-delivery prediction.

For monitoring requests, Figure 7 shows that the FlexTPP model with enhanced marks and previous-day context obtains the lowest OTD among the evaluated configurations. This result indicates that measurement-aware marks and recent patient-level history provide useful signal for predicting future vital-sign request sequences, provided that the model architecture can use the additional conditioning information effectively.

For medication-delivery requests, Figure 8 shows that FlexTPP-based models also outperform the other evaluated prediction models. However, unlike the monitoring case, the enhanced mark representation and previous-day context do not consistently improve OTD. One likely explanation is that medication-code features are substantially sparser than vital-sign marks, so the richer representation increases feature granularity without providing enough repeated evidence for the model to exploit. Based on these results, the rollout experiments use the best-performing monitoring and medication prediction configurations identified by this ablation to generate future request scenarios.

Table 2
Selected heterogeneous team composition.

Phase	Monitoring robots	Delivery robots
Histogram initialization	6	2
Joint historical verification	6	3

5.4. Team-Sizing Results

We next evaluate the historical team-composition procedure described in Section 4.8. This step determines the fixed heterogeneous team $\mathbf{M}$ used in the subsequent online policy experiments. The procedure is important because monitoring and delivery robots provide different capabilities: additional delivery robots cannot resolve monitoring bottlenecks, and additional monitoring robots cannot resolve medication-delivery bottlenecks. The sizing procedure must therefore select enough capacity for each robot family while also verifying that the resulting joint team composition is feasible when all request types, timing constraints, and routing interactions are evaluated together.

Figure 9 shows the day-wise robot requirements produced by the histogram-based initialization phase in Algorithm 4. For each historical floor-day, the sizing routine increases the number of robots in the relevant family until all requests of that family can be served feasibly. The monitoring histogram summarizes the number of monitoring robots required across historical floor-days, and the delivery histogram summarizes the corresponding requirement for medication-delivery robots. Using the 99-th percentile of these empirical distributions gives an initial team composition of 6 monitoring robots and 2 delivery robots.

The initialized composition is then checked using the joint historical verification phase described in Algorithm 5. Table 2 summarizes the composition before and after this verification step. The monitoring fleet remains unchanged at 6 robots, while the delivery fleet increases from 2 to 3 robots.

The increase in delivery capacity illustrates why the verification phase is necessary: marginal percentile estimates for each robot family do not necessarily guarantee that the combined team achieves the target empirical feasibility level once shared timing constraints, routing interactions, and multi-request schedules are evaluated jointly. The verified team composition $\mathbf{M} = (6, 3)$ is used as the fixed heterogeneous robot team in the remaining policy-comparison

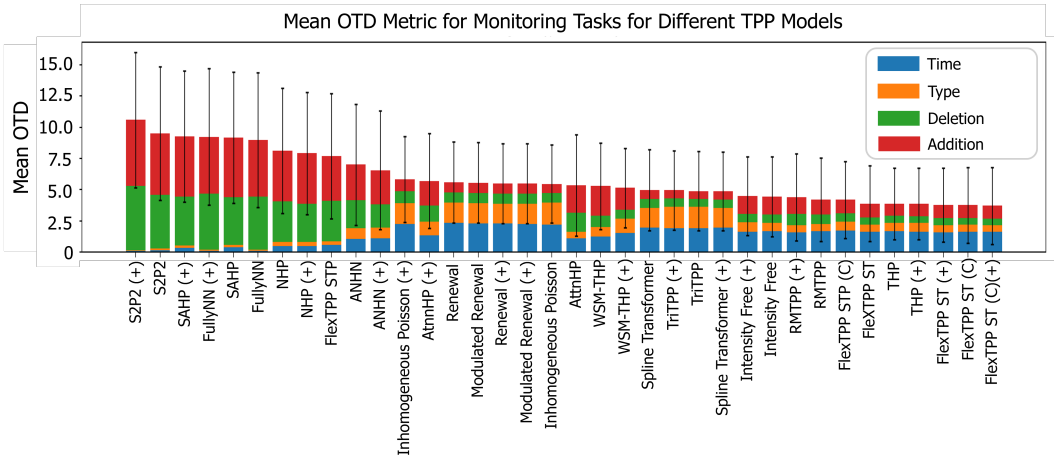

Figure 7: Comparison of temporal point process (TPP) models for monitoring-request prediction using the OTD metric. Each stacked bar reports mean OTD for one model configuration, decomposed into timing, request-type, deletion, and insertion components. Error bars indicate variability across evaluation samples. Lower values indicate better agreement between predicted and observed request sequences, while smaller deletion and insertion components correspond to fewer missed or spurious predicted events. TPP model names and references are given at the beginning of Section 5.3. In the model labels, (+) denotes the use of enhanced marks, (C) denotes conditioning on previous-day information, and $(C)(+)$ denotes the use of both enhanced mark and previous day conditioning.

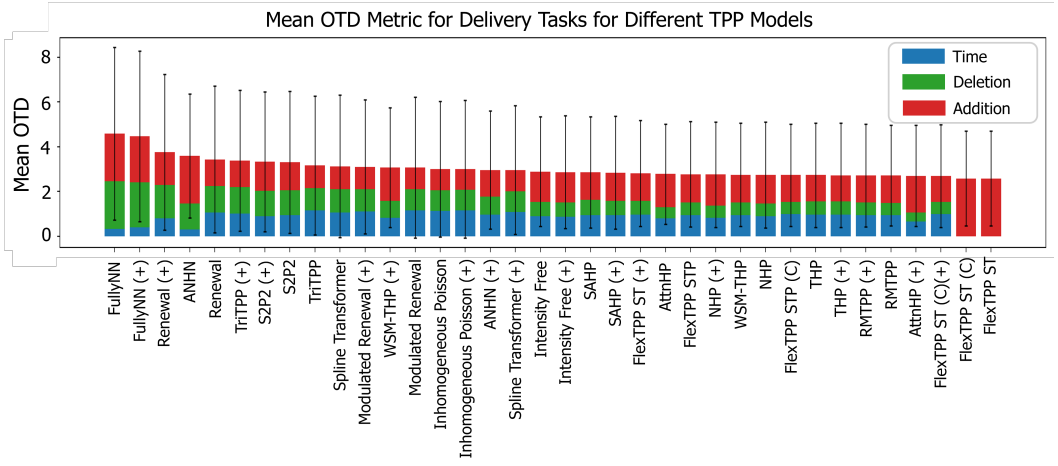

Figure 8: Comparison of temporal point process (TPP) models for medication-delivery prediction using the OTD metric. Each stacked bar reports mean OTD for one model configuration, decomposed into timing, request-type, deletion, and insertion components. Error bars indicate variability across evaluation samples. Lower values indicate better agreement between predicted and observed request sequences, while smaller deletion and insertion components correspond to fewer missed or spurious predicted events.TPP model names and references are given at the beginning of Section 5.3. In the model labels, (+) denotes the use of enhanced marks, (C) denotes conditioning on previous-day information, and $(C)(+)$ denotes the use of both enhanced mark and previous day conditioning.

and ablation experiments, so subsequent performance differences reflect the online assignment policies rather than differences in available fleet capacity.

5.5. Policy Comparison

We next compare the proposed adaptive rollout policy against reactive, deadline-aware, prediction-based, and myopic assignment baselines. All policies are evaluated on the same held-out floor-days defined in Section 5.1.1, using the fixed heterogeneous team composition selected in Section 5.4. Thus, differences in performance reflect the online assignment policy rather than differences in fleet capacity, request streams, traversal graphs, or robot capabilities.

The main proposed method is the adaptive rollout policy with future scheduled requests, temporal-point-process prediction scenarios, adaptive prediction reweighting, and selective re-optimization enabled. This policy uses predicted requests only inside rollout simulations; immediate assignment actions are restricted to requests that have already entered the system.

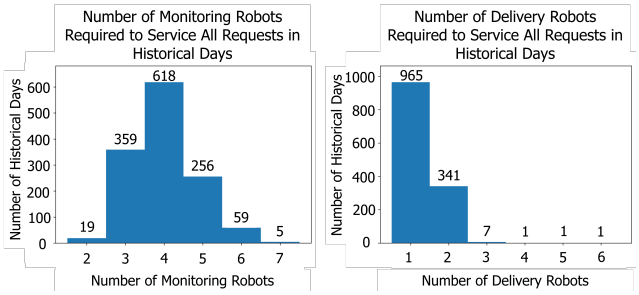

Figure 9: Day-wise robot requirements obtained during team-size initialization. For each historical day, the sizing routine increases the number of monitoring or delivery robots until all requests are feasibly served. The histograms summarize the resulting feasible robot counts across historical days and are used to initialize the heterogeneous team composition at the desired empirical percentile.

5.5.1. Baselines for Comparison

The comparison includes the following policies.

Fleet Manager. The Fleet Manager baseline is a reactive queue-based policy adapted from [83]. Requests are processed according to their release times. When a compatible robot is available, the policy assigns the next request to the available robot that can reach it most quickly. A collision-free path is then planned for the assignment, while previously committed paths are treated as fixed and immutable. This baseline represents a simple operational dispatching strategy that does not explicitly reason about future requests.

Token Passing (TP). The Token Passing baseline follows the multi-agent task-allocation framework of [62]. Robots request a shared token when they are available. When a robot holds the token, it selects an unassigned task using a local cost based on travel distance to the task, without explicitly accounting for deadline urgency or future demand. After the task is selected, the robot plans a path to the task location while avoiding the trajectories of other robots. This baseline captures a standard decentralized assignment strategy with prioritized path planning.

Token Passing with Task Swaps (TPTS). The Token Passing with Task Swaps baseline extends token passing by allowing unstarted tasks to be reassigned when a different robot can serve them more effectively [62]. When a robot considers a task that is already assigned but not yet started, the policy may remove that assignment and assign the task to the current robot, provided that the displaced robot can obtain another feasible assignment. This baseline introduces limited reassignment flexibility while remaining reactive and myopic.

Token Passing with Deadlines (TP-D). The deadline-aware token-passing baseline incorporates request deadlines into the task-selection objective [63]. Instead of ranking tasks only by travel cost, the policy uses a weighted score that combines travel cost and deadline urgency. We set the deadline-weight parameter to $\alpha = 0.2$, which was selected by grid search over $\alpha \in 0.1, 0.2, 0.3, 0.4$ on the tuning runs. This baseline tests whether explicit deadline awareness improves service reliability relative to distance-based token passing.

Deadline-Aware Token Passing with Task Swaps (D-TPTS). This baseline combines deadline-aware task scoring with reassignment of unstarted tasks [63]. Newly available robots request the token, evaluate deadline-aware assignment costs, and may deallocate lower-priority unstarted tasks when a more urgent task arrives. We again use $\alpha = 0.2$, selected from the same grid search. This policy is the strongest token-passing baseline because it combines deadline awareness with limited schedule repair.

Idle Rebalance. The Idle Rebalance baseline uses predictions of future requests to reposition idle robots [30]. When robots are idle, predicted requests can act as pseudo-tasks that influence robot positioning. This differs from the proposed rollout policy: in our method, predicted requests are never immediate assignment actions and affect decisions only through the simulated future-cost estimate.

Greedy Assignment Policy (Base Policy). The greedy assignment baseline is the standalone myopic policy that we proposed as the base policy for rollout in Section 4.5. It orders currently known requests by urgency, evaluates compatible robots, and assigns the request to the robot that results in the lowest heuristic assignment cost.

Proposed Adaptive Rollout (Our Approach). The proposed method is the prediction-aware adaptive rollout policy developed in Section 4. It restricts immediate candidate actions to currently observed eligible requests, evaluates each candidate through sampled future request scenarios, scores predicted requests using adaptive confidence weights, and selectively re-optimizes assigned but unstarted requests when newly observed demand makes the current schedule undesirable.

5.5.2. Performance Metrics

We evaluate policies using service-level, service-quality, and computational metrics.

The service-level metrics are the number of serviced and rejected requests. A request is counted as serviced if it is assigned to a compatible robot and completed within its service window. A request is counted as rejected if the policy cannot assign it feasibly before its deadline. We report these counts separately for high-, medium-, and low-demand days, because policy differences are expected to be most pronounced when robot capacity and request deadlines are most constrained.

The service-quality metrics are computed over requests that are successfully serviced. For each serviced request r_j, the wait time is $h_j = \max\{0, C_j - t_j^{\text{des}}\}$, where C_j is the realized completion time and t_j^{des} is the desired service time. For

Table 3
Service-level results by demand regime. Each entry reports serviced and rejected requests for the corresponding policy and load level.

Policy	High demand		Medium demand		Low demand	
	Serviced	Rejected	Serviced	Rejected	Serviced	Rejected
Fleet Manager	36,948	0	30,824	0	25,300	0
Token Passing	36,920	28	30,824	0	25,300	0
Token Passing with Task Swaps	36,920	28	30,824	0	25,300	0
Token Passing with Deadlines	36,948	0	30,824	0	25,300	0
Deadline-Aware Token Passing with Task Swaps	36,948	0	30,824	0	25,300	0
Idle Prediction	36,948	0	30,824	0	25,300	0
Greedy Assignment Policy	36,948	0	30,824	0	25,300	0
Proposed Adaptive Rollout	36,948	0	30,824	0	25,300	0

each policy and day, we compute the mean serviced-request wait time and the 95-th percentile serviced-request wait time. We then plot the distribution of these daily statistics across days, stratified by demand regime. The mean wait-time plots summarize typical service responsiveness, while the 95-th percentile plots capture tail delays.

The computational metric is the planning time required to generate a decision for one request. We report the distribution of per-request planning times across the evaluation runs. This metric is important because the proposed rollout policy evaluates multiple candidate actions over sampled future scenarios, whereas the baselines generally make more local decisions.

5.5.3. Service-Level Results

Table 3 reports the number of serviced and rejected requests for each policy, stratified by demand regime. This metric evaluates whether each policy can maintain feasibility with the heterogeneous team selected in Section 5.4. High-demand days are the most informative for service-level performance because robot availability and deadline constraints are most likely to become binding. Medium- and low-demand days test whether the same policies remain feasible when the system is less capacity constrained.

The selected team composition is sufficient to service all requests for most policies across all demand regimes. The only service-level failures occur on high-demand days for Token Passing and Token Passing with Task Swaps, which reject 28 requests out of 36,948 high-demand requests. These rejections do not occur for the deadline-aware token-passing variants, the Fleet Manager baseline, Idle Prediction, the Greedy Assignment Policy, or the proposed adaptive rollout policy.

This result has two implications. First, it supports the fleet-sizing procedure: the selected team composition provides enough monitoring and delivery capacity to avoid persistent overload on the held-out floor-days. Second, it shows that aggregate rejection counts alone do not fully distinguish the stronger policies in this experiment, because most policies achieve zero rejections once the fleet is properly sized. The remaining comparisons therefore focus on service quality and computational cost. In particular, the wait-time results below evaluate whether policies that service the same number of requests differ in how close they complete those requests to their desired service times.

5.5.4. Request Wait-Time Results

The service-level results in Table 3 show that most policies complete nearly all requests under the selected team composition. We therefore use wait-time metrics to compare the quality of the schedules produced by the policies. For each serviced request r_j, wait time is computed as $h_j = \max 0, C_j - t_j^{\text{des}}$, where C_j is the realized completion time and t_j^{des} is the desired service time. For each floor-day, we compute two daily summary statistics: the mean wait time over serviced requests and the 95-th percentile wait time over serviced requests. The box plots below summarize the distribution of these daily statistics across evaluation instances for each demand regime.

Figure 10 shows the wait-time comparison for the highest-demand weeks. In this regime, robot capacity and deadline constraints are most restrictive, so assignment decisions have the largest effect on downstream congestion. The proposed adaptive rollout policy achieves the lowest median wait time among the evaluated methods for both the daily mean and daily 95-th percentile metrics. The median improvement is approximately 5% for the mean wait-time metric and approximately 10% for the 95-th percentile metric. The reduction is even more pronounced in the upper tail: the proposed method reduces the 75-th percentile and maximum values of both wait-time metrics by more than 15% relative to the baselines.

These high-demand results indicate that the main benefit of prediction-aware rollout is not only a reduction in typical delay, but also a reduction in severe delay. This is consistent with the role of rollout: candidate assignments are evaluated based on their downstream effect on future capacity, which helps avoid decisions that are locally feasible but create later bottlenecks. The comparison with Idle Prediction is also informative. Although Idle Prediction uses future-demand information, it treats predicted requests as positioning targets. Its weaker performance shows that inaccurate or mistimed predictions can bias robot movement when predictions are treated too directly. The proposed rollout policy instead uses

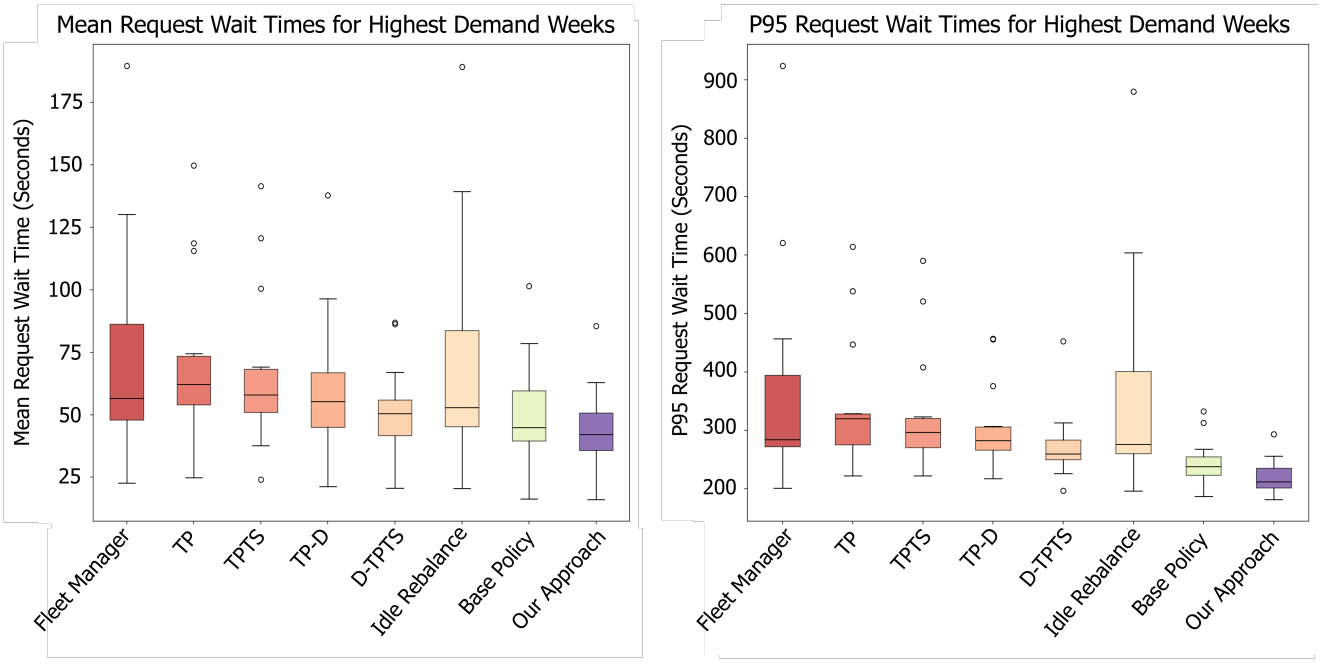

Figure 10: Comparison of request wait times across assignment policies during the highest-demand evaluation weeks. The left panel reports the distribution of daily mean wait times for serviced requests, and the right panel reports the distribution of daily 95-th percentile wait times for serviced requests. Each box summarizes performance across evaluation instances, with lower values indicating shorter delays. Abbreviated policy labels correspond to the assignment policies defined in Section 5.5.1.

predictions as uncertain inputs to future-cost estimation, with adaptive weighting and re-optimization limiting the effect of unreliable forecasts.

Figure 11 reports the same comparison for medium-demand weeks. This regime represents nominal operating conditions, where the system is less congested than in the high-demand case but still has enough request volume for assignment quality to matter. The proposed adaptive rollout policy again obtains the lowest median wait time for both metrics. The median reduction is approximately 2% for daily mean wait time and approximately 5% for daily 95-th percentile wait time. At the 75-th percentile of the daily distributions, the improvement is approximately 5% for the mean metric and approximately 15% for the 95-th percentile metric.

The medium-demand results show that rollout remains useful even when the system is not persistently overloaded. Under these conditions, future request predictions are generally informative enough to improve assignment timing, but direct prediction-based repositioning is still less effective than evaluating sampled futures through a scheduling-aware cost estimate. The proposed method benefits from this distinction: it uses multiple sampled futures and the interaction-aware base policy to estimate downstream cost, rather than greedily moving robots toward individual predicted requests.

Figure 12 shows the results for the lowest-demand weeks. In this regime, more robots are idle for longer periods, so even simple policies can often complete requests without rejection. Nevertheless, the proposed adaptive rollout policy still achieves the lowest median wait time for both the mean and 95-th percentile metrics. The median improvement is approximately 2% for daily mean wait time and approximately 7% for daily 95-th percentile wait time. At the 75-th percentile, the proposed method reduces both metrics by approximately 10% relative to the baselines.

The low-demand results show that the proposed method does not rely only on congestion to provide benefit. When capacity is abundant, the main opportunity is to position or preserve robots so that requests can be served quickly after they arrive. Prediction-aware methods can help in this setting, but only if the planner avoids overcommitting to uncertain predictions. The proposed rollout policy improves over both reactive baselines and the Idle Prediction baseline because it evaluates predicted demand through sampled future costs while keeping immediate assignments restricted to requests that have actually entered the system.

Across all demand regimes, the largest and most consistent gains appear in the 95-th percentile wait-time metric. This suggests that the proposed adaptive rollout policy is particularly useful for reducing tail delays among serviced requests. Since most policies achieve similar service levels under the selected fleet size, these wait-time results provide the strongest evidence that prediction-aware rollout improves schedule quality beyond simply completing requests.

5.5.5. Policy Planning-Time Results

Finally, we evaluate the computational cost of each policy. This comparison is important because the proposed

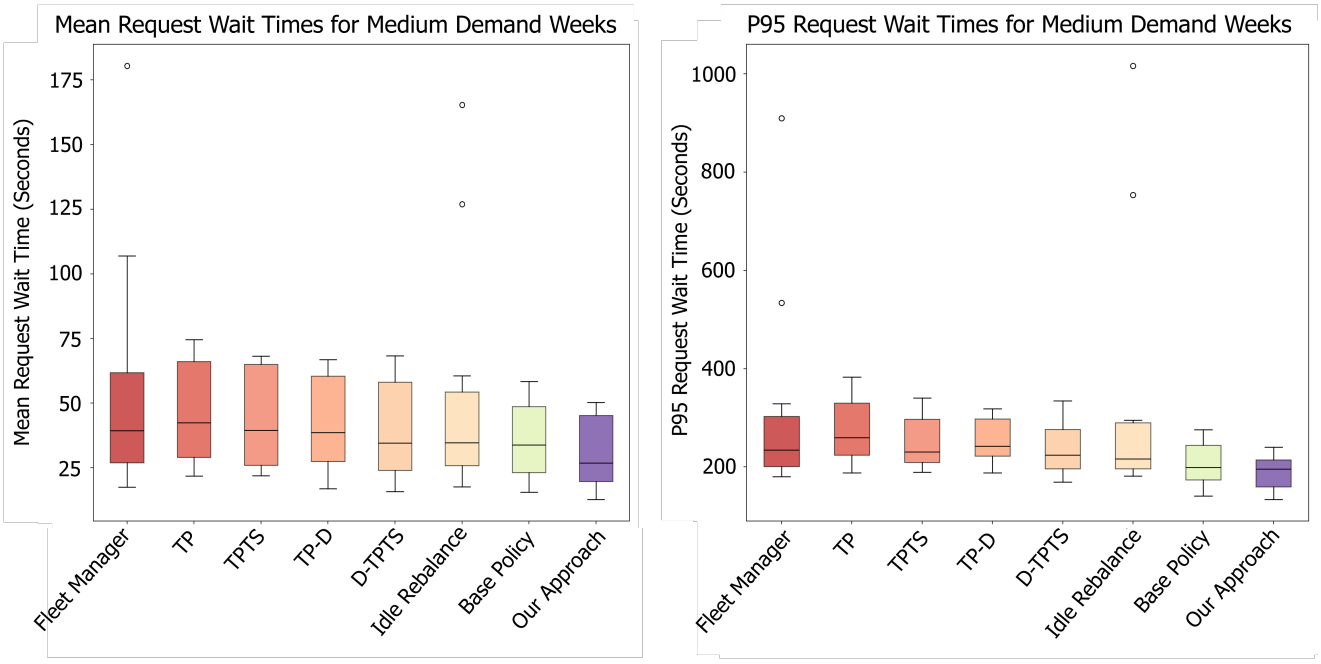

Figure 11: Comparison of request wait times across assignment policies during medium-demand evaluation weeks. The left panel reports the distribution of daily mean wait times for serviced requests, and the right panel reports the distribution of daily 95-th percentile wait times for serviced requests. Each box summarizes performance across evaluation instances, with lower values indicating shorter delays. Abbreviated policy labels correspond to the assignment policies defined in Section 5.5.1.

adaptive rollout policy performs substantially more computation than the reactive baselines: it generates robot-local candidate actions, evaluates those candidates over sampled future request scenarios, applies the interaction-aware base policy inside each rollout simulation, and scores the resulting simulated states. The candidate limit L, sample count S, one-robot-at-a-time decision rule, and decision-suppression mechanism described in Section 5.2 are therefore used to keep the computation compatible with online execution.

Figure 13 reports the distribution of planning time required to process a request-triggered decision across the evaluated policies. As expected, the reactive and token-passing baselines require less computation because they make local assignment decisions without simulating sampled future demand. The proposed adaptive rollout policy has the largest planning time among the evaluated methods, but its median planning time remains approximately 2 minutes per request-triggered decision. In the hospital case study, requests have a 5-minute lead time between entry into the system and the earliest time at which they may be serviced. Thus, the observed median planning time remains within the operational window available for online decision making.

These results show that the improved wait-time performance of the proposed policy comes with a measurable computational cost. However, the runtime remains practical for the simulated hospital setting under the chosen rollout configuration. The implementation used for these experiments evaluates rollout simulations sequentially, even though many parts of the computation are naturally parallelizable. In particular, different sampled future scenarios, and in some cases different candidate-action evaluations, can be evaluated independently. A parallel implementation would therefore be expected to reduce wall-clock planning time without changing the policy logic. The reported runtimes should consequently be interpreted as a conservative estimate of the computational cost of the proposed rollout policy.

5.6. Adaptive Rollout Ablation

The ablation study isolates the two adaptive mechanisms introduced in Sections 4.6 and 4.7: adaptive prediction reweighting and selective re-optimization of unstarted assignments. All variants use the same one-robot-at-a-time rollout framework, the same candidate-generation procedure, the same future scheduled requests, and the same sampled prediction scenarios. The only difference across variants is whether predicted request costs are reweighted online and whether assigned but unstarted requests can be released for reconsideration.

We evaluate four rollout variants:

Rollout without adaptation. This variant uses future scheduled requests and sampled predicted requests in the rollout future-cost estimate, but prediction costs are not downweighted online and assigned but unstarted requests are not reconsidered.

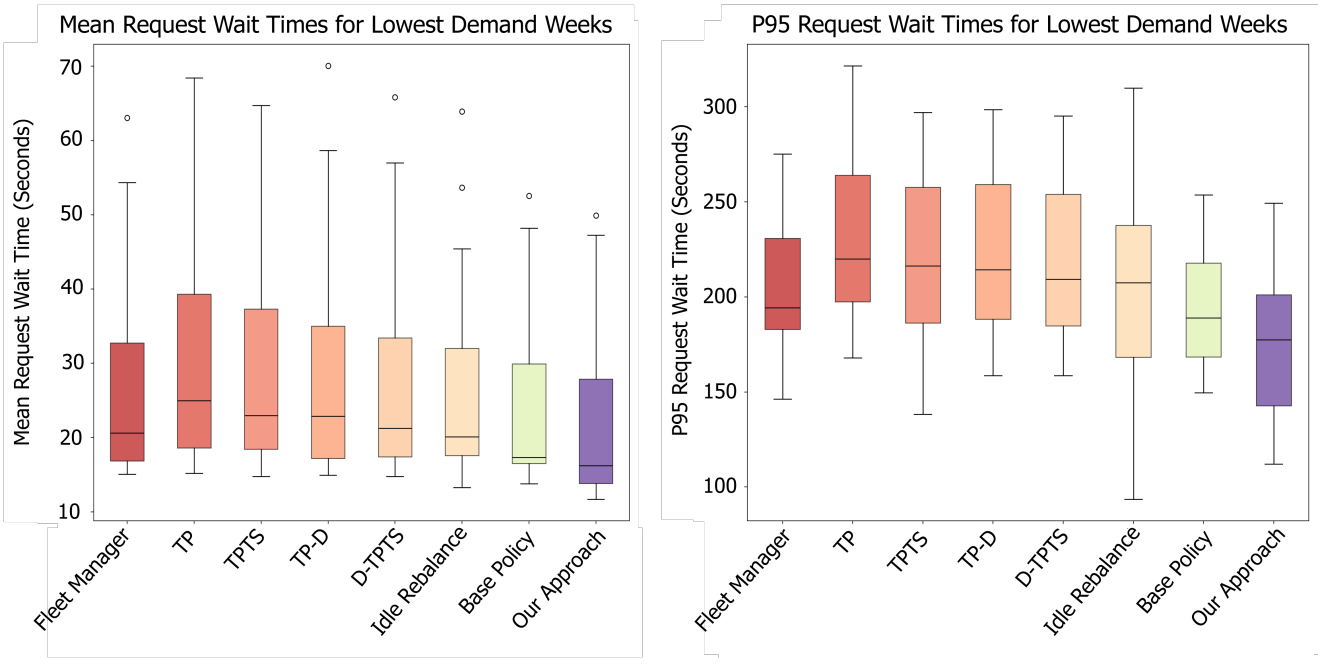

Figure 12: Comparison of request wait times across assignment policies during the lowest-demand evaluation weeks. The left panel reports the distribution of daily mean wait times for serviced requests, and the right panel reports the distribution of daily (95)-th percentile wait times for serviced requests. Each box summarizes performance across evaluation instances, with lower values indicating shorter delays. Abbreviated policy labels correspond to the assignment policies defined in Section 5.5.1.

Table 4
Ablation study of the adaptive rollout components. All variants use the same rollout framework and include future scheduled requests in the future-cost estimate. The table reports the mean and standard deviation of request wait times, in seconds, for each demand level.

Policy	High demand		Medium demand		Low demand	
	Mean	Std.	Mean	Std.	Mean	Std.
Rollout without adaptation	49.25	90.02	34.23	78.21	25.65	68.72
Rollout with re-optimization only	46.78	86.88	32.18	74.08	24.36	65.97
Rollout with prediction reweighting only	49.39	90.49	33.70	77.25	25.17	67.69
Proposed adaptive rollout	**46.01**	**85.41**	**31.88**	**73.54**	**24.28**	**65.91**

Rollout with re-optimization only. This variant allows assigned but unstarted requests to be returned to the pending set when newly observed requests make the previous assignment order undesirable, but predicted request costs are not adaptively reweighted.

Rollout with prediction reweighting only. This variant updates the contribution of predicted requests to the rollout objective using recent forecast errors, but it does not release previously assigned unstarted requests.

Proposed adaptive rollout. This is the full proposed policy. It combines adaptive prediction reweighting with selective re-optimization of unstarted assignments.

Table 4 reports the mean and standard deviation of request wait times for each demand regime. The mean captures average service responsiveness, while the standard deviation captures the spread of request delays. Lower values are better for both metrics.

The full adaptive rollout policy achieves the lowest mean wait time and the lowest wait-time standard deviation in every demand regime. Relative to rollout without adaptation, the full policy reduces mean wait time by approximately 6% under high demand, 7% under medium demand, and 5% under low demand. It also reduces the standard deviation of wait times by approximately 5%, 6%, and 4%, respectively. These reductions show that the adaptive mechanisms improve both average responsiveness and delay variability.

The ablation also shows that the two mechanisms play different roles. Re-optimization provides the larger individual improvement: by releasing assigned but unstarted requests, it allows the planner to repair schedules that become undesirable after new information arrives. Prediction reweighting alone provides smaller benefits and, under high demand, performs similarly to the non-adaptive rollout variant. This is expected because reweighting changes the influence of future predicted requests prospectively, but it cannot by itself revise assignments that have already been made.

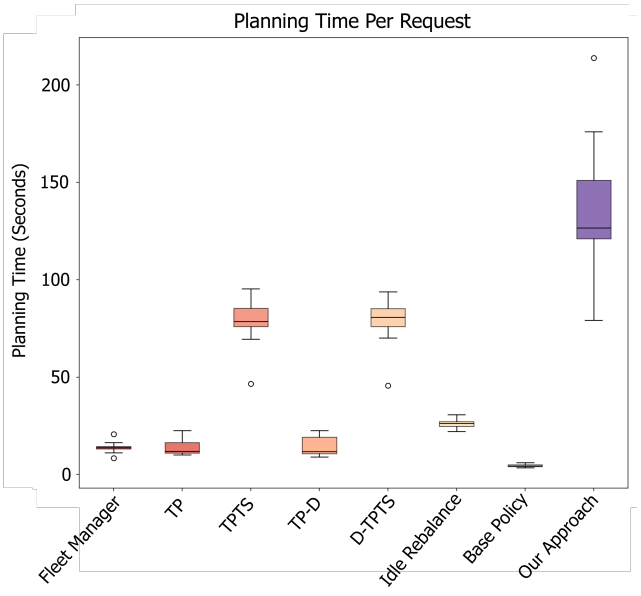

Figure 13: Planning time per request-triggered decision across the evaluated assignment policies. Each box plot summarizes the computation time required to process a decision during the evaluation runs. Lower values indicate faster online decision making. The proposed adaptive rollout policy requires more computation than myopic and token-passing baselines because it evaluates candidate assignments through sampled future scenarios, but its median planning time remains within the operational lead time of the case-study setting. Abbreviated policy labels correspond to the assignment policies defined in Section 5.5.1.

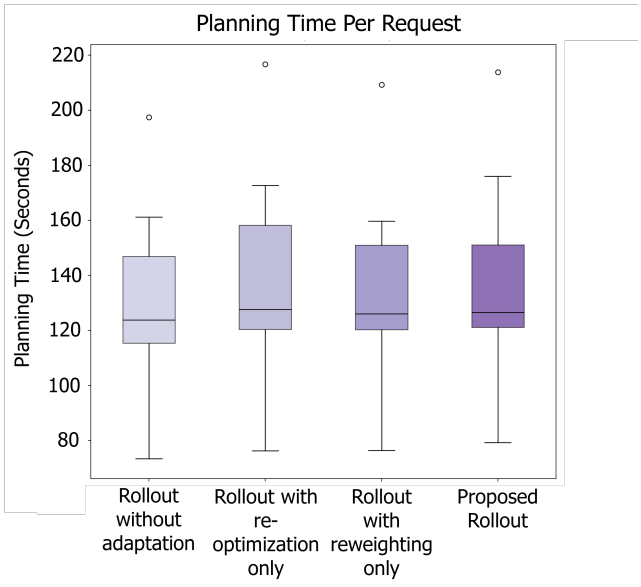

Figure 14: Planning time per request-triggered decision across the ablation policies. Each box plot shows the distribution of computation time required to process a decision during the evaluation runs. Lower values indicate faster online decision making.

When reweighting and re-optimization are combined, the policy obtains the best performance in all regimes, indicating that the two mechanisms are complementary: reweighting reduces the future influence of unreliable predictions, while re-optimization repairs eligible commitments made under earlier forecasts.

Figure 14 compares the planning-time distributions for the rollout ablation variants. The adaptive mechanisms add only modest overhead relative to rollout without adaptation, because the dominant computational cost is evaluating candidate actions over sampled future scenarios. Prediction reweighting requires only lightweight updates to prediction-confidence weights, and re-optimization changes the set of eligible pending requests only when the trigger condition is satisfied.

The runtime results also show that the full adaptive rollout policy does not incur the largest tail planning times among the ablation variants. In particular, combining reweighting with re-optimization can reduce unnecessary schedule repair by lowering the influence of unreliable predictions before they lead to poor assignments. Thus, the full method improves wait-time performance without substantially increasing the computational cost of the underlying rollout procedure.

6. Conclusion

This paper studied online heterogeneous multi-robot task assignment and scheduling with scheduled requests, real-time requests, service-window constraints, and uncertain future demand. We formulated the problem as a finite-horizon stochastic dynamic program in which controls update the assignments and schedules of known requests, while future real-time requests enter through a stochastic disturbance process. The formulation explicitly distinguishes between the true deployment distribution $\mathbb{P}$ and the learned predictive distribution $\widehat{\mathbb{P}}$, which makes it possible to reason about prediction error and distribution shift within the online decision-making problem.

To address this problem, we proposed a prediction-aware adaptive rollout framework. The key design choice is to separate immediate commitments from future-demand estimates: robots can only be assigned to requests that have actually entered the system, while predicted requests influence decisions through sampled rollout simulations. This allows the planner to account for the opportunity cost of current assignments without prematurely committing robots to requests that may never occur. The rollout policy is made computationally tractable using one-robot-at-a-time decision making, heuristic candidate pruning, wait actions, and an interaction-aware base policy for future-cost estimation.

The framework also includes adaptive mechanisms for robustness under prediction error. Prediction-confidence weights are updated online from recent forecast errors and used to reduce the contribution of unreliable predicted requests in the rollout objective. Selective re-optimization

complements this prospective correction by allowing assigned but unstarted requests to be reconsidered when new observations make the current schedule undesirable. Together, these mechanisms allow predictions to guide online allocation when they are useful, while limiting their influence when they become misleading.

The hospital-floor case study demonstrated how the framework can be instantiated in a realistic heterogeneous service setting. Historical request data were used both to train temporal point process models for future-request scenario generation and to select a fixed heterogeneous team composition before deployment. The empirical results showed that the selected team composition was sufficient to avoid persistent overload on the held-out floor-days, and that the proposed adaptive rollout policy reduced request wait times relative to reactive, token-passing, prediction-positioning, and myopic greedy baselines. The largest improvements appeared in tail wait-time metrics, indicating that prediction-aware rollout is especially useful for avoiding severe delays. The ablation study further showed that re-optimization and prediction reweighting play complementary roles: re-optimization repairs eligible commitments after new information arrives, while reweighting reduces the future influence of unreliable predictions.

Several directions remain for future work. First, the current implementation uses a finite set of sampled future scenarios and a heuristic base policy for downstream cost estimation. Future work could study stronger scenario-reduction methods, learned value approximations, or uncertainty-aware base policies that preserve rollout quality while reducing runtime. Second, the prediction-confidence mechanism currently relies on count-based forecast errors within predefined contexts; richer calibration methods could account for spatial structure, task dependencies, and correlations between local request processes. Third, the selective re-optimization rule could be extended to reason about the cost of schedule disruption more explicitly, especially in settings where changing unstarted assignments affects human workflows or resource availability. Finally, future deployments should evaluate the framework in closed-loop physical or high-fidelity hospital operations, where communication delays, navigation uncertainty, human interaction, and operational constraints may further affect the value of prediction-aware decision making.

Overall, the proposed framework provides a practical way to use learned future-demand predictions in online heterogeneous multi-robot scheduling. By restricting immediate actions to observed requests while adaptively weighting predicted requests in the lookahead objective, the method exploits useful forecasts without allowing unreliable predictions to dominate online decisions.

7. Acknowledgments

Authors are grateful for the Amazon Gift that funded this research in part. This work was also supported in part by the Defense Advanced Research Projects Agency (DARPA) under Grant No. $D24AP00319-01$. The views and conclusions expressed in this paper are those of the authors and do not reflect the official policy or position of the U.S. Army, U.S. Department of War, or U.S. Government.

CRediT authorship contribution statement

Daniel Garces: Conceptualization, Formal Analysis, Investigation, Methodology, Software, Validation, Visualization, Writing - original draft, Writing - review and editing. **Sara Castro:** Conceptualization, Data curation, Methodology, Writing - review and editing. **Adrian Haimovich:** Data curation, Supervision. **Byron Crowe:** Conceptualization, Data curation, Supervision. **Stephanie Gil:** Funding Acquisition, Supervision, Writing - review and editing.

References

[1] Akella, P., Dixit, A., Ahmadi, M., Lindemann, L., Chapman, M.P., Pappas, G.J., Ames, A.D., Burdick, J.W., 2025. Risk-aware robotics: Tail risk measures in planning, control, and verification [focus on education]. IEEE Control Systems 45, 46–78. doi:10.1109/MCS.2025.3577050.

[2] Arribas, E., Cholvi, V., Mancuso, V., 2023. Optimizing uav resupply scheduling for heterogeneous and persistent aerial service. IEEE Transactions on Robotics 39, 2639–2653. doi:10.1109/TRO.2023.3263077.

[3] Asadi, K., Misra, D., Kim, S., Littman, M.L., 2019. Combating the compounding-error problem with a multi-step model. CoRR abs/1905.13320. URL: http://arxiv.org/abs/1905.13320, arXiv:1905.13320.

[4] Aswale, A., Pinciroli, C., 2023. Heterogeneous coalition formation and scheduling with multi-skilled robots, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5402–5409. doi:10.1109/IROS55552.2023.10342489.

[5] Augusto Zagatti, G., Kiong Ng, S., Bressan, S., 2024. Learning multivariate temporal point processes via the time-change theorem, in: Dasgupta, S., Mandt, S., Li, Y. (Eds.), Proceedings of The 27th International Conference on Artificial Intelligence and Statistics, PMLR. pp. 3241–3249. URL: https://proceedings.mlr.press/v238/augusto-zagatti24a.html.

[6] Aziz, H., Chan, H., Cseh, A., Li, B., Ramezani, F., Wang, C., 2021. Multi-robot task allocation-complexity and approximation, in: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC. p. 133–141.

[7] Bai, X., Li, C., Zhang, B., Wu, Z., Ge, S.S., 2024a. Efficient performance impact algorithms for multirobot task assignment with deadlines. IEEE Transactions on Industrial Electronics 71, 14373–14382. doi:10.1109/TIE.2024.3374366.

[8] Bai, Y., Lindqvist, B., Nordström, S., Kanellakis, C., Nikolakopoulos, G., 2024b. Cluster-based multi-robot task assignment, planning, and control. International Journal of Control, Automation and Systems 22, 2537–2550. doi:10.1007/s12555-023-0745-4.

[9] Banfi, J., Messing, A., Kroninger, C., Stump, E., Hutchinson, S., Roy, N., 2022. Hierarchical planning for heterogeneous multi-robot routing problems via learned subteam performance. IEEE Robotics and Automation Letters 7, 4464–4471. doi:10.1109/LRA.2022.3148489.

[10] Berbeglia, G., Cordeau, J.F., Laporte, G., 2010. Dynamic pickup and delivery problems. European Journal of Operational Research 202, 8–15. URL: https://www.sciencedirect.com/science/article/pii/S0377221709002999, doi:https://doi.org/10.1016/j.ejor.2009.04.024.

[11] Bertsekas, D., 2019. Reinforcement Learning and Optimal Control. Athena Scientific optimization and computation series, Athena Scientific, Nashua, NH, USA. URL: https://books.google.com/books?id=2f85EAAAQBAJ.

[12] Bertsekas, D., 2020. Rollout, Policy Iteration, and Distributed Reinforcement Learning. Athena scientific optimization and computation series, Athena Scientific., Nashua, NH, USA. URL: https://books.google.com/books?id=Hbo-EAAAQBAJ.

[13] Binny, A.E., Dixit, A., 2025. Who moved my distribution? conformal prediction for interactive multi-agent systems. ArXiv abs/2511.11567. URL: https://api.semanticscholar.org/CorpusID:283055266.

[14] Bischoff, E., Kohn, S., Hahn, D., Braun, C., Rothfuß, S., Hohmann, S., 2024. Heuristic reoptimization of time-extended multi-robot task allocation problems. Networks 84, 64–83. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.22217, doi:https://doi.org/10.1002/net.22217, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.22217.

[15] Bopardikar, S.D., Smith, S.L., Bullo, F., 2014. On dynamic vehicle routing with time constraints. IEEE Transactions on Robotics 30, 1524–1532. doi:10.1109/TRO.2014.2344451.

[16] Calvo, A., Capitan, J., 2024. Heterogeneous multi-robot task allocation for long-endurance missions in dynamic scenarios. URL: https://arxiv.org/abs/2411.02062, arXiv:2411.02062.

[17] Camisa, A., Testa, A., Notarstefano, G., 2023. Multi-robot pickup and delivery via distributed resource allocation. IEEE Transactions on Robotics 39, 1106–1118. doi:10.1109/TRO.2022.3216801.

[18] Cao, H., Meng, Z., Ke, T., Zhou, F., 2024. Is score matching suitable for estimating point processes?, in: The Thirty-eighth Annual Conference on Neural Information Processing Systems. URL: https://openreview.net/forum?id=HQgHCVZiHw.

[19] Chang, Y., Boyd, A.J., Xiao, C., Kass-Hout, T., Bhatia, P., Smyth, P., Warrington, A., 2026. Deep continuous-time state-space models for marked event sequences, in: The Thirty-ninth Annual Conference on Neural Information Processing Systems. URL: https://openreview.net/forum?id=74SvE2GZwW.

[20] Chang, Y., J Boyd, A., Smyth, P., 2024. Probabilistic modeling for sequences of sets in continuous-time, in: Dasgupta, S., Mandt, S., Li, Y. (Eds.), Proceedings of The 27th International Conference on Artificial Intelligence and Statistics, PMLR. pp. 4357–4365. URL: https://proceedings.mlr.press/v238/chang24a.html.

[21] Chaouach, L.M., Boskos, D., Oomen, T., 2022. Uncertain uncertainty in data-driven stochastic optimization: towards structured ambiguity sets, in: 2022 IEEE 61st Conference on Decision and Control (CDC), pp. 4776–4781. doi:10.1109/CDC51059.2022.9992405.

[22] Chee, K.Y., Silva, T.C., Hsieh, M.A., Pappas, G.J., 2024. Uncertainty quantification and robustification of model-based controllers using conformal prediction, in: Abate, A., Cannon, M., Margellos, K., Papachristodoulou, A. (Eds.), Proceedings of the 6th Annual Learning for Dynamics; Control Conference, PMLR. pp. 528–540. URL: https://proceedings.mlr.press/v242/chee24a.html.

[23] Chen, Y., Arkin, J., Zhang, Y., Roy, N., Fan, C., 2024. Scalable multi-robot collaboration with large language models: Centralized or decentralized systems?, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), pp. 4311–4317. doi:10.1109/ICRA57147.2024.10610676.

[24] Chen, Z., Alonso-Mora, J., Bai, X., Harabor, D.D., Stuckey, P.J., 2021. Integrated task assignment and path planning for capacitated multi-agent pickup and delivery. IEEE Robotics and Automation Letters 6, 5816–5823. doi:10.1109/LRA.2021.3074883.

[25] Dai, W., Rai, U., Chiun, J., Yuhong, C., Sartoretti, G., 2025. Heterogeneous multi-robot task allocation and scheduling via reinforcement learning. IEEE Robotics and Automation Letters , 1–8doi:10.1109/LRA.2025.3534682.

[26] Das, G., McGinnity, T., Coleman, S., Behera, L., 2015. A distributed task allocation algorithm for a multi-robot system in healthcare facilities. Journal of Intelligent & Robotic Systems 80. doi:10.1007/s10846-014-0154-2.

[27] Dhanaraj, N., Kang, J.H., Mukherjee, A., Nemlekar, H., Nikolaidis, S., Gupta, S.K., 2024. Multi-robot task allocation under uncertainty via hindsight optimization, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), pp. 16574–16580. doi:10.1109/ICRA57147.2024.10611370.

[28] Draxler, F., Meng, Y., Nelson, K., Laskowski, L., Yang, Y., Karaletsos, T., Mandt, S., 2026. Transformers for mixed-type event sequences, in: The Thirty-ninth Annual Conference on Neural Information Processing Systems. URL: https://openreview.net/forum?id=MtwsRjPZhf.

[29] Du, N., Dai, H., Trivedi, R., Upadhyay, U., Gomez-Rodriguez, M., Song, L., 2016. Recurrent marked temporal point processes: Embedding event history to vector, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, NY, USA. p. 1555–1564. URL: https://doi.org/10.1145/2939672.2939875, doi:10.1145/2939672.2939875.

[30] Fan, H., Li, D., Ouyang, B., Yan, Z., Wang, Y., 2023. Improving scheduling in multi-agv systems by task prediction. Journal of Scheduling , 1–10URL: https://api.semanticscholar.org/CorpusID:265207678.

[31] Ferreira, B.A., Petrović, T., Bogdan, S., 2022. Distributed mission planning of complex tasks for heterogeneous multi-robot systems, in: 2022 IEEE 18th International Conference on Automation Science and Engineering (CASE), pp. 1224–1231. doi:10.1109/CASE49997.2022.9926542.

[32] Fu, B., Smith, W., Rizzo, D.M., Castanier, M., Ghaffari, M., Barton, K., 2023a. Robust task scheduling for heterogeneous robot teams under capability uncertainty. IEEE Transactions on Robotics 39, 1087–1105. doi:10.1109/TRO.2022.3216068.

[33] Fu, B., Smith, W., Rizzo, D.M., Castanier, M., Ghaffari, M., Barton, K., 2023b. Robust task scheduling for heterogeneous robot teams under capability uncertainty. IEEE Transactions on Robotics 39, 1087–1105. doi:10.1109/TRO.2022.3216068.

[34] Gao, P., Siva, S., Micciche, A., Zhang, H., 2023. Collaborative scheduling with adaptation to failure for heterogeneous robot teams, in: 2023 IEEE International Conference on Robotics and Automation (ICRA), pp. 1414–1420. doi:10.1109/ICRA48891.2023.10161502.

[35] Gibbs, I., Candès, E., 2024. Conformal inference for online prediction with arbitrary distribution shifts. J. Mach. Learn. Res. 25.

[36] Gibbs, I., Candès, E.J., 2021. Adaptive conformal inference under distribution shift, in: Proceedings of the 35th International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA.

[37] Gosrich, W., Mayya, S., Narayan, S.R., Malencia, M., Agarwal, S., Kumar, V.R., 2022. Multi-robot coordination and cooperation with task precedence relationships. 2023 IEEE International Conference on Robotics and Automation (ICRA) , 5800–5806URL: https://api.semanticscholar.org/CorpusID:252595835.

[38] Gupta, P., Isele, D., Sachdeva, E., Huang, P.H., Dariush, B., Lee, K., Bae, S., 2025. Generalized mission planning for heterogeneous multi-robot teams via llm-constructed hierarchical trees. URL: https://arxiv.org/abs/2501.16539, arXiv:2501.16539.

[39] Ham, A., 2021. Transfer-robot task scheduling in job shop. International Journal of Production Research 59, 813–823. URL: https://doi.org/10.1080/00207543.2019.1709671, doi:10.1080/00207543.2019.1709671.

[40] Janner, M., Fu, J., Zhang, M., Levine, S., 2019. When to trust your model: model-based policy optimization. Curran Associates Inc., Red Hook, NY, USA.

[41] Ji, R., Lejeune, M., Fan, Z., 2019. Distributionally robust portfolio optimization with starr performance measure. SSRN Electronic Journal doi:10.2139/ssrn.3542667.

[42] Ji, T., Luo, Y., Sun, F., Jing, M., He, F., Huang, W., 2022. When to update your model: Constrained model-based reinforcement learning, in: Oh, A.H., Agarwal, A., Belgrave, D., Cho, K. (Eds.), Advances in Neural Information Processing Systems. URL: https://openreview.net/forum?id=9a1oV7UunyP.

[43] Jiang, N., Kulesza, A., Singh, S., Lewis, R., 2015. The dependence of effective planning horizon on model accuracy, in: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC. p. 1181–1189.

[44] Jokinen, J.P., Sihvola, T., Hyytiä, E., Sulonen, R., 2011. Why urban mass demand responsive transport?, in: 2011 IEEE Forum on Integrated and Sustainable Transportation Systems, pp. 317–322. doi:10.1109/FISTS.2011.5973631.

[45] Kalempa, V.C., Piardi, L., Limeira, M., de Oliveira, A.S., 2021. Multi-robot preemptive task scheduling with fault recovery: A novel approach to automatic logistics of smart factories. Sensors 21. URL: https://www.mdpi.com/1424-8220/21/19/6536, doi:10.3390/s21196536.

[46] Kannan, S.S., Venkatesh, V.L.N., Min, B.C., 2024. Smart-llm: Smart multi-agent robot task planning using large language models, in: 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 12140–12147. doi:10.1109/IROS58592.2024.10802322.

[47] Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. CoRR abs/1412.6980. URL: https://api.semanticscholar.org/CorpusID:6628106.

[48] Kondor, D., Bojic, I., Resta, G., Duarte, F., Santi, P., Ratti, C., 2022. The cost of non-coordination in urban on-demand mobility. Scientific Reports 12. doi:10.1038/s41598-022-08427-2.

[49] Laney, D.B., 2002. Improved control charts for attributes. Quality Engineering 14, 531–537. doi:10.1081/QEN-120003555.

[50] Lee, S., Shahzaad, B., Alkouz, B., Lakhdari, A., Bouguettaya, A., 2023. Autonomous delivery of multiple packages using single drone in urban airspace, in: Adjunct Proceedings of the 2022 ACM International Joint Conference on Pervasive and Ubiquitous Computing and the 2022 ACM International Symposium on Wearable Computers, Association for Computing Machinery, New York, NY, USA. p. 72–74. URL: https://doi.org/10.1145/3544793.3560330, doi:10.1145/3544793.3560330.

[51] Lee, Y., Williams, J., Marklund, H., Sharma, A., Mitchell, E., Singh, A., Finn, C., 2025. Inference-time alignment via hypothesis reweighting, in: Second Workshop on Test-Time Adaptation: Putting Updates to the Test! at ICML 2025. URL: https://openreview.net/forum?id=tl2nXqQSQJ.

[52] Li, Y., Jiao, X.Y., Sun, B.Q., Zhang, Q.H., Yang, J.Y., 2021. Multi-welfare-robot cooperation framework for multi-task assignment in healthcare facilities based on multi-agent system, in: 2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR), pp. 413–416. doi:10.1109/ISR50024.2021.9419496.

[53] Lin, V., Kaur, R., Yang, Y., Dutta, S., Kantaros, Y., Roy, A., Jha, S., Sokolsky, O., Lee, I., 2025. Safety monitoring for learning-enabled cyber-physical systems in out-of-distribution scenarios, in: Proceedings of the ACM/IEEE 16th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2025), Association for Computing Machinery, New York, NY, USA. URL: https://doi.org/10.1145/3716550.3722022, doi:10.1145/3716550.3722022.

[54] Lindemann, L., Cleaveland, M., Shim, G., Pappas, G.J., 2023. Safe planning in dynamic environments using conformal prediction. IEEE Robotics and Automation Letters 8, 5116–5123. doi:10.1109/LRA.2023.3292071.

[55] Lindemann, L., Zhao, Y., Yu, X., Pappas, G.J., Deshmukh, J.V., 2025. Formal verification and control with conformal prediction: Practical safety guarantees for autonomous systems. IEEE Control Systems 45, 72–122. doi:10.1109/MCS.2025.3611545.

[56] Liu, K., Tang, Z., Wang, D., Wang, Z., Zhao, B., Li, X., 2024. Coherent: Collaboration of heterogeneous multi-robot system with large language models. URL: https://arxiv.org/abs/2409.15146, arXiv:2409.15146.

[57] Liu, R., Gao, P., Shen, Y., Lin, M., Tokekar, P., 2026. Adaptive conformal guidance for learning under uncertainty, in: The Fourteenth International Conference on Learning Representations. URL: https://openreview.net/forum?id=1gxP0WtOoO.

[58] Lowalekar, M., Varakantham, P., Jaillet, P., 2018. Online spatio-temporal matching in stochastic and dynamic domains. Artificial Intelligence 261, 71–112. URL: https://www.sciencedirect.com/science/article/pii/S0004370218302030, doi:https://doi.org/10.1016/j.artint.2018.04.005.

[59] Luo, R., Zhao, S., Kuck, J., Ivanovic, B., Savarese, S., Schmerling, E., Pavone, M., 2024. Sample-efficient safety assurances using conformal prediction. The International Journal of Robotics Research 43, 1409–1424.

[60] Lyu, Y., Chow, C.Y., Lee, V.C., Ng, J.K., Li, Y., Zeng, J., 2019. Cb-planner: A bus line planning framework for customized bus systems. Transportation Research Part C: Emerging Technologies 101, 233–253. URL: https://www.sciencedirect.com/science/article/pii/S0968090X18305126, doi:https://doi.org/10.1016/j.trc.2019.02.006.

[61] Ma, H., Hönig, W., Kumar, T.K.S., Ayanian, N., Koenig, S., 2019. Lifelong path planning with kinematic constraints for multi-agent pickup and delivery, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 7651–7658. doi:10.1609/aaai.v33i01.33017651.

[62] Ma, H., Li, J., Kumar, T.S., Koenig, S., 2017. Lifelong multi-agent path finding for online pickup and delivery tasks, in: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC. p. 837–845.

[63] Makino, H., Ito, S., 2024. Online multi-agent pickup and delivery with task deadlines, in: 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 8428–8434. doi:10.1109/IROS58592.2024.10802597.

[64] Mathew, N., Smith, S.L., Waslander, S.L., 2015. Planning paths for package delivery in heterogeneous multirobot teams. IEEE Transactions on Automation Science and Engineering 12, 1298–1308. doi:10.1109/TASE.2015.2461213.

[65] McAllister, R.D., Rawlings, J.B., 2024. On the inherent distributional robustness of stochastic and nominal model predictive control. IEEE Transactions on Automatic Control 69, 741–754. doi:10.1109/TAC.2023.3273420.

[66] Mei, H., Eisner, J., 2017. The neural Hawkes process: A neurally self-modulating multivariate point process, in: Advances in Neural Information Processing Systems, Long Beach. URL: https://arxiv.org/abs/1612.09328.

[67] Mei, H., Yang, C., Eisner, J., 2022. Transformer embeddings of irregularly spaced events and their participants, in: International Conference on Learning Representations. URL: https://openreview.net/forum?id=Rty5g9imm7H.

[68] Miao, F., He, S., Pepin, L., Han, S., Hendawi, A., Khalefa, M.E., Stankovic, J.A., Pappas, G., 2021. Data-driven distributionally robust optimization for vehicle balancing of mobility-on-demand systems. ACM Trans. Cyber-Phys. Syst. 5. URL: https://doi.org/10.1145/3418287, doi:10.1145/3418287.

[69] Miloradović, B., Çürüklü, B., Ekström, M., Papadopoulos, A.V., 2023. Optimizing parallel task execution for multi-agent mission planning. IEEE Access 11, 24367–24381. doi:10.1109/ACCESS.2023.3254900.

[70] Mohajerin Esfahani, P., Kuhn, D., 2018. Data-driven distributionally robust optimization using the wasserstein metric: Performance guarantees and tractable reformulations. Mathematical Programming 171, 115–166.

[71] Montgomery, D.C., 2019. Introduction to Statistical Quality Control. 8 ed., John Wiley & Sons.

[72] National Library of Medicine, 2026. ATC Product Classes in RxClass: ATCPROD. https://lhncbc.nlm.nih.gov/RxNav/applications/RxClass-ATC-product-classes-ATCPROD.html. Accessed: 2026-01-26.

[73] Neville, G., Chernova, S., Ravichandar, H., 2023. D-itags: A dynamic interleaved approach to resilient task allocation, scheduling, and motion planning. IEEE Robotics and Automation Letters 8, 1037–1044. doi:10.1109/LRA.2023.3234824.

[74] Neville, G., Messing, A., chaandar Ravichandar, H., Hutchinson, S.A., Chernova, S., 2021. An interleaved approach to trait-based task allocation and scheduling. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) , 1507–1514URL: https://api.semanticscholar.org/CorpusID:236924745.

[75] Nishi, T., Ando, M., Konishi, M., 2005. Distributed route planning for multiple mobile robots using an augmented lagrangian decomposition and coordination technique. IEEE Transactions on Robotics 21, 1191–1200. doi:10.1109/TRO.2005.853489.

[76] Notomista, G., Mayya, S., Emam, Y., Kroninger, C., Bohannon, A., Hutchinson, S., Egerstedt, M., 2022. A resilient and energy-aware task allocation framework for heterogeneous multirobot systems. IEEE Transactions on Robotics 38, 159–179. doi:10.1109/TRO.2021.3102379.

[77] Omi, T., ueda, n., Aihara, K., 2019. Fully neural network based model for general temporal point processes, in: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/39e4973ba3321b80f37d9b55f63ed8b8-Paper.pdf.

[78] Ongel, A., Loewer, E., Roemer, F., Sethuraman, G., Chang, F., Lienkamp, M., 2019. Economic assessment of autonomous electric microtransit vehicles. Sustainability 11. URL: https://www.mdpi.com/2071-1050/11/3/648, doi:10.3390/su11030648.

[79] Pan, F., He, J., Tu, D., He, Q., 2020. Trust the model when it is confident: masked model-based actor-critic, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA.

[80] Park, B., Kang, C., Choi, J., 2022. Cooperative multi-robot task allocation with reinforcement learning. Applied Sciences 12. URL: https://www.mdpi.com/2076-3417/12/1/272, doi:10.3390/app12010272.

[81] Park, J., Messing, A., Ravichandar, H., Hutchinson, S., 2023. Risk-tolerant task allocation and scheduling in heterogeneous multi-robot teams, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5372–5379. doi:10.1109/IROS55552.2023.10341837.

[82] Phillips, M., Likhachev, M., 2011. SIPP: Safe interval path planning for dynamic environments, in: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 5628–5635. doi:10.1109/ICRA.2011.5980306.

[83] Popolizio, F., Vinetti, M., Combrink, A., Roselli, S.F., Pia Fanti, M., Fabian, M., 2024. Online conflict-free scheduling of fleets of autonomous mobile robots, in: 2024 IEEE 20th International Conference on Automation Science and Engineering (CASE), pp. 3063–3068. doi:10.1109/CASE59546.2024.10711693.

[84] Reed, S., Campbell, A.M., Thomas, B.W., 2022. The value of autonomous vehicles for last-mile deliveries in urban environments. Management Science 68, 280–299. URL: https://doi.org/10.1287/mnsc.2020.3917, doi:10.1287/mnsc.2020.3917, arXiv:https://doi.org/10.1287/mnsc.2020.3917.

[85] Robey, A., Chamon, L., Pappas, G.J., Hassani, H., 2022. Probabilistically robust learning: Balancing average and worst-case performance, in: International Conference on Machine Learning, PMLR. pp. 18667–18686.

[86] Shaheen, S., Cohen, A., 2019. Shared ride services in north america: definitions, impacts, and the future of pooling. Transport Reviews 39, 427–442. URL: https://www.sciencedirect.com/science/article/pii/S0144164722001301, doi:https://doi.org/10.1080/01441647.2018.1497728.

[87] Shchur, O., Biloš, M., Günnemann, S., 2020a. Intensity-free learning of temporal point processes, in: International Conference on Learning Representations. URL: https://openreview.net/forum?id=HygOjhEYDH.

[88] Shchur, O., Gao, N., Biloš, M., Günnemann, S., 2020b. Fast and flexible temporal point processes with triangular maps, in: Advances in Neural Information Processing Systems (NeurIPS).

[89] Shen, J., Lai, H., Liu, M., Zhao, H., Yu, Y., Zhang, W., 2023. Adaptation augmented model-based policy optimization. Journal of Machine Learning Research 24, 1–35. URL: http://jmlr.org/papers/v24/22-0606.html.

[90] Shida, Y., Jimbo, T., Odashima, T., Matsubara, T., 2024. Reinforcement learning of multi-robot task allocation for multi-object transportation with infeasible tasks. URL: https://arxiv.org/abs/2404.11817, arXiv:2404.11817.

[91] Silver, D., 2005. Cooperative pathfinding, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, pp. 117–122. doi:10.1609/aiide.v1i1.18726.

[92] Sorbelli, F.B., Carpin, S., Corò, F., Das, S.K., Navarra, A., Pinotti, C.M., 2022. Speeding up routing schedules on aisle graphs with single access. IEEE Transactions on Robotics 38, 433–447. doi:10.1109/TRO.2021.3082021.

[93] Street, C., Lacerda, B., Mühlig, M., Hawes, N., 2024. Right place, right time: Proactive multi-robot task allocation under spatiotemporal uncertainty. Journal of Artificial Intelligence Research 79, 137–171.

[94] Talvitie, E., 2017. Self-correcting models for model-based reinforcement learning, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI Press. p. 2597–2603.

[95] Testa, A., Notarstefano, G., 2022. Generalized assignment for multi-robot systems via distributed branch-and-price. IEEE Transactions on Robotics 38, 1990–2001. doi:10.1109/TRO.2021.3120046.

[96] Verma, A., Gautam, A., Singh Shekhawat, V., Mohan, S., 2024. Dta-hmr-tt: Dynamic task allocation for a heterogeneous team of mobile robots with task transfer. IEEE Access 12, 177050–177063. doi:10.1109/ACCESS.2024.3505947.

[97] Vincent, J.A., Feldman, A.O., Schwager, M., 2024. Guarantees on robot system performance using stochastic simulation rollouts. IEEE Transactions on Robotics 40, 3984–4002. doi:10.1109/TRO.2024.3444070.

[98] Wang, J., Lian, Z., Liu, C., Liu, K., 2023. Iterated clustering optimization of the split-delivery vehicle routing problem considering passenger walking distance. Transportation Research Interdisciplinary Perspectives 17, 100751. URL: https://www.sciencedirect.com/science/article/pii/S2590198222002123, doi:https://doi.org/10.1016/j.trip.2022.100751.

[99] Wang, Z., Liu, C., Gombolay, M., 2022. Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints. Auton. Robots 46, 249–268. URL: https://doi.org/10.1007/s10514-021-09997-2, doi:10.1007/s10514-021-09997-2.

[100] Wei, C., Ji, Z., Cai, B., 2020. Particle swarm optimization for cooperative multi-robot task allocation: A multi-objective approach. IEEE Robotics and Automation Letters 5, 2530–2537. doi:10.1109/LRA.2020.2972894.

[101] WHO Collaborating Centre for Drug Statistics Methodology, 2026. ATC Classification Index with DDDs, 2026. https://atcddd.fhi.no/atc_ddd_index_and_guidelines/atc_ddd_index/. Accessed: 2026-06-26.

[102] Xidias, E., Zacharia, P., Nearchou, A., 2022. Intelligent fleet management of autonomous vehicles for city logistics. Applied Intelligence 52. doi:10.1007/s10489-022-03535-y.

[103] Xue, S., Shi, X., Chu, Z., Wang, Y., Hao, H., Zhou, F., Jiang, C., Pan, C., Zhang, J.Y., Wen, Q., Zhou, J., Mei, H., 2024. Easytpp: Towards open benchmarking temporal point processes, in: International Conference on Learning Representations (ICLR). URL: https://arxiv.org/abs/2307.08097.

[104] Xue, S., Shi, X., Hao, H., Ma, L., Zhang, J., Wang, S., Wang, S., 2021. A graph regularized point process model for event propagation sequence, in: 2021 International Joint Conference on Neural Networks (IJCNN), pp. 1–7. doi:10.1109/IJCNN52387.2021.9533830.

[105] Yang, G., Li, J., Geng, Z., Luo, C., 2024. Quality-aware experience exploitation in model-based reinforcement learning, in: 2024 IEEE International Conference on Big Data (BigData), pp. 1161–1166. doi:10.1109/BigData62323.2024.10825395.

[106] Yang, I., 2021. Wasserstein distributionally robust stochastic control: A data-driven approach. IEEE Transactions on Automatic Control 66, 3863–3870. doi:10.1109/TAC.2020.3030884.

[107] Yang, T., Feng, P., Guo, Q., Zhang, J., Ning, J., Wang, X., Mao, Z., 2025. Autohma-llm: Efficient task coordination and execution in heterogeneous multi-agent systems using hybrid large language models. IEEE Transactions on Cognitive Communications and Networking , 1–1doi:10.1109/TCCN.2025.3528892.

[108] Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J., Levine, S., Finn, C., Ma, T., 2020. Mopo: model-based offline policy optimization, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA.

[109] Zhang, D., Dong, P., Peng, P., Dong, Y., 2025a. A graph reinforcement learning framework for real-time distributed multi-robot task allocation. Aerospace Systems , 1–12.

[110] Zhang, Q., Lipani, A., Kirnap, O., Yilmaz, E., 2020. Self-attentive Hawkes process, in: III, H.D., Singh, A. (Eds.), Proceedings of the 37th International Conference on Machine Learning, PMLR. pp. 11183–11193. URL: https://proceedings.mlr.press/v119/zhang20q.html.

[111] Zhang, Y., Zhang, Z., Lim, A., Sim, M., 2021. Robust data-driven vehicle routing with time windows. Operations Research 69, 469–485. URL: https://doi.org/10.1287/opre.2020.2043, doi:10.1287/opre.2020.2043, arXiv:https://doi.org/10.1287/opre.2020.2043.

[112] Zhang, Y., Zhou, P., Zhang, R., Lu, S., Chai, T., 2025b. Semi-supervised concept drift detection and adaptation based on conformal martingale framework. Journal of Process Control 147, 103374. URL: https://www.sciencedirect.com/science/article/pii/S0959152425000022, doi:https://doi.org/10.1016/j.jprocont.2025.103374.

[113] Zhao, D., Yang, C., Zhang, T., Yang, J., Hiroshi, Y., 2022. A task allocation approach of multi-heterogeneous robot system for elderly care. Machines 10. URL: https://www.mdpi.com/2075-1702/10/8/622, doi:10.3390/machines10080622.

[114] Zheng, C., Li, L., Xu, F., Sun, F., Ding, M., 2005. Evolutionary route planner for unmanned air vehicles. IEEE Transactions on Robotics 21, 609–620. doi:10.1109/TRO.2005.844684.

[115] Zuo, H., Cao, B., Zhao, Y., Shen, B., Zheng, W., Huang, Y., 2021. High-capacity ride-sharing via shortest path clustering on large road networks. J. Supercomput. 77, 4081–4106. URL: https://doi.org/10.1007/s11227-020-03424-6, doi:10.1007/s11227-020-03424-6.

[116] Zuo, S., Jiang, H., Li, Z., Zhao, T., Zha, H., 2020. Transformer hawkes process, in: Proceedings of the 37th International Conference on Machine Learning, JMLR.org.

Daniel Garces is a Computer Science Ph.D. student in the School of Engineering and Applied Sciences at Harvard University, advised by Prof. Stephanie Gil. His research focuses on the development of model-based multi-agent reinforcement learning algorithms for task allocation in real world applications. He is interested in adaptation mechanisms and task allocation problems under uncertainty. He received his Bachelor's degree in Computer Engineering from Columbia University in 2021.

Sara Castro, MD, MPH, is an OB/GYN resident in the Harvard Mass General Brigham program. Her work focuses on patient perspectives and preferences regarding the use of robotics and emerging technologies in healthcare, with particular interest in equity, communication, care navigation, and language access. She earned her MPH in Healthcare Management, with a concentration in Public Health Leadership, from the Harvard T.H. Chan School of Public Health and her MD from Harvard Medical School. She also holds a BA in Medicine, Literature, and Society from Columbia University.

Dr. Adrian Haimovich is an Assistant Professor of Emergency Medicine at Harvard Medical School and Director of the Division of Artificial Intelligence in the Department of Emergency Medicine at Beth Israel Deaconess Medical Center in Boston, Massachusetts. His research is at the intersection between emergency medicine and healthcare AI.

Dr. Byron Crowe is Chief Medical Officer at Doctronic, where he leads clinical strategy and development of its AI-native care model. He is also a Clinical Assistant Professor of Medicine at Stanford, focused on improving complex systems at scale. Previously, he served as CMO at Solera Health and on the faculty of Harvard Medical School, where he studied AI in clinical reasoning and health policy. Named to Modern Healthcare's 2024 "40 Under 40," he earned his MD from Emory and a master's in clinical informatics management from Stanford, and is board-certified in internal medicine and clinical informatics.

Stephanie Gil is an Assistant Professor in the Computer Science Department at Harvard University's School of Engineering and Applied Sciences, where she directs the Robotics, Embedded Autonomy, and Communication Theory (REACT) Lab. Previously, she served as an Assistant Professor at Arizona State University. Her research explores multi-robot systems, focusing on how communication and information exchange impact resilience and trusted coordination. She has received several recognitions, including the NSF CAREER Award (2019), ONR Young Investigator Program Award, DARPA Young Faculty Award, and was named 2020 Alfred P. Sloan Fellow. Stephanie earned her PhD from the Massachusetts Institute of Technology in 2014.