Multi-Robot Allocation and Optimization in a Multi-Mission Framework ⋆

Branko Miloradović * **Mirgita Frasheri** **

* *Department of Intelligent Future Technologies, Mälardalen University, Sweden. (e-mail: branko.miloradovic@mdu.se)*
** *DIGIT, Department of Electrical and Computer Engineering, Aarhus University, Denmark. (e-mail: mirgita.frasheri@ece.au.dk)*

Abstract: This paper presents a framework for multi-mission multi-robot task allocation that integrates continuous-time routing with a lightweight bucketed reservation layer. Rather than collapsing all objectives into a single global mission, the framework keeps missions distinct and enables controlled sharing of robots across stakeholders with differing priorities and limited information exchange. The reservation layer overlays coarse time buckets on the planning horizon, allowing planners to specify time-phased mission quotas and enforce one-mission-per-robot commitments within each interval, all while preserving continuous task timing. This structure provides an operational control interface through which operators can adjust mission priorities over time without disclosing internal task details, enabling responsive, interpretable, and privacy-aware coordination. The results show that the proposed framework delivers feasible, continuous-time schedules that respect mission-level policies and achieve coordinated mission progress.

Keywords: Multi-Robot Missions, Multi-Mission Planning, MRTA

1. INTRODUCTION

In real-world multi-robot operations, tasks rarely form a single monolithic mission. Instead, they arise as multiple distinct missions, often driven by different stakeholders, operational needs, or events over time. Each mission may have its own objectives, priorities, deadlines, and constraints. For example, one mission may involve surveillance in a high-risk area under strict communication limits, while another may focus on time-critical search and rescue (see Fig. 1). Treating such missions as one unified problem would require combining all objectives and constraints into a centralized framework, which is often computationally impractical and organizationally unrealistic.

Moreover, missions do not usually appear all at once. In practice, they arise at different times and evolve with changing operational demands. Modeling missions separately allows robots to be reallocated as new missions emerge, without requiring complete foresight. This is essential for maintaining flexibility and responsiveness when resources are limited.

Security and confidentiality also matter. Missions may remain separate because stakeholders cannot—or do not wish to—fully share their objectives, data, or plans. For instance, civil authorities assessing critical infrastructure may keep vulnerability information classified, while humanitarian organizations may need to protect the privacy of rescue operations. Even under such restrictions, robots can still be shared across missions, making cross-mission optimization important for efficient use of scarce resources.

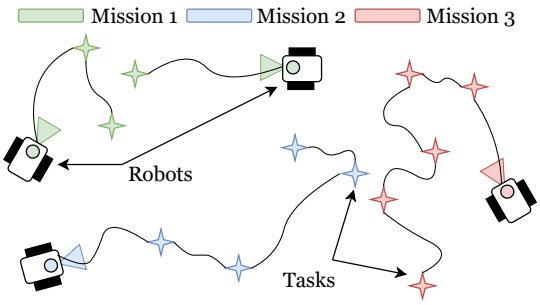

Fig. 1. An Illustration of a Multi-Mission.

This situation arises in many operational contexts, such as: (i) Defense operations: A military surveillance mission and a civilian evacuation mission may operate in the same area. Robots can be shared, but operational details such as objectives, reconnaissance targets, or troop positions must remain compartmentalized. (ii) Industrial collaboration: Competing companies deploying robots in offshore oil fields may conduct exploration and maintenance missions in overlapping regions. Because of commercial sensitivity, mission details cannot be shared, even though pooling robotic assets could reduce costs. (iii) Space exploration: Multiple space agencies may operate robotic rovers or drones on the same planetary surface. Scientific data and mission objectives may be restricted for intellectual property or national security reasons, but sharing resources such as communication relays or transport robots can improve mission success.

⋆ This work is funded by: The Knowledge Foundation (KKS), MARC project No. #20240011, and the Digital Twin Center for Open Research and Engineering (DT-CORE) project.

Real multi-robot operations depend not only on *what* tasks must be performed, but also on *when* a mission should receive the robots' available work effort. A purely continuous-time optimizer can yield efficient schedules, but it lacks simple levers for *time-phased*[1] *priority* and *inter-stakeholder coordination*. We therefore contribute with a lightweight *bucketed reservation layer*: a coarse time grid (e.g., 5–15 minute "buckets") overlaid on a continuous-time Multi-Robot Task Allocation (MRTA) model. Binary *reservation* variables (y) and *start-bucket* selectors (b) linearly encode per-bucket mission quotas and one-mission-per-robot-per-bucket policies, while indicator-timed constraints retain continuous start times. This design (i) gives operators explicit, explainable levers for temporal priorities; and (ii) supports partial-information coordination across stakeholders without exposing mission internals.

2. USE CASE

Consider a scenario in which a team of heterogeneous robots is deployed to locate trapped humans, assist in rescue operations, and assess the structural stability of a building after an earthquake. Floor plans are available to both the robots and the rescue teams; however, detailed information about the interior layout—such as the positions of chairs, desks, and shelves—is unknown.

To tackle such an emergency effectively, multiple missions must run in parallel and be coordinated over time. For example, the first mission focuses on **structural inspection**, where robots equipped with specialized sensors explore rooms and corridors to estimate the risk of collapse within a defined time window. This corresponds to the first time bucket, during which specific zones of the building are allocated to a subset of the available robots.

The framework is motivated by operational settings in which new task information may arrive over time, prompting periodic re-optimization of the remaining mission workload. If obstructions or debris slow down the inspection process, additional robots may need to be deployed to maintain progress. Rather than committing all robots at once, the system reserves a portion of them for later buckets. This approach allows for adaptive reinforcement once updated information becomes available, while preserving reserve capacity for later buckets when additional support may be needed.

Meanwhile, other missions can run concurrently. Once sections of the structure are deemed stable, a **search and rescue** mission is initiated, deploying robots equipped with passive infrared sensors to detect trapped humans. These robots operate in later buckets but can overlap temporally with ongoing inspection tasks in other parts of the building. As new information arrives from earlier buckets, reserved robots can be activated immediately and reassigned to assist the search and rescue mission, ensuring rapid and efficient resource reallocation.

This coordinated, bucket-based allocation enables proactive planning under uncertainty: robots are reserved for upcoming missions before the exact demand is known, allowing the system to react swiftly as real-time conditions evolve inside the disaster site. In addition, it allows stakeholders to coordinate on how much robot capacity should be reserved over time for inspection, rescue, or triage, while detailed mission content and mission-specific observations remain compartmentalized.

[1] "Time-phased" refers to requirements or priorities that are specified for particular portions of the timeline.

3. BACKGROUND AND MOTIVATION

Mission-aware planners typically rely on aggregated objectives (such as weighted sums of mission completion times) or add penalties for mission switches to steer the schedule toward desired mission priorities (Miloradović et al., 2022; Smith et al., 2007; Spaan et al., 2015). Such *soft* mechanisms influence averages but cannot guarantee that specific missions receive capacity during specified intervals. On the other extreme, fully time-indexed formulations (time-expanded networks, minute-level binaries) can encode per-interval capacities and fairness, but their size and numerical fragility restrict practical use in rolling-horizon settings (Desaulniers et al., 2005). Rolling-horizon replanning is widely used in dynamic domains (disaster response, intralogistics) to incorporate new information and reallocate resources, yet most works still lack an explicit, interpretable layer for stakeholder policies that evolve in time (Glomb et al., 2022; Cuisinier et al., 2021; Powell, 2007). The TAMER framework (Task Allocation in Multi-Robot Systems through an Entity-Relationship Model) introduced multi-mission structure as a distinct MRTA dimension, emphasizing that missions may differ in stakeholders, privacy requirements, and objectives and thus should not be collapsed into a single global problem (Miloradović et al., 2019). However, prior literature stops short of a tractable optimization construct that enforces *time-phased* service guarantees across missions while preserving continuous-time fidelity.

4. PROBLEM STATEMENT

The problem consists of allocating n tasks, grouped into p missions, to u robots with respect to given constraints. The planning horizon is partitioned into h buckets of fixed size. Each bucket defines per-mission reservation constraints that limit which robots can operate on which missions during that time interval. The formulation combines three classical optimization components. First, the routing layer determines robot paths from depots through assigned tasks and back to depots, which is structurally related to multi-vehicle routing. Second, the scheduling layer assigns continuous task start times and mission completion times subject to travel and service durations. Third, the bucketed reservation layer acts as a mission-capacity allocation mechanism by controlling how many robots are reserved for each mission in each time interval.

Let $\mathbb{R} = \{r_1, \ldots, r_u\}$ be the set of robots, $\mathbb{T} = \{t_1, \ldots, t_n\}$ the set of tasks, $\mathbb{M} = \{m_1, \ldots, m_p\}$ the set of missions, and $\mathbb{B} = \{b_1, \ldots, b_h\}$ the set of time buckets, where $n, u, p, h \in \mathbb{N}_{>0}$. Missions have distinct stakeholders/goal sets that run simultaneously. Missions don't share private internals, but they do share robots. Tasks are atomic service points without precedence constraints. Time is continuous and nonnegative. Each task $t \in \mathbb{T}$ belongs to exactly one mission through the mapping $\mu : \mathbb{T} \to \mathbb{M}$. For each mission

$m \in \mathbb{M}$, let $\mathbb{T}_m \doteq \{\, t \in \mathbb{T} \mid \mu(t) = m \,\}$. Each bucket $b_\ell \in \mathbb{B}$, represents the interval $[(\ell-1)\Delta, \ell\Delta)$, where $\Delta > 0$ is the bucket duration. Each mission m may require a minimum reserved robot capacity in each bucket b, represented by the quota parameter $q_{m,b}$.

The parameter $s_t \geq 0$ denotes the service time associated with task $t \in \mathbb{T}$. For each robot $r \in \mathbb{R}$ and task $t \in \mathbb{T}$, $c^{\mathrm{d}}_{r,t} \geq 0$ denotes the travel time from the depot of robot r to task t, and $c^{\mathrm{e}}_{r,t} \geq 0$ denotes the travel time from task t back to the depot of robot r. Moreover, $c^{\mathrm{tt}}_{t,t'} \geq 0$ represents the travel time between tasks $t, t' \in \mathbb{T}$. The binary parameter $\mathrm{compat}_{r,t} \in \{0,1\}$ indicates whether robot r is capable of executing task t, taking value 1 if compatible and 0 otherwise. Finally, $q_{m,b} \in \mathbb{Z}_{\geq 0}$ defines the minimum number of robots reserved for mission m in bucket $b \in \mathbb{B}$, while $\Delta > 0$ denotes the duration of each time bucket.

We have five binary decision variables and two continuous nonnegative variables. The binary decision variable $x^{\mathrm{d}}_{r,t} \in \{0,1\}$ indicates whether robot r starts from its depot and travels directly to task t. Likewise, $x^{\mathrm{e}}_{r,t} \in \{0,1\}$ indicates whether robot r terminates its route after task t. The binary variable $x_{r,t,t'} \in \{0,1\}$ takes value 1 if robot r travels directly from task t to task t', and 0 otherwise. Furthermore, $\beta_{r,t,b} \in \{0,1\}$ indicates whether task t, when assigned to robot r, both starts and finishes within bucket b. The binary variable $y_{r,b,m} \in \{0,1\}$ specifies whether robot r is reserved for mission m during bucket b. Finally, the continuous variable $\tau_{r,t} \geq 0$ represents the start time of task t on robot r, and $T_m \geq 0$ represents the completion time of mission m.

4.1 Constraints

Each robot $r \in \mathbb{R}$ can only execute tasks $t \in \mathbb{T}$ for which it is deemed compatible, as specified by a binary compatibility matrix $\mathrm{compat}_{r,t} \in \{0,1\}$. Incompatible assignments are fixed at zero by construction: $x^{\mathrm{d}}_{r,t} = 0, x^{\mathrm{e}}_{r,t} = 0, x_{r,t,t'} = 0$, i.e., $x_{r,t',t} = 0$ if $\mathrm{compat}_{r,t} = 0$ or $\mathrm{compat}_{r,t'} = 0$. This eliminates infeasible variables at model construction time, allowing the solver to disregard all incompatible robot–task pairs without introducing explicit linear constraints. Additionally, $x_{r,t,t} = 0$ for all r, t rules out self-loops, preventing a robot from "traveling" from a task back to itself.

$$\sum_{r\in\mathbb{R}} x^{\mathrm{d}}_{r,t} + \sum_{r\in\mathbb{R}} \sum_{t'\in\mathbb{T}} x_{r,t',t} = 1, \quad \forall t \in \mathbb{T}. \tag{1}$$

Eq. (1) ensures that each task t must be entered exactly once across all robots — either directly from a depot or from another task. The equality "$= 1$" ensures that each task t is entered exactly once by a single robot, preventing any task from being assigned to multiple robots or visited more than once.

$$\sum_{r\in\mathbb{R}} x^{\mathrm{e}}_{r,t} + \sum_{r\in\mathbb{R}} \sum_{t'\in\mathbb{T}} x_{r,t,t'} = 1, \quad \forall t \in \mathbb{T}. \tag{2}$$

Each task t must also be left exactly once across all robots, either leading to another task or to the depot at the end of a route. This is enforced by Eq. (2), i.e., the equality "$= 1$" enforces that every task has exactly one outgoing connection.

$$\sum_{t\in\mathbb{T}} x^{\mathrm{d}}_{r,t} = \sum_{t\in\mathbb{T}} x^{\mathrm{e}}_{r,t} \leq 1, \quad \forall r \in \mathbb{R}. \tag{3}$$

Each robot r has at most one start and one end (Eq. (3)). The total number of "departures" from a depot equals the total number of "returns" to a depot. This constraint also ensures that each robot executes at most one route; i.e., a robot may either remain idle (both sums equal to zero) or perform one continuous route from its depot to its endpoint (both sums equal to one).

$$x^{\mathrm{d}}_{r,t} + \sum_{t'\in\mathbb{T}} x_{r,t',t} = x^{\mathrm{e}}_{r,t} + \sum_{t'\in\mathbb{T}} x_{r,t,t'}, \forall r \in \mathbb{R}, t \in \mathbb{T}. \tag{4}$$

For every task t and robot r, the number of arcs entering t equals the number of arcs leaving it. Eq (4) is ensuring flow conservation. This guarantees that if a robot enters a task, it must also leave it (unless it is the last task), and if it leaves a task, it must have entered it. As a result, each task has balanced inflow and outflow, preventing dangling connections or disconnected task chains.

$$\tau_{r,t} \geq c^{\mathrm{d}}_{r,t} - M\big(1 - x^{\mathrm{d}}_{r,t}\big), \forall r \in \mathbb{R},\ t \in \mathbb{T}, \tag{5}$$

$$\tau_{r,t'} \geq \tau_{r,t} + s_t + c^{\mathrm{tt}}_{t,t'} - M\big(1 - x_{r,t,t'}\big), \quad \forall r \in \mathbb{R},\ t, t' \in \mathbb{T},\ t \neq t', \tag{6}$$

$$T_m \geq \tau_{r,t} + s_t - M(2 - x^{\mathrm{d}}_{r,t} - \sum_{t'\in\mathbb{T}} x_{r,t',t}), \quad \forall r \in \mathbb{R},\ t \in \mathbb{T}. \tag{7}$$

These constraints ensure that task start times, travel durations, and mission completion times are temporally consistent with the routing decisions. More specifically, Eq. (5) ensures that if robot r starts its route at task t, then the start time of t is no earlier than the travel time from its depot. Eq. (6) enforces temporal ordering between consecutive tasks: if robot r travels directly from task t to task t', then t' must start only after completing service at t and traveling to it. Finally, Eq. (7) ensures that the mission completion time $\mathbb{T}_m$ is not smaller than the completion time of any visited task t belonging to that mission. The Big-M constant is chosen sufficiently large ($M \geq h\Delta$) to deactivate the constraint whenever the corresponding binary variables are zero. As in standard mixed-integer scheduling formulations, excessively large Big-M values may weaken the relaxation and degrade numerical conditioning.

$$\sum_{b\in\mathbb{B}} \beta_{r,t,b} = x^{\mathrm{d}}_{r,t} + \sum_{t'\in\mathbb{T}} x_{r,t',t}, \quad \forall r \in \mathbb{R},\ t \in \mathbb{T}. \tag{8}$$

Eq. (8) enforces consistency between routing and bucket assignment. The left-hand side indicates whether task t assigned to robot r is scheduled to start in any time bucket. The right-hand side, equals 1 if and only if task t is visited by robot r (either as the first task or following another). Thus, this constraint ensures that a task receives exactly one bucket assignment if it is part of the robot's route, and none otherwise. It couples the routing decisions to the temporal discretization of the schedule.

$$(\ell-1)\Delta - M(1 - \beta_{r,t,b}) \leq \tau_{r,t} \leq \ell\Delta - s_t + M(1 - \beta_{r,t,b}), \quad \forall r \in \mathbb{R},\ t \in \mathbb{T},\ \ell \in \mathbb{B}. \tag{9}$$

Eq. (9) ensures that if task t of robot r is assigned to bucket b (i.e., $\beta_{r,t,b} = 1$), then its start time $\tau_{r,t}$ must fall within the interval $\big[(\ell-1)\Delta,\ \ell\Delta - s_t\big]$. If $\beta_{r,t,b} = 0$, the inequalities are relaxed by the big-M term. Thus, each task fits entirely within the time window of its assigned bucket.

As a result, the choice of bucket duration Δ must be made in relation to the task service-time distribution. If some tasks are longer than the bucket length, the formulation may introduce idle time or render such tasks infeasible to assign within a single bucket. Extending the model to allow tasks to span multiple buckets is left for future work.

$$\sum_{m \in \mathbb{M}} y_{r,b,m} \leq 1, \quad \forall r \in \mathbb{R},\ b \in \mathbb{B}. \tag{10}$$

Eq. (10) prevents a robot from being reserved simultaneously for multiple missions in the same bucket, i.e., it ensures that each robot can be reserved for at most one mission in each time bucket.

$$\beta_{r,t,b} \leq y_{r,b,m}, \forall r \in \mathbb{R},\ t \in \mathbb{T},\ b \in \mathbb{B},\ m \in \mathbb{M}. \tag{11}$$

Eq. (11) enforces consistency between task start times and mission reservations. If task t (belonging to mission m) starts in bucket b by robot r (i.e., $\beta_{r,t,b} = 1$), then the robot must be reserved for that same mission during that bucket, enforced by $y_{r,b,m} = 1$. This constraint blocks situations where a robot initiates a task associated with one mission while its reservation indicates commitment to another mission in the same time interval.

$$\sum_{r \in \mathbb{R}} y_{r,b,m} \geq q_{m,b}, \quad \forall m \in \mathbb{M},\ b \in \mathbb{B}. \tag{12}$$

Eq. (12) enforces per-bucket mission quotas, ensuring that each mission m receives at least $q_{m,b}$ robots during bucket b. The variable $y_{r,b,m}$ indicates whether robot r is reserved for mission m in that bucket, so $\sum_{r \in \mathbb{R}} y_{r,b,m}$ counts the total number of robots assigned to the mission. To capture the overall finishing time of the system, we introduce an auxiliary variable $T_{\max}$ satisfying

$$T_{\max} \ \geq \ T_m, \qquad \forall m \in \mathbb{M}. \tag{13}$$

where T_m is the completion time of mission m. We then minimize the makespan:

$$\min\ T_{\max}. \tag{14}$$

This objective minimizes the latest mission completion time across all missions. This means that by minimizing the makespan we discourage solutions where some missions finish early while others are significantly delayed, a common outcome when objectives focus only on aggregate completion measures. Instead, the optimizer is encouraged to allocate robots in a way that reduces the worst-case mission duration, thereby providing a more uniform level of service across all stakeholders.

5. BUCKETS AS AN OPERATIONAL CONTROL LAYER

The proposed formulation adds a bucketed reservation layer to provide time-phased control of mission capacity, unlike a continuous-time baseline that only optimizes routes and start times under a global objective. While the baseline can decide which robots perform which tasks, it cannot guarantee when a mission receives capacity. As a result, it may allocate robots efficiently in aggregate while still violating time-phased service requirements.

Soft penalties in continuous time can encourage temporal balance, but they do not guarantee per-bucket mission quotas and cannot enforce Eq. (12). Objective weights may shift mission timing, but they provide no formal service guarantees over time. The bucket layer addresses this limitation by allowing planners to reserve capacity through $y_{r,b,m}$ without specifying exact tasks in advance. This makes it possible to express commitments such as reserving robots for a mission during a given time interval. Buckets therefore provide a simple and interpretable way to enforce time-phased allocations, support partial information, and give operators high-level control. If missions are static, the base continuous-time model may be sufficient; in dynamic multi-stakeholder settings, the bucket layer provides the missing operational control.

Time-phased priority that holds on the clock. Per-bucket quotas $q_{m,b}$ express policies such as "0–30 minutes: ≥ 2 robots on Infrastructure; 30–60: ≥ 2 on Medical", or "in bucket b_ℓ all robots on Mission 1". Such rules are brittle if encoded only via global objective weights, because changing mission weights does not guarantee that the desired number of robots will actually occupy the correct mission during the intended time windows. Bucket quotas enforce these requirements explicitly and make the policy robust to travel delays, long tasks, or competing mission demands.

Security and compartmentalization. The framework supports a compartmentalized coordination through a reduced mission-level interface rather than implementing a formal privacy-preserving mechanism. Each mission keeps internal details local (task semantics, fine-grained task structure, stakeholder-specific objectives, and mission-specific collected data [2]). Coordination occurs at the bucket level, where information can be shared on the common bucket structure, per-bucket mission quotas $q_{m,b}$, and, when applicable, reservation decisions $y_{r,b,m}$ indicating which robots must be available for which missions in which buckets.

Stability and anti-thrash. Because allocation decisions are bucketed, one can penalize or cap reservation changes and mission switches between adjacent buckets, a control that is predictable in purely continuous models.

6. RESULTS

We evaluate the framework on an illustrative scenario to highlight its modeling capabilities, temporal expressiveness, and operational interpretability. [3] Rather than benchmarking against a specific baseline, the results show how the bucketed reservation layer enables precise time-phased mission-level resource allocation, while the continuous-time routing component ensures task-level feasibility. In all experiments, CPLEX generated schedules that satisfy task–robot compatibility, travel and service-time constraints, and the one-mission-per-robot-per-bucket rule. Although formulated for a static instance, the model can be reused for replanning as new information arrives. At each replanning point, started tasks are removed, each robot's progress is captured through its current location and effective release time, new tasks are added, bucket quotas can be updated to reflect revised mission priorities, and the problem is re-solved over the remaining horizon.

[2] Note that, robots can collect different types of data during their execution that should only be shared with the appropriate stakeholders according to given privacy policies. Ensuring the latter is outside the scope of this paper.

[3] The implementation is available online at https://github.com/mdh-planner/Multi-Mission-Buckets

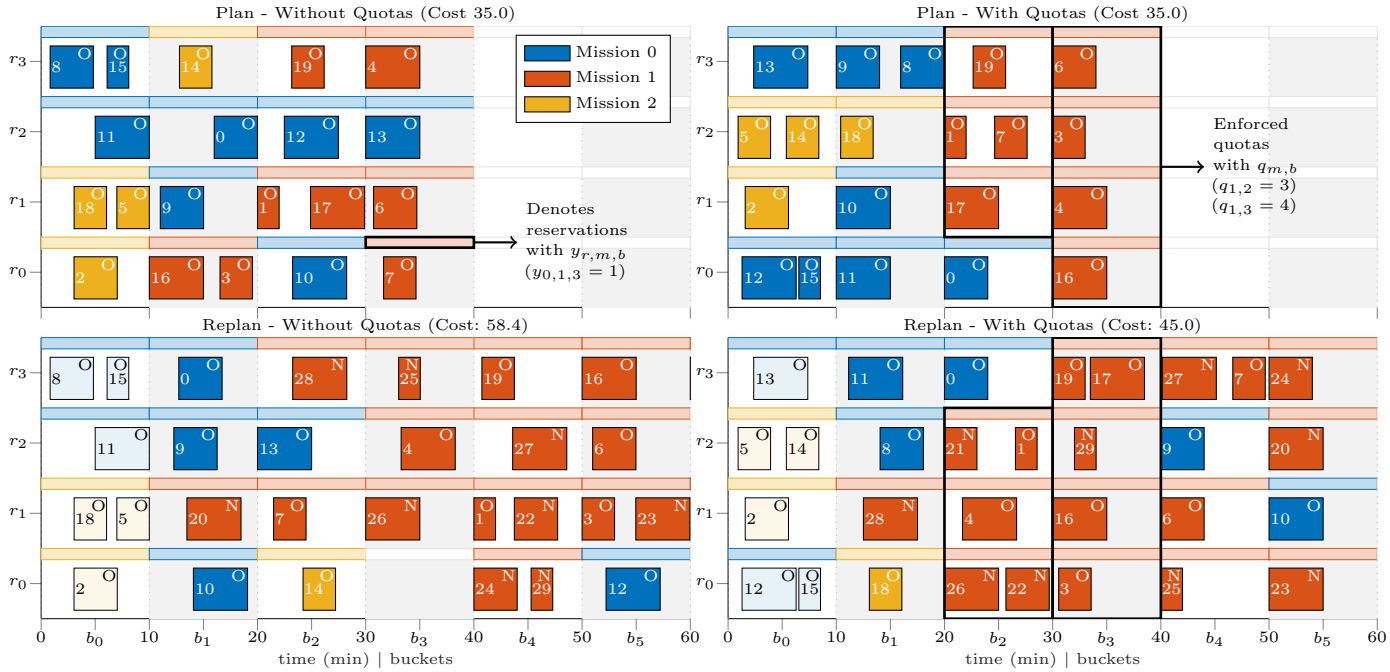

Fig. 2. Top row: initial plans for 20 (IDs from 0 to 19) tasks (labelled O). Bottom row: replanned schedules after the arrival of 10 (IDs from 19 to 29) new tasks (labelled N) assigned to Mission 1. Robots perform Infrastructure (blue), Triage (yellow), and Delivery/Evacuation (red) tasks across 10-minute buckets (denoted with b_0 to b_5). Lightly colored tasks in b_0, bottom row, have been completed.

The reservation variables $y_{r,b,m}$ give operators a direct way to enforce time-phased policies through bucket quotas, reallocating robots without requiring stakeholders to share task or mission details. This is especially useful in dynamic multi-stakeholder settings with separate objectives, privacy constraints, or distinct authorities. The framework is also flexible under updates. When tasks or priorities change, bucket quotas can be adjusted to reshape allocations in an immediate and interpretable way. In the illustrative scenario (Fig. 2), changing bucket assignments rebalances robots between missions while preserving continuous-time feasibility.

Overall, the main contribution is not improved solver performance, but structured, time-phased, and privacy-aware coordination across missions. The bucketed reservation layer enables high-level control of robot allocations, while the continuous-time model ensures that resulting plans remain executable.

Toy Example. Fig. 2 illustrates a post-storm city response with four robots (r_0–r_3) executing three missions: infrastructure inspection (M0, blue), delivery and evacuation support (M1, red), and medical triage (M2, yellow). Time is divided into 10-minute buckets over the first hour. The top row shows the initial plan (Plan 1). At 10 minutes, after bucket b_0, a major update introduces 10 new high-priority red tasks. Started tasks are removed, each robot's partial execution state is carried into the replan, and the schedule is recomputed over the remaining horizon. The bottom panel shows the replanned schedule (Plan 2), where bucket quotas shift to prioritize the new red tasks. The optimizer inserts all 10 new tasks while maintaining bucket-level reservation limits and continuous-time feasibility. As a result, red tasks dominate buckets 2–5, while blue and yellow missions continue at reduced levels in line with the updated policy. In the no-quotas schedules (left column), robots may drift toward locally convenient tasks. Reservations still appear there, but they simply reflect the mission of the task executed in each bucket rather than encoding mission-level intent.

The reservation- and quota-enforced schedules (right column) incorporate partial mission knowledge—specifically, which robots must be available for particular missions in particular buckets (reservations, e.g., $y_{0,1,3} = 1$), and how many robots each mission requires in those buckets (quotas, e.g., $q_{1,2} = 3$, $q_{1,3} = 4$). As observed in the right column, the set of reserved robots for bucket b_2 changes between the plan and the replan: initially robots r_1–r_3 are reserved, while in the replan robots r_0–r_2 fulfill the requirement. Critically, although the identity of the reserved robots changes, the quota for Mission 1 in bucket b_2 remains three. If even more control over the assignment is required, specific $y_{r,m,b}$ variables can be directly fixed to desired values, thereby enforcing that particular robots serve particular missions in particular buckets. Reservations constrain *which* robot must serve a mission in a given bucket, whereas quotas constrain *how many* robots must do so. Incorporating this prior mission knowledge ensures that the required capacity is preserved for the missions when the corresponding buckets occur. Even though cost is not the primary focus of this example, it is informative to observe how the different formulations influence the objective. In both initial plan cases, the total cost is identical, since the same set of 20 tasks is scheduled and the reservations in the no-quotas case simply reflect the missions of the tasks already being executed in each bucket.

In the replan stage, the situation changes after the ten new tasks arrive. The quota-enforced formulation achieves lower cost in this instance because it preserves robot capacity for future mission demands, enabling more efficient assignment of the new tasks. By contrast, the no-quotas formulation may use robots too early, making later tasks hard or impossible to serve. This is visible in Fig. 2 (bottom left), where robot r_0 has an empty slot in bucket b_3: after its earlier assignment, it cannot feasibly reach task 24 from task 14 in time. The bucket is therefore left unassigned, increasing cost or delaying execution. This should not be interpreted as a general dominance result. In this case, quotas improve replanning because they preserve capacity for later-arriving tasks. If quotas are mis-specified or overly restrictive, they can instead overconstrain the scheduler and lead to higher cost, longer travel, or idle time. The main purpose here is not cost minimization alone, but ensuring that sufficient robot capacity is reserved for the required missions in the designated buckets.

7. CONCLUSION

We presented a mixed-integer framework for multi-mission multi-robot task allocation that combines continuous-time routing with a bucketed reservation layer. Using binary reservation variables and start-bucket selectors, the model enforces per-bucket mission quotas and one-mission-per-robot-per-bucket policies while preserving continuous start times. The makespan objective, via $T_{\max}$, promotes coordinated mission progress by minimizing the latest completion time. A post-storm response scenario showed how bucket-level reservations provide an operational control interface: changing quotas over time reconfigures robot allocations according to updated policies, which a purely continuous-time baseline cannot guarantee. The formulation focuses on continuous-time routing with bucketed mission-capacity control. It does not yet include battery depletion, explicit inter-task precedence, mission-level ordering rules, or heterogeneous mission-priority weights beyond time-phased quotas; these are natural directions for future work.

REFERENCES

Cuisinier, E., Ruby, A., Bourasseau, C., Lemaire, P., and Penz, B. (2021). Rolling horizon optimization: new approaches to balance short-term and long-term decisions for energy planning. In *ROADEF 2021-22*.

Desaulniers, G., Desrosiers, J., and Solomon, M.M. (eds.) (2005). *Column Generation*. GERAD 25th Anniversary Series. Springer, New York, NY.

Glomb, L., Liers, F., and Rösel, F. (2022). A rolling-horizon approach for multi-period optimization. *EJOR*, 300(1), 189–206.

Miloradović, B., Frasheri, M., Cürüklü, B., Ekström, M., and Papadopoulos, A.V. (2019). TAMER: Task allocation in multi-robot systems through an entity-relationship model. In *PRIMA*, 478–486. Springer.

Miloradović, B., Çürüklü, B., Ekström, M., and Papadopoulos, A.V. (2022). GMP: A genetic mission planner for heterogeneous multirobot system applications. *IEEE Transactions on Cybernetics*, 52(10), 10627–10638.

Powell, W.B. (2007). *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley, Hoboken, NJ.

Smith, S.L., Hsieh, M.A., and Kumar, V. (2007). The dynamic allocation of tasks in multi-robot systems. *IEEE Transactions on Robotics*, 23(5), 991–1000.

Spaan, M.T.J., Messias, J.V., and Lima, P.U. (2015). A decision-theoretic approach to dynamic task allocation in multi-robot teams. *IJRR*, 35(8), 1099–1121.