Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Dynamic Multi-Robot Task Allocation in Multi-Floor Hospitals: An Urgency-Aware Auction Framework

Yupeng Zou

zouyupeng@upc.edu.cn

China University of Petroleum, East China

Qifan He

China University of Petroleum, East China

Zhishen Zhou

China University of Petroleum, East China

Xuejin Zhu

China University of Petroleum, East China

Yongfei Ma

China University of Petroleum, East China

Yujie Li

China University of Petroleum, East China

He Zhao

China University of Petroleum, East China

Xutao Chen

China University of Petroleum, East China

Suli Zou

Beijing Institute of Technology

Research Article

Keywords:

Posted Date: November 19th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11532793/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Dynamic Multi-Robot Task Allocation in Multi-Floor Hospitals: An Urgency-Aware Auction Framework

Yupeng Zou[1,2*], Qifan He[1,2], Zhishen Zhou[1,2], Xuejin Zhu[1,2], Yongfei Ma[1,2], Yujie Li[1,2], He Zhao[1,2], Xutao Chen[1,2], Suli Zou[3*]

[1*]Shandong Key Laboratory of Design and Manufacturing for High-end Offshore Oil and Gas Equipment, College of Mechanical and Electronic Engineering, China University of Petroleum (East China), Qingdao, 266000, Shandong, China.

[2]National Engineering Research Center of Marine Geophysical Prospecting and Exploration and Development Equipment, China University of Petroleum (East China), Qingdao, 266000, Shandong, China.

[3]School of Automation, Beijing Institute of Technology, Beijing, 100081, China.

*Corresponding author(s). E-mail(s): zouyupeng@upc.edu.cn; sulizou@bit.edu.cn;

Abstract

Multi-robot hospital delivery involves coordinating the transport of medical supplies across different floors while accommodating dynamically arriving requests and urgent delivery demands. Inter-floor elevator transfers and the need to serve multiple destinations make task allocation dependent not only on robot assignments but also on the composition and execution order of delivery queues. To address these challenges, this paper proposes a Greedy Coalition Auction Algorithm (GCAA) for dynamic task allocation in multi-floor hospitals. The algorithm constructs capacity-limited task queues through marginal-utility-based bidding, accounting for the travel costs associated with serving multiple destinations in a single delivery trip. The scheduling framework combines batch queue construction with online task adjustment. Newly arriving ordinary tasks are considered in subsequent allocation opportunities, while urgent requests trigger a preemption-and-rollback procedure that releases affected unfinished assignments

for later scheduling. Experiments conducted in a multi-floor hospital simulation environment evaluate GCAA against adapted task-allocation methods and examine its performance under different workloads and hospital layouts. The results indicate that batch queue construction improves overall delivery timeliness, while urgent-task preemption supports timely responses to emergency requests at the expense of some ordinary-task performance. GCAA achieves favorable results relative to the adapted CBBA and DSTA methods on several measures, although the adapted ADA method achieves better overall service and travel performance. These findings illustrate the trade-offs between routine delivery efficiency and urgent-task response in dynamic multi-robot hospital scheduling.

Keywords: multi-robot system, dynamic task allocation, auction mechanism, urgent-task preemption, batch queue scheduling, hospital logistics

1 Introduction

Multi-robot systems (MRS), with their autonomous navigation and cooperative decision-making capabilities, have attracted considerable research interest in logistics delivery [1], disaster relief, and agriculture, where resource allocation and robot coordination remain important research issues [2]. In hospital environments, the timely transportation of medicines, laboratory specimens, and emergency supplies is closely related to clinical workflows and the organization of daily logistics. Multi-robot collaborative delivery has therefore been investigated as a means of coordinating such transport activities across different hospital departments [3].

Multi-Robot Task Allocation (MRTA) is a core decision-making component of these systems [4]. In multi-floor hospitals, robots must accommodate delivery requests that arise during ongoing operations, including urgent tasks with tight time constraints [5]. Meanwhile, transporting supplies from a common pharmacy to multiple destinations across different floors requires consideration of elevator transfers and multi-stop delivery routes. Consequently, the travel cost and completion time associated with a robot depend not only on which tasks are assigned to it, but also on how those tasks are grouped and ordered. The arrival of urgent requests further requires the scheduler to reconsider unfinished assignments while preserving the continuity of ordinary delivery whenever possible.

Existing MRTA methods provide useful approaches to task assignment, but the above requirements call for joint consideration of delivery queues and dynamic scheduling. Nearest-neighbor greedy strategies are computationally simple, yet their task-by-task decisions may overlook the path-coupling benefits of collecting supplies once and serving several destinations along a common route, potentially resulting in repeated pharmacy visits or inter-floor travel. Consensus-based distributed auction methods such as the Consensus-Based Bundle Algorithm (CBBA) [6] coordinate assignment through bidding, bundle construction, and conflict resolution; however, their general task-allocation formulation is not specifically tailored to the common-pharmacy, multi-floor delivery queues considered in this work. Moreover, urgent

requests may require an ongoing assignment to be interrupted and its remaining ordinary tasks to be rescheduled, introducing a further interaction between routine delivery efficiency and emergency response.

To address these scheduling requirements, this paper proposes a Greedy Coalition Auction Algorithm (GCAA) for dynamic task allocation in multi-floor hospitals, adopting a hybrid framework that combines batch queue construction with online task adjustment. During batch allocation, candidate tasks are organized into capacity-limited queues according to their marginal utility, allowing task rewards and the nominal travel time of multi-destination routes to be considered together. During online operation, newly arriving ordinary tasks enter subsequent allocation opportunities, whereas urgent requests are processed through a preemption-and-rollback procedure that releases affected unfinished ordinary assignments for later reallocation. The main contributions are summarized as follows.

(1) A multi-floor hospital delivery model is established to describe within-floor paths, nominal elevator transfers, dynamically released tasks, and capacity-limited delivery queues. The corresponding reference optimization formulation considers task rewards, travel costs, and predicted deadline feasibility, providing a scheduling framework for the allocation problem.
(2) A hybrid framework combining batch queue allocation with online adjustment is developed. The batch layer constructs delivery queues for tasks accumulated within an allocation window, while the online layer accommodates newly released requests and updates assignments as execution proceeds.
(3) A marginal-utility-driven queue-growth mechanism is combined with bid-information exchange and shared task-ownership updates. For urgent requests, a preemption-and-rollback procedure prioritizes their handling and returns affected unfinished ordinary tasks to the available pool for subsequent allocation.
(4) The performance of GCAA is examined in a multi-floor hospital simulation environment through comparisons with adapted CBBA, DSTA, and ADA methods. Component ablations, restricted small-instance exact benchmarks, and experiments involving different hospital layouts, problem scales, and algorithm parameters provide complementary evidence on scheduling performance and its limitations.

The remainder of this paper is organized as follows. Section 2 reviews related work; Section 3 presents the system model and problem formulation; Section 4 elaborates the GCAA algorithm; Section 5 reports experimental results and analysis; and Section 6 concludes the paper.

2 Related Work

Gerkey and Matarić [7] introduced a classical taxonomy along three dimensions: the task requirement on robot count (SR/MR), the robot parallel capability (ST/MT), and the allocation time horizon (IA/TA). Hospital delivery falls into the MT–SR–TA category, where each task is executed by a single robot, each robot may carry and

deliver a batch of tasks per trip, and allocation unfolds along the time axis as tasks arrive dynamically, requiring route-dependent allocation decisions.

2.1 Centralized methods

Centralized methods rely on a central controller for unified decision making. Martin et al. [8] use branch-and-bound for small instances and a genetic algorithm for larger-scale approximate allocation under multiple nonlinear criteria; Nait Chabane and Guenounou [9] design an enhanced genetic algorithm for heterogeneous multi-robot allocation and planning; Zhuang et al. [10] apply cooperative task-planning optimization to heterogeneous multi-robot order picking in a warehouse; Yang et al. [11] propose a maximin-based multi-objective evolutionary algorithm for multi-robot scheduling. Exact centralized methods can attain global optima on tractable instances, although their computational burden can grow with scale; centralized coordination may also introduce single-point dependencies.

2.2 Swarm intelligence and learning

These methods offer heuristic alternatives. For instance, Zhu and Yang [12] applied self-organizing maps to task assignment in non-stationary environments; Qizilbash et al. [13] apply ant-colony optimization to joint path-and-task planning; Alitappeh and Jeddisaravi [14] design a metaheuristic optimization for task allocation under multi-robot exploration. Deep reinforcement learning has also been introduced to MRTA: Ho et al. [15] employ federated deep reinforcement learning for heterogeneous robot scheduling; Dai et al. [16] use reinforcement learning for heterogeneous multi-robot allocation and scheduling; Yu et al. [17] propose a deep-RL-assisted multimodal multi-objective bilevel optimization method. These approaches show dynamic-adaptation potential but training cost and engineering deployment remain challenging.

2.3 Market-based and auction methods

Market-based allocation methods use bids and utility valuations to coordinate robot tasks [18, 19]. Dabass and Sangwan [20] provide a comprehensive review of optimization techniques in multi-robot systems. Auction methods treat robots as bidders and tasks or task groups as auction items; allocation can be coordinated centrally or through decentralized bid exchange. The auction algorithm of Bertsekas [21] laid the theoretical foundation for this direction. Choi et al. [6] combined auction and information consensus, proposing the Consensus-Based Bundle Algorithm (CBBA), which constructs task bundles and has convergence guarantees under specified assumptions.

Related auction-based studies address coordination and task structure. Berhault et al. [22] use combinatorial auctions to exploit spatial synergy, but the bid space grows exponentially with the task count; Zhang et al. [23] introduce stochastic clustering and simulated annealing to escape local optima; Bai et al. [24] propose group-based distributed auctions that assign feasible groups of delivery tasks under capacity and time-window constraints.

Mahato et al. [25] present a consensus-based fast and energy-efficient multi-robot allocation method that effectively reduces communication delay and energy consumption; Chen et al. [26] address dynamic allocation under strict time constraints with local information sharing and real-time reallocation; Deng et al. [27] combine distributed auctions with coalitions for dynamic reallocation.

For task structure, Zlot and Stentz [28] describe tasks as trees that support distributed trading at different abstraction levels, but construction depends on domain knowledge. Coalition formation methods [29, 30] let multiple robots team up for one task; Dutta et al. [31] build coalitions via correlation clustering, while Mouradian et al. [32] design a coalition formation algorithm for large-scale natural disasters.

For urgent tasks, Ostergaard et al. [33] study distributed allocation for emergency handling, and Parker [34] proposes the fault-tolerant ALLIANCE architecture. Recently, Verma et al. [35] investigated online diverse task allocation for heterogeneous mobile robots.

Despite these advances, the hospital delivery setting considered here combines (i) shared-pickup multi-destination queues with route-dependent task interactions, (ii) dynamic arrivals requiring updates to existing allocations, and (iii) urgent-task preemption with rollback of unfinished ordinary assignments. GCAA combines batch queue construction and online allocation updates to address these requirements.

3 Problem Formulation

3.1 Multi-floor hospital environment

We represent the reference hospital environment by a five-floor occupancy-grid topology. Let $F = \{1, \dots, 5\}$ and let each floor contain a 40×100 grid. Its traversable graph is $H^{(f)} = (V^{(f)}, E^{(f)})$, where $V^{(f)}$ denotes free grid nodes and $E^{(f)}$ their adjacent connections. Eight-neighbor movements have lengths 1 and $\sqrt{2}$ for axial and diagonal steps, respectively. These are grid-length units, not physically calibrated meters. The reference layout represents a stylized hospital environment rather than a physically calibrated floor plan; additional layouts are considered in the experimental evaluation.

A location is specified by $p = (x, y, f)$, with (x, y) its planar coordinates and $f \in F$ its floor. The environment contains one pharmacy $p_P = (x_P, y_P, 1)$ and one robot base $p_O = (x_O, y_O, 1)$, both on the first floor. Two elevator shafts, represented by $E_v = \{(x_e^k, y_e^k, f) : k = 1, 2,\ f \in F\}$, connect the floors. Each delivery tour starts with batch loading at the pharmacy, followed by visits to the assigned destinations.

Let $\ell_f(a, b)$ denote the traversable within-floor path length obtained from the A* route geometry, and let v_r be robot r's nominal speed in grid-length units per simulation-time unit. The nominal inter-floor transfer duration is $\tau^E(f_a, f_b) = T_{\text{open}} + \xi |f_a - f_b|$, where T_{open} and ξ denote the fixed transfer delay and per-floor transfer duration. Accordingly, the nominal travel time is

$$\tau_r(a, b) = \begin{cases} \ell_{f_a}(a, b)/v_r, & f_a = f_b, \\ \min\limits_{(e_s, e_t) \in \mathcal{E}(f_a, f_b)} \left[\dfrac{\ell_{f_a}(a, e_s)}{v_r} + \tau^E(f_a, f_b) + \dfrac{\ell_{f_b}(e_t, b)}{v_r} \right], & f_a \neq f_b, \end{cases} \tag{1}$$

where $\mathcal{E}(f_a, f_b)$ contains admissible elevator-endpoint pairs and unreachable routes have infinite travel time. The planar path length is converted into travel time before adding the elevator-transfer duration, ensuring dimensional consistency. This nominal model does not account for elevator contention, queuing delays, or inter-robot path conflicts.

Figure 1 provides a schematic illustration of robot-assisted hospital delivery, showing the transportation of medical supplies from the pharmacy to different clinical departments. Regular requests arrive over time, whereas urgent requests invoke a separate online scheduling procedure. The scope of this work is task allocation and queue scheduling, not conflict-free multi-robot path planning.

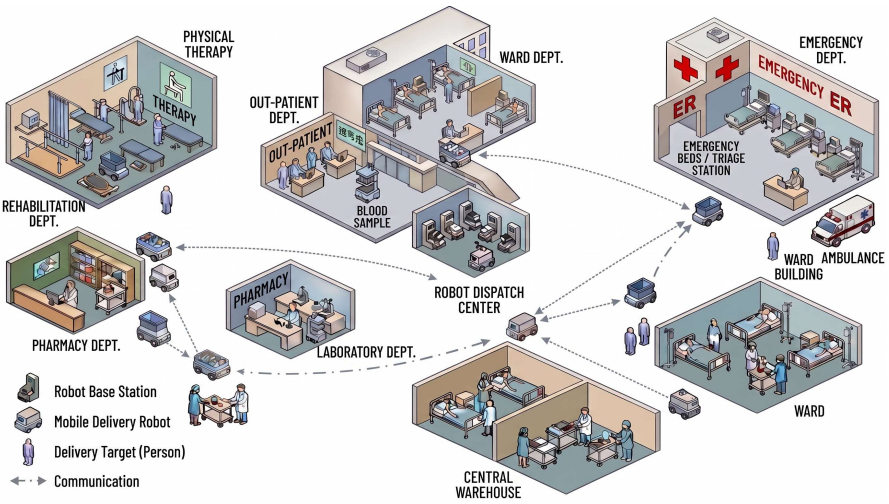


Fig. 1 Schematic illustration of the in-hospital delivery scenario

3.2 Robots, tasks, and queue travel time

The system consists of n robots, denoted by $R = \{r_1, \ldots, r_n\}$. For allocation and travel-time evaluation, robot r is characterized by its current position $p_r = (x_r, y_r, f_r)$ and nominal speed v_r, summarized as $s_r^{\text{alloc}} = (p_r, v_r)$. The simulated robots share a common nominal speed, while execution-related states are represented separately. Each robot has a batch capacity L_r, corresponding to the maximum number of tasks served after one pickup, and all robots are initially located at p_O. Battery level is not included in the allocation decisions or constraints.

At an allocation epoch, let $T = \{t_1, \ldots, t_m\}$ be the set of released tasks available for assignment. A task t is described by

$$t = (g_t, R_t, \tau_t^{\text{rel}}, \tau_t^{\text{dl}}, u_t, \delta_t), \tag{2}$$

where $g_t = (x_t, y_t, f_t)$ is its destination, R_t the task reward, τ_t^{rel} the release timestamp, τ_t^{dl} the absolute deadline, $u_t \in \{0, 1\}$ the urgency flag, and δ_t the delivery-service

duration. The deadline allowance is defined as $D_t = \tau_t^{\mathrm{dl}} - \tau_t^{\mathrm{rel}}$, and $u_t = 1$ denotes an urgent task. All temporal quantities share a common simulation clock. Deadline-feasibility constraints are activated in the deadline-enabled configuration.

For robot r, let $\sigma_r = (\sigma_r(1), \ldots, \sigma_r(k_r))$ denote its ordered delivery queue, with $k_r = |\sigma_r|$. Tasks in a queue are collected at the pharmacy before the delivery tour begins.

For $k_r > 0$, the nominal queue travel duration is

$$\begin{aligned} T_r^{\mathrm{trav}}(\sigma_r) = \tau_r(p_r, p_P) + \tau_r(p_P, g_{\sigma_r(1)}) \\ + \sum_{j=2}^{k_r} \tau_r(g_{\sigma_r(j-1)}, g_{\sigma_r(j)}), \end{aligned} \tag{3}$$

with $T_r^{\mathrm{trav}}(\varnothing) = 0$. The travel cost in the allocation objective is defined as $C_r(\sigma_r) = T_r^{\mathrm{trav}}(\sigma_r)$, excluding both pharmacy loading and delivery-service durations.

Let $\hat{\tau}_{r,t}^{\mathrm{fin}}(\sigma_r)$ denote the predicted absolute completion timestamp under the current planning state. For $t = \sigma_r(k)$, the deadline-enabled predictor follows

$$\begin{aligned} \hat{\tau}_{r,\sigma_r(k)}^{\mathrm{fin}}(\sigma_r) = \tau_r^{\mathrm{start}} + \tau_r(p_r, p_P) + \delta_P(k_r) \\ + \sum_{j=1}^{k} \big(\tau_{r,j}^{\mathrm{leg}} + \delta_{\sigma_r(j)}\big), \end{aligned} \tag{4}$$

where τ_r^{start} is the planning timestamp and $\delta_P(k_r)$ is the batch-dependent pharmacy dwell duration. For $j \geq 2$, $\tau_{r,j}^{\mathrm{leg}}$ is the nominal travel time between consecutive destinations. For the first delivery leg, the planar travel time is adjusted by a pharmacy-departure speed factor κ_P, whereas the elevator-transfer duration remains unchanged:

$$\tau_{r,1}^{\mathrm{leg}} = \frac{\ell_{P,1}}{\kappa_P v_r} + \tau_{P,1}^{E},$$

where $\ell_{P,1}$ denotes the planar path length from the pharmacy to the first destination, $\tau_{P,1}^{E}$ is the corresponding elevator-transfer duration (zero for a same-floor trip), and $\kappa_P = 2$ in the reference configuration. Thus, predicted completion includes the loading and service periods relevant to execution, whereas the travel penalty T_r^{trav} excludes them. The realized completion time, τ_t^{fin}, is measured during simulation.

3.3 Reference allocation problem and scheduling utility

According to the taxonomy of Gerkey and Matarić [7], the problem belongs to the MT–SR–TA class: each task requires one robot, each robot can perform multiple tasks sequentially, and task allocation is updated as new requests arrive. Let $x_{r,t} \in \{0, 1\}$ indicate whether task t is assigned to robot r.

The global reference allocation problem is formulated as

$$
\begin{aligned}
\text{(P0)} \quad & \max_{\{x_{r,t}\},\{\sigma_r\}} \sum_{r\in R}\sum_{t\in T} x_{r,t} R_t - \lambda \sum_{r\in R} T_r^{\mathrm{trav}}(\sigma_r) \\
\text{s.t.} \quad & \textstyle\sum_{r\in R} x_{r,t} \le 1, \quad \forall t \in T, \\
& |\sigma_r| \le L_r, \quad \forall r \in R, \\
& x_{r,t} = 1 \;\Leftrightarrow\; t \in \sigma_r, \quad \forall r, t, \\
& x_{r,t} = 1 \implies \hat{\tau}_{r,t}^{\mathrm{fin}}(\sigma_r) \le \tau_t^{\mathrm{dl}}, \quad \forall r, t, \\
& x_{r,t} \in \{0,1\},
\end{aligned}
\tag{5}
$$

where each queue contains distinct released, reachable tasks and the deadline constraint applies to the deadline-enabled reference problem. The objective rewards assigned tasks while penalizing nominal travel duration; $\lambda > 0$ converts time-valued travel cost to the reward scale. Problem (P0) is a global reference formulation for assessing reward–travel trade-offs under the stated constraints; GCAA does not solve it exactly.

The GCAA bidding mechanism evaluates candidate queues using the heuristic utility

$$
U_r(\sigma_r) = \beta(|\sigma_r|) \sum_{t\in\sigma_r} R_t p_{r,t} - \lambda C_r(\sigma_r), \tag{6}
$$

where $\beta(k)$ denotes the batch-size reward factor for a queue of k tasks, with its reference values given in Table 1. The factor $p_{r,t} \in [0,1)$ is a synthetic robot–task valuation modifier generated for the paired workloads, rather than an empirically calibrated delivery-success probability or a modeled random failure event. Candidate insertions are assessed by their marginal queue utility,

$$
\Delta U_r(t \mid \sigma_r) = U_r(\sigma_r \oplus t) - U_r(\sigma_r), \tag{7}
$$

where $\sigma_r \oplus t$ denotes insertion followed by the queue-order evaluation. This utility provides the queue valuations used in GCAA's local bidding decisions.

3.4 Urgent tasks and evaluation indicators

Urgent requests are managed by an event-driven preemption-and-rollback procedure, distinct from the ordinary-task bidding rule. In the deadline-enabled configuration, candidate urgent assignments are subject to predicted-deadline feasibility checks. Candidate robots are ranked using the nominal travel-response quantity $\tau_{r,t}^{\mathrm{urg,trav}} = \tau_r(p_r, p_P) + \tau_r(p_P, g_t)$, which differs from predicted completion time. Preemption gives the urgent request procedural precedence within the affected queue, but does not guarantee its completion before the deadline.

For the comparative evaluation in Section 5, performance indicators are derived from recorded task outcomes and robot travel segments. Let T_{done} be the completed-task set and N the number of released tasks. The completion ratio is $\eta = |T_{\mathrm{done}}|/N$;

in deadline-enabled runs, the on-time completion ratio is

$$\eta_{\text{on}} = \frac{1}{N} \sum_{t \in T_{\text{done}}} \mathbf{1}\{\tau_t^{\text{fin}} \leq \tau_t^{\text{dl}}\}. \tag{8}$$

The urgent on-time completion ratio is defined analogously, with the number of released urgent tasks in the denominator. Unfinished tasks remain in the relevant denominators but are not counted as on-time completions.

For a completed task, the release-to-completion latency is $L_t^{\text{fin}} = \tau_t^{\text{fin}} - \tau_t^{\text{rel}}$; its mean and the mean urgent-task latency are computed over their respective completed-task subsets. Urgent release-to-assignment latency uses the recorded assignment timestamp and is distinct from release-to-completion latency and the nominal travel-response cost used for urgent robot selection. Mean completion latency is reported only when the relevant category contains completed tasks.

The aggregate travel cost C_{tot} sums recorded nominal travel durations over the applicable robot legs, counting shared batch travel only once. Total completed reward is $W_{\text{tot}} = \sum_{t \in T_{\text{done}}} R_t$.

For workload balance, let C_r^{tot} be the counted travel cost of robot r. The coefficient of variation and Jain's fairness index are

$$\text{CV} = \frac{\text{std}(C_r^{\text{tot}})}{\text{mean}(C_r^{\text{tot}})}, \qquad J = \frac{(\sum_r C_r^{\text{tot}})^2}{n \sum_r (C_r^{\text{tot}})^2}. \tag{9}$$

When total recorded travel is zero, we set $\text{CV} = 0$ and $J = 1$. Allocation runtime is the cumulative wall-clock time spent in scheduling calls; communication indicators count logical message exchanges and data reads rather than physical network transmissions.

4 GCAA-Inspired Sequential Queue-Allocation Framework

This section presents a GCAA-inspired sequential queue-allocation heuristic that combines batch queue construction, a one-pass exchange of neighbor bid and winner-state information, and event-driven preemption of ordinary assignments for urgent tasks. Robots are processed in scheduler order, and each accepted assignment updates the shared task state before the next eligible robot is considered. Allocation outcomes can therefore depend on the processing order.

4.1 Hybrid task-allocation framework

The scheduler separates ordinary-task allocation from event-driven urgent-task handling (Algorithm 1). Newly released ordinary tasks enter the unassigned pool and do not interrupt a robot's active ordinary delivery queue. They remain available until an idle or standby robot reaches an allocation opportunity. At that opportunity, the allocator first reconciles local task records with the authoritative assignment state, then constructs a queue up to the robot's capacity, applies the configured queue-ordering procedure and, when enabled, deadline-feasibility checks, and commits the resulting

assignments. Execution then advances on the simulator clock and updates task and robot states.

Urgent arrivals are handled separately through a direct-assignment procedure. Robots already committed to another urgent task are excluded from direct urgent selection; when deadline constraints are enabled, a candidate must also pass the nominal predicted-completion check. Among the remaining candidates, the scheduler selects the robot with the smallest nominal travel-response time. If necessary, unfinished ordinary assignments are released and the robot's execution state is reset before the urgent task is committed. If no direct candidate passes the active checks, the urgent task remains pending for a subsequent batch-allocation opportunity. These rules define procedural precedence and a nominal feasibility filter, not a guarantee that an urgent task will be completed by its deadline.

4.2 Decomposition of queue travel cost

Under the nominal travel-time model in Eq. (1), the travel cost of robot r for a nonempty queue σ_r is decomposed into the trip to the pharmacy and the subsequent delivery tour:

$$\begin{aligned} C_r^P &= \tau_r(p_r, p_P), \\ C_r^D(\sigma_r) &= \tau_r(p_P, g_{\sigma_r(1)}) + \sum_{k=2}^{k_r} \tau_r(g_{\sigma_r(k-1)}, g_{\sigma_r(k)}), \\ C_r(\sigma_r) &= C_r^P + C_r^D(\sigma_r) = T_r^{\mathrm{trav}}(\sigma_r). \end{aligned} \tag{10}$$

For an empty queue, $C_r(\varnothing) = 0$. The point-to-point primitive $\tau_r(a, b)$ uses calibrated path length divided by robot speed, plus the modeled deterministic elevator-transfer duration for cross-floor travel. Loading and delivery-service durations are not part of this travel-cost term; they enter the queue-completion prediction used by the deadline check.

For a fixed robot and allocation state, the trip from its current position to the pharmacy is common to all candidate queues. The incremental travel cost of appending task t is therefore

$$\Delta C_r(t \mid \sigma_r) = C_r(\sigma_r \oplus t) - C_r(\sigma_r), \tag{11}$$

where $\sigma_r \oplus t$ denotes the candidate queue after applying the configured ordering procedure. The pharmacy leg is included in the travel cost used to compute both total queue utility and the stored bid. For a fixed robot, this leg is common to all nonempty candidate queues and therefore does not affect their relative ranking. It cancels from the incremental cost when the current queue is nonempty; when the current queue is empty, it contributes to the first-task marginal cost because $C_r(\varnothing) = 0$. Adding a task can also change the order of existing deliveries, so its effect cannot generally be represented by the direct travel time to the new destination alone.

Algorithm 1 Hybrid Multi-Robot Task Scheduling Framework

Require: Robot set $\mathcal{R}$, task set $\mathcal{T}$, deadline-policy flag
Ensure: Robot queues σ_r and consistent task-assignment states

```
Initialize system states, task ownership, and robot queues
while the simulation is running do
    Update the task pool and robot states
    if new tasks are released then
        for all newly released tasks t do
            if t is ordinary then
                Add t to the unassigned task pool
            else
                Form the urgent candidate set using active-urgent eligibility
                if deadline checks are enabled then
                    Retain only candidates passing nominal deadline feasibility
                end if
                if the urgent candidate set is nonempty then
                    Select the candidate with minimum nominal travel-response time
                    Release unfinished ordinary assignments if preemption is enabled
                    Commit the urgent task and synchronize ownership and queue state
                else
                    Keep t pending for a subsequent batch-allocation opportunity
                end if
            end if
        end for
    end if
    for all eligible idle or standby robots in scheduler order do
        if unassigned tasks are available then
            BatchAuctionAssignment(r, T)
        end if
    end for
    Advance execution and update task and robot states
end while
```

4.3 Construction of revenue and utility functions

The queue utility combines a task-valuation term, a batch-size reward factor, and nominal queue travel cost. For robot r and task t, $p_{r,t}$ denotes the robot–task valuation multiplier supplied to the allocator. In the current implementation it is used as a multiplier of the task reward; it should not be interpreted as an empirically calibrated delivery-success probability. The queue valuation and its batch-adjusted form are

$$\bar{R}_r(\sigma_r) = \sum_{t \in \sigma_r} R_t p_{r,t}, \qquad \widetilde{R}_r(\sigma_r) = \beta(|\sigma_r|)\bar{R}_r(\sigma_r), \tag{12}$$

with $\widetilde{R}_r(\varnothing) = 0$. The batch-size factor is

$$\beta(k) = \begin{cases} 1, & k = 1, \\ 1 + \varepsilon_2, & k = 2, \\ 1 + \varepsilon_3, & k \geq 3, \end{cases} \tag{13}$$

where the reference configuration uses $\varepsilon_2 = 0.08$ and $\varepsilon_3 = 0.15$. The batch-size factors are heuristic reward adjustments, not empirically calibrated probabilities of successful delivery. The queue utility is

$$U_r(\sigma_r) = \widetilde{R}_r(\sigma_r) - \lambda C_r(\sigma_r), \tag{14}$$

where λ converts the time-valued travel cost to the reward scale, as defined in Section 3.

For a candidate task t, the marginal contribution is

$$\begin{aligned} \Delta U_r(t \mid \sigma_r) &= U_r(\sigma_r \oplus t) - U_r(\sigma_r) \\ &= \widetilde{R}_r(\sigma_r \oplus t) - \widetilde{R}_r(\sigma_r) - \lambda \Delta C_r(t \mid \sigma_r). \end{aligned} \tag{15}$$

The reward difference includes any change in the batch-size factor applied to the queue, while ΔC_r accounts for the changed delivery tour and its ordering. At a given queue-growth step, $U_r(\sigma_r)$ is constant across candidate tasks; therefore, maximizing the candidate-queue utility is equivalent to maximizing ΔU_r. The two quantities play different roles: the marginal utility is compared with the queue-growth acceptance threshold, whereas peer competition uses the total utility of the candidate queue as the stored bid.

4.4 Auction-based task allocation and batch queue construction

The allocator constructs a queue through successive candidate evaluations and acceptance decisions (Algorithm 2). Local bid and winner records support heuristic screening, while the shared task-ownership fields determine committed assignments.

4.4.1 Auction variables and one-pass neighbor exchange

Let $b_{r,t}$ denote robot r's stored bid for task t, and let $(w_{r,t}, b^{\text{win}}_{r,t})$ denote its locally recorded winner identity and winner-bid value. At the beginning of an allocation call, robot r reads the bid vectors stored by its configured neighbors. For task t, a strictly higher observed bid replaces the local winner-bid record:

$$\text{if } b_{q,t} > b^{\text{win}}_{r,t}, \quad (b^{\text{win}}_{r,t}, w_{r,t}) \leftarrow (b_{q,t}, q), \qquad q \in \mathcal{N}_r. \tag{16}$$

This single scan provides one-hop bid information under the configured topology. Since task ownership is committed in robot-processing order, winner records serve as local screening information; the shared ownership fields remain authoritative.

Algorithm 2 Batched Bidding Task Allocation Algorithm

Require: Robot r, unassigned task set $\mathcal{T}$, bid/ownership records, ordering and deadline settings

Ensure: Robot queue σ_r and updated task assignments

Read configured neighbors' bid vectors once for this allocation call
Remove completed/released stale holdings and retain only valid owned tasks
while the queue has not reached its capacity limit **do**
    Form candidates from unassigned, uncompleted tasks not already in the queue
    Apply peer-bid screening when the exchange component is enabled
    Resolve each candidate queue's order and apply deadline feasibility if enabled
    Select the remaining candidate with maximum queue utility
    if no eligible candidate remains **then**
        break
    end if
    if the applicable first-task or subsequent-task acceptance rule passes **then**
        Append the selected task and update local bid/winner records
    else
        break
    end if
end while
if $\sigma_r \neq \varnothing$ **then**
    Resolve the final executable queue order and recheck deadlines if enabled
    Commit task ownership and synchronize the robot queue
end if

4.4.2 Candidate screening and marginal-utility queue growth

At each queue-growth step, the candidate pool contains released, uncompleted, unassigned tasks that are not already in the current queue. When winner/bid exchange is enabled, a candidate can be screened out if another robot's recorded bid exceeds the task-specific screening level

$$b_{r,t}^{\text{win}} > \phi_H R_t p_{r,t}, \tag{17}$$

with $\phi_H = 1.20$ in the reference configuration. This is a heuristic pre-screen; it is distinct from the later acceptance comparison.

For each candidate that survives screening, the allocator evaluates the utility of the resulting queue using the configured ordering routine. When deadline constraints are enabled, it first resolves the planned queue order and rejects the candidate if the nominal predicted completion time of any task in that queue exceeds its absolute deadline. The remaining candidate with the highest queue utility is selected, equivalently the candidate with the highest marginal contribution in Eq. (15).

For an empty queue, the highest-ranked eligible first task is accepted even with non-positive marginal utility, subject to peer-bid screening and any enabled deadline check. In the reference configuration, subsequent additions must satisfy the marginal-utility threshold. When bid exchange is enabled, the selected candidate must also satisfy the winner-bid comparison:

$$\begin{aligned} \Delta U_r(t^* \mid \sigma_r) > U_{\min}, \\ w_{r,t^*} = 0 \text{ or } U_r(\sigma_r \oplus t^*) > \phi_W b^{\text{win}}_{r,t^*}. \end{aligned} \tag{18}$$

Here, $U_{\min} = -5$ and $\phi_W = 1.01$. The marginal threshold is controlled independently of bid exchange. When exchange is disabled, peer-bid screening and the winner-bid comparison are not applied. The winner-bid condition compares the candidate queue's total utility with the stored winner bid, not marginal utility with marginal utility. If the selected candidate fails a required post-selection condition, queue growth stops rather than falling back to the second-ranked candidate. The resulting queue may therefore remain below capacity even when other candidates have not been tested.

4.4.3 Assignment commitment, release, and state consistency

Once a candidate is accepted, its total candidate-queue utility is written to the robot's task-bid and winner-bid records, and the local winner identity is updated. After queue construction and final checks, the task's authoritative assignment fields and the robot's executable queue are committed together. A task is retained in a robot's local held set only while the shared ownership state identifies that robot and the task remains unfinished. During allocation cleanup, completed tasks are removed from the current robot's local held set and the corresponding local bid/winner entries are cleared. Rollback or route failure releases an affected unfinished task by updating the shared ownership records and clearing stale queue, bid, and winner references across robots before the task becomes eligible for reassignment. This separation prevents a stale local winner record from overriding the committed task state.

4.4.4 Queue ordering and deadline feasibility

When enabled, queue ordering first constructs a nearest-neighbor sequence from the pharmacy and then applies 2-opt segment reversals. The implementation uses a bounded search of at most 50 iterations and accepts a reversal only when it reduces nominal travel time by more than 0.01 simulation-time units. This is a local-search heuristic and does not guarantee a globally shortest tour. The ordering routine is applied during candidate-queue evaluation, and the final queue is ordered again before commitment.

When the deadline policy is enabled, predicted completion times are computed on the common simulation clock using the nominal travel-time primitive and the configured service/dwell durations. A candidate queue is retained only if all tasks in its planned order satisfy their absolute deadlines under this prediction; the final queue is checked again before commitment. The check is a model-based feasibility filter under the current state and the modeled travel- and service-time assumptions, not a

guarantee of on-time completion under unmodeled disturbances. When the deadline policy is disabled, this feasibility filter is not applied.

4.5 Online task adjustment

Online adjustment handles ordinary arrivals and urgent arrivals through different paths (Algorithm 3). Ordinary tasks extend the pool of work available for a later allocation opportunity. Urgent tasks invoke the direct selection procedure and may trigger rollback of unfinished ordinary assignments. Both paths use the same authoritative task-ownership state, while the urgency handler applies its own eligibility, response-time, and (when enabled) deadline checks.

Algorithm 3 Dynamic Task Handling Algorithm

Require: Robot set $\mathcal{R}$, newly released tasks $\mathcal{T}_{\text{dyn}}$
Ensure: Updated task assignments and robot queues

```
for all newly released tasks t ∈ T_dyn do
    if t is ordinary then
        Add t to the unassigned task pool
        Defer t to the next eligible allocation opportunity
    else if t is urgent then
        Exclude robots already committed to another urgent task
        if deadline checks are enabled then
            Retain only robots passing urgent-task deadline feasibility
        end if
        if eligible robots remain then
            Select r* with minimum nominal urgent travel-response time
            if preemption is enabled and r* has unfinished ordinary tasks then
                Release those tasks and synchronize ownership/bid records
                Reset execution-local state for r*
            end if
            Assign t as the selected robot's priority queue
        else
            Keep t pending for a subsequent batch-allocation opportunity
        end if
    end if
end for
```

4.5.1 Incremental insertion of ordinary tasks

An ordinary task released during execution does not interrupt an active ordinary delivery queue. It enters the unassigned pool and remains pending until an idle or standby robot reaches a subsequent allocation opportunity. At that point, it is evaluated through the same queue-capacity, utility, peer-bid, ordering, and (when enabled) predicted-deadline rules as other ordinary tasks. Tasks not selected remain in the pool.

The ordinary path therefore updates the available workload incrementally rather than globally re-optimizing all active robot queues after every release.

4.5.2 Urgent selection, preemption, and rollback

For an urgent task t ($u_t = 1$), the direct handler excludes robots that already have an active urgent commitment. When deadline checks are enabled, each remaining robot is separately tested by predicting completion of the urgent task under the current nominal state; a candidate that fails this test is not considered for direct assignment. Among the remaining candidates, the selection criterion is the nominal travel-response time

$$\tau_{r,t}^{\mathrm{urg,trav}} = \tau_r(p_r, p_P) + \tau_r(p_P, g_t), \qquad r^* \in \arg\min_{r \in \mathcal{R}_t^{\mathrm{elig}}} \tau_{r,t}^{\mathrm{urg,trav}}, \tag{19}$$

where $\mathcal{R}_t^{\mathrm{elig}}$ is the set remaining after the active eligibility and, when enabled, deadline checks. The same point-to-point travel-time primitive is used for the two legs. This ranking quantity excludes service/dwell time and is not identical to predicted completion time; its minimization alone does not establish deadline feasibility.

If a robot is selected and preemption is enabled, all of its unfinished ordinary assignments in the affected queue are released and made available for later allocation. The shared ownership records are updated, and stale local and peer bid/winner entries for those tasks are cleared. Execution state from the previous route is then reset before the urgent-only queue is installed. An active urgent task is not preempted by this rule, and the same robot is not selected for two urgent tasks within one release event. If no direct candidate passes the active checks, the urgent task remains pending and may later be considered by the subsequent batch-allocation procedure.

4.5.3 Fallback handling and task-state consistency

The ordinary allocator and urgent handler share the task pool and committed ownership fields. When preemption is enabled, a direct urgent assignment releases any affected ordinary work for later allocation. If a pending urgent task is subsequently assigned through batch allocation, it is placed at the head of the executable queue and the active-urgent identity is updated accordingly. If a route fails, the affected task is released through the same ownership-update procedure and may be reconsidered by the appropriate allocation path.

4.6 Algorithm properties and complexity analysis

Let N be the number of robots, M the number of available candidate tasks, L the maximum queue capacity, $I_{\max}$ the 2-opt iteration limit, and T_{path} the computational cost of one point-to-point travel-time query on the fixed occupancy-grid environment. Each robot evaluates at most M candidates at each of at most L queue-growth steps, giving $O(NML)$ candidate-utility evaluations. This count alone is not the complete computational complexity because each candidate may require route ordering and deadline prediction.

Without 2-opt, evaluating the travel cost of a queue of length at most L requires $O(L)$ point-to-point travel queries, giving a conservative $O(NML^2T_{\mathrm{path}})$ bound for

queue construction. With 2-opt enabled, the nearest-neighbor initialization requires at most $O(L^2)$ travel queries. Each 2-opt iteration considers $O(L^2)$ segment reversals, and each candidate reversal recomputes a route cost using up to $O(L)$ legs. Thus, a conservative per-candidate ordering bound is $O(I_{\max} L^3 T_{\text{path}})$, leading to an upper bound of $O(NMI_{\max} L^4 T_{\text{path}})$ for the candidate queue evaluations when ordering is applied to every candidate. These are worst-case operation bounds; early stopping, cached travel queries, and the small configured queue capacity can reduce practical work.

For a communication graph with edge set E_c, a complete pass that invokes the allocator once per eligible robot reads at most M task bids per incident neighbor, yielding $O(M|E_c|)$ logical bid reads (or $O(N^2M)$ in the fully connected case). Each urgent arrival examines up to N robots and invokes travel-time and, when enabled, deadline-prediction checks. Actual wall-clock runtime therefore depends on route-query cost, cache behavior, queue capacity, topology, and the number of urgent arrivals; it is measured separately from simulated task-execution time. The communication count refers to logical information reads/exchanges, not physical network transmissions.

5 Experimental Results and Analysis

5.1 Simulation Platform and Experimental Protocol

The experiments were conducted in a MATLAB-based simulator representing a multi-floor hospital delivery system. The reference environment consists of five 40×100 grid floors, one pharmacy and one robot base on the first floor, and two aligned elevator shafts. The main comparison uses 14 robots, queue capacity three, 500 ordinary tasks and 30 generated urgent tasks per run, 20 normal-workload seeds (101–120), and a fixed urgent generator seed (8008). The simulator uses a 0.2-s update interval, with one simulation second corresponding to 60 modeled real seconds. Ordinary and urgent tasks have deadline allowances of 60 and 15 modeled real minutes, respectively. All methods share nominal movement, elevator-transfer, pharmacy-service, and delivery-handover settings. This reference configuration is used for the paired baseline comparison and ablation study. The multi-layout, scalability, sensitivity, and restricted exact experiments use the inputs specified in their respective subsections.

The three external baselines are hospital-adapted implementations of CBBA, following the standard bundle construction and conflict-resolution procedures [36]; DSTA, based on the sample-greedy allocation method of Shin et al. [37]; and ADA, based on the group-auction method of Bai et al. [24].

To control workload variation, each comparison method was evaluated using the same pre-generated task streams and common physical simulation settings. Their internal objectives differ: CBBA and DSTA use hospital-adapted reward–travel scores, whereas the ADA adapter selects task groups using average travel cost without reward/probability scoring. Accordingly, the comparison characterizes the performance of the hospital-adapted implementations under a common simulation protocol.

Performance is assessed using the task completion ratio, overall on-time completion ratio, nominal travel cost, release-to-completion latency, and urgent on-time

Table 1 Reference configuration and evaluation provenance.

Setting	Reference configuration
Reference map	`create_hospital_maps1`, five 40×100 floors
Pharmacy / base	One each, both on floor 1
Elevator shafts	Two, aligned across floors
Main fleet / capacity	14 robots / 3 tasks per queue
Workload	500 ordinary + 30 urgent requests in reference run
Paired repetitions	20, normal seeds 101–120; urgent seed 8008
Time step / mapping	0.2 simulation s / 1 simulation s = 60 modeled real s
Deadline allowance	60 min ordinary / 15 min urgent (modeled real time)
GCAA default λ / $U_{\min}$	0.12 / -5.0
GCAA $\beta(1)$, $\beta(2)$, $\beta(k \geq 3)$	1.00 / 1.08 / 1.15
High-bid / win-bid factors	1.20 / 1.01
Comparison algorithms	GCAA, adapted CBBA, adapted DSTA, adapted ADA

completion ratio. Both on-time ratios use the number of generated tasks in their respective categories as the denominator. Algorithm runtime is measured as wall-clock computation time and is kept separate from simulated task-execution time. Logical message-exchange counters are interpreted according to the instrumented communication model. "On-time completed reward" sums nominal rewards from tasks actually completed by their deadlines and is distinguished from the probability-weighted internal GCAA valuation.

5.2 Comparison with Adapted Baseline Algorithms

Each algorithm was evaluated on 20 matched task streams generated with seeds 101–120. Unless otherwise indicated, numerical results are reported as means across these paired runs. Higher completion ratios indicate greater service coverage, whereas lower travel cost, completion latency, and allocation runtime indicate more efficient operation. The reported cumulative allocation runtime measures computation across planning calls, whose frequencies may differ among the algorithms.

The comparison results are summarized in Fig. 2. Across the 20 reference workloads, the on-time completion ratios are 88.08% for GCAA, 87.15% for CBBA, 83.79% for DSTA and 98.65% for ADA; corresponding pure travel costs are 2870.38, 2958.29, 3138.25 and 2142.94, respectively. Their cumulative allocator wall-clock times are 203.51, 408.14, 24.43 and 249.89 s, respectively. GCAA yields somewhat higher completion and on-time ratios and lower travel cost than the adapted CBBA method, with lower cumulative allocation runtime. It also improves those service and travel metrics over the adapted DSTA method, although DSTA has substantially lower allocation runtime. The ADA adapter, however, achieves higher completion and on-time ratios, lower travel cost, and shorter mean completion latency than GCAA. ADA also achieves a slightly higher urgent on-time ratio. The algorithms differ in their allocation criteria: GCAA evaluates reward-aware batch queues, whereas the adapted ADA method selects task groups according to average travel cost.

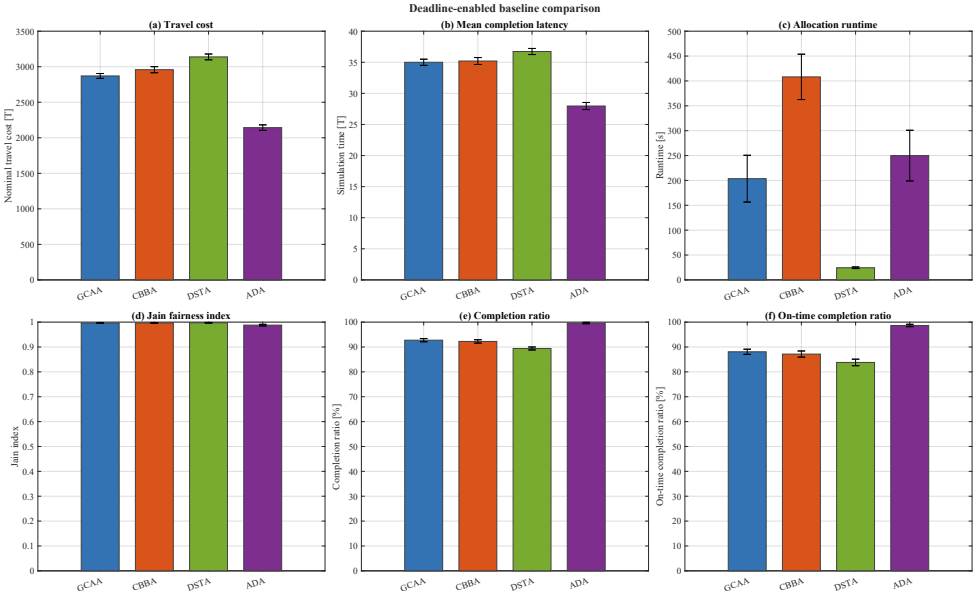


Fig. 2 Comparison of GCAA with the adapted CBBA, DSTA, and ADA baselines over 20 paired reference workloads. The six panels report service, travel, latency, runtime, and communication measures.

Urgent-task completion and deadline satisfaction must also be distinguished: GCAA's urgent completion ratio is 100% in the reference runs, whereas urgent on-time completion is 90.83%. Procedural prioritization is therefore distinguished from realized deadline satisfaction. Across the 20 paired workloads, the sample standard deviations of overall on-time completion were 1.03 percentage points for GCAA, 1.25 for CBBA, 1.31 for DSTA, and 0.48 for ADA. This spread quantifies variability across the sampled task streams.

5.3 Ablation Study of GCAA Components

To examine the contribution of individual scheduling mechanisms, the full GCAA implementation was compared with four controlled variants using the same 20 paired reference workloads (seeds 101–120). "No batching" restricts each queue to one task, which also disables multi-task bonuses and 2-opt queue reordering. "No marginal ranking" changes candidate ordering but retains marginal-utility acceptance. "No exchange" suppresses imported neighbor bids while retaining shared-state ownership. "No urgent rollback" disables immediate preemption and restricts urgent assignment to idle or waiting robots. The no-batching comparison therefore captures the joint effect of single-task queues, removal of multi-task bonuses, and omission of 2-opt reordering.

Table 2 reports the absolute outcomes of the five variants as the per-seed mean and sample standard deviation (SD). Table 3 gives the per-seed paired differences from Full GCAA, showing both effect direction and run-to-run variation. Percentage-point (pp) differences are used for completion ratios; proportional changes in travel

cost and algorithm wall-clock runtime are evaluated within each seed before averaging. The reported paired effects are descriptive and are not used to infer population-level statistical significance.

Table 2 Component-ablation outcomes (20 paired workloads per variant; mean ± sample SD). Percentage-column values and SDs are in percentage points; latency is conditional on task completion.

Variant	Completion (%)	On-time (%)	Urgent on-time (%)	Travel cost (nominal time)	Mean latency (sim s)
Full GCAA	92.75 ± 0.64	88.08 ± 1.03	90.83 ± 3.73	2870.38 ± 34.12	35.00 ± 0.49
No batching	65.88 ± 0.77	63.78 ± 0.85	91.50 ± 2.53	3399.43 ± 37.41	29.25 ± 0.53
No marginal ranking	96.02 ± 0.56	84.42 ± 1.67	92.17 ± 2.48	2976.37 ± 37.56	39.88 ± 0.76
No bid exchange	92.75 ± 0.64	88.08 ± 1.03	90.83 ± 3.73	2870.38 ± 34.12	35.00 ± 0.49
No urgent rollback	94.47 ± 0.62	90.65 ± 0.87	14.50 ± 2.48	2830.65 ± 36.83	34.26 ± 0.41

Table 3 Per-seed paired effect of disabling each GCAA component relative to the full algorithm (mean ± sample SD, $n = 20$). A positive value indicates an increase under the ablated configuration. Travel and runtime changes are ratios computed within each seed before averaging; communication changes are event-count differences. In this table, No marginal denotes No marginal ranking and No rollback denotes No urgent rollback.

Variant	ΔOn-time (pp)	ΔUrgent on-time (pp)	ΔTravel (%)	ΔRuntime (%)	ΔExch. (events)
No batching	-24.29 ± 1.27	0.67 ± 4.91	18.45 ± 1.79	-5.33 ± 17.31	-11384 ± 5364
No marginal	-3.66 ± 1.82	1.33 ± 4.24	3.70 ± 1.31	-52.55 ± 7.26	-9901 ± 5400
No bid exchange	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	-57.73 ± 2.99	-42099 ± 3969
No rollback	2.58 ± 1.05	-76.33 ± 4.45	-1.38 ± 1.05	-31.67 ± 9.36	3046 ± 5326

Restricting the queue to a single task reduces overall on-time completion by 24.29 ± 1.27 pp and raises travel cost by $18.45 \pm 1.79\%$ in paired comparisons. It also completes a smaller proportion of generated tasks; its lower conditional mean latency among *completed* tasks must not be interpreted as superior overall service because the completion denominators differ.

Removing marginal candidate ranking increases the fraction eventually completed to 96.02% from 92.75%, but on-time completion declines by 3.66 ± 1.82 pp and travel cost increases by $3.70 \pm 1.31\%$ on a paired basis. These results indicate a completion-versus-timeliness trade-off under the tested conditions. Suppressing winner/bid exchange leaves all reported service and travel outcomes unchanged for every paired seed, while reducing the mean number of logical exchanges by 42,099 events. Thus, under the tested fully connected, shared-state configuration, message exchange contributes to computational overhead without changing the observed task outcomes.

Disabling urgent rollback raises aggregate on-time completion by 2.58 ± 1.05 pp, but lowers urgent on-time completion by 76.33 ± 4.45 pp. This difference illustrates

the role of preemption in favoring urgent-task service at the expense of some aggregate outcomes. The wall-clock runtime values in Table 3 measure allocator computation across the respective scheduling calls.

5.4 Small-Instance Exact Benchmark

The solution quality of GCAA was assessed against exhaustive enumeration on five deliberately small, static instances. The objective is total selected-task reward minus λ times nominal pure queue travel cost. All benchmark probabilities are set to one; the objective excludes dynamic arrivals, deadlines, urgency handling, stochastic failure outcomes, and service/dwell terms. Enumeration includes empty sequences and every ordered feasible robot–task sequence subject to uniqueness, capacity, and reachability. The resulting optimum is therefore a reference for this restricted static objective, rather than for the full dynamic scheduling formulation.

Table 4 Restricted static exact benchmark on five test instances.

Case	Robots	Tasks	Exact objective	GCAA objective	Gap (%)
E01 same floor	1	2	109.5989	109.5989	0.0000
E02 same floor, oversubscribed	2	5	211.4481	211.3650	0.0393
E03 cross floor, oversubscribed	2	5	208.8676	208.8676	0.0000
E04 mixed floor, oversubscribed	3	7	325.7030	325.3408	0.1112
E05 mixed floor, oversubscribed	3	8	331.7482	330.8382	0.2743

Exhaustive enumeration completed in all five cases, with relative objective gaps between 0% and 0.2743%. These results provide a solution-quality reference for the tested static instances; they do not establish an approximation bound for larger or dynamic problems.

5.5 Evaluation Across Hospital Layouts

To examine sensitivity to hospital topology, GCAA was evaluated on six synthetic layouts (L01–L06), with 20 independent task-stream seeds per layout, giving 120 runs. The layouts vary corridor and obstacle topology while preserving the modeled multi-floor pickup-and-delivery structure. Seeds are matched within each layout; equal seed indices across different layouts do not represent identical spatial task instances. The layouts are synthetic grids rather than calibrated hospital floor plans.

Figure 3 presents the first-floor geometry of the six layouts. Overall on-time completion ranged from 88.08% to 94.85%: 88.08% for L01, 93.83% for L02, 94.22% for L03, 88.62% for L04, 90.77% for L05, and 94.85% for L06. These results demonstrate variation in scheduling outcomes across the six synthetic hospital layouts, which differ in corridor geometry and spatial task locations.

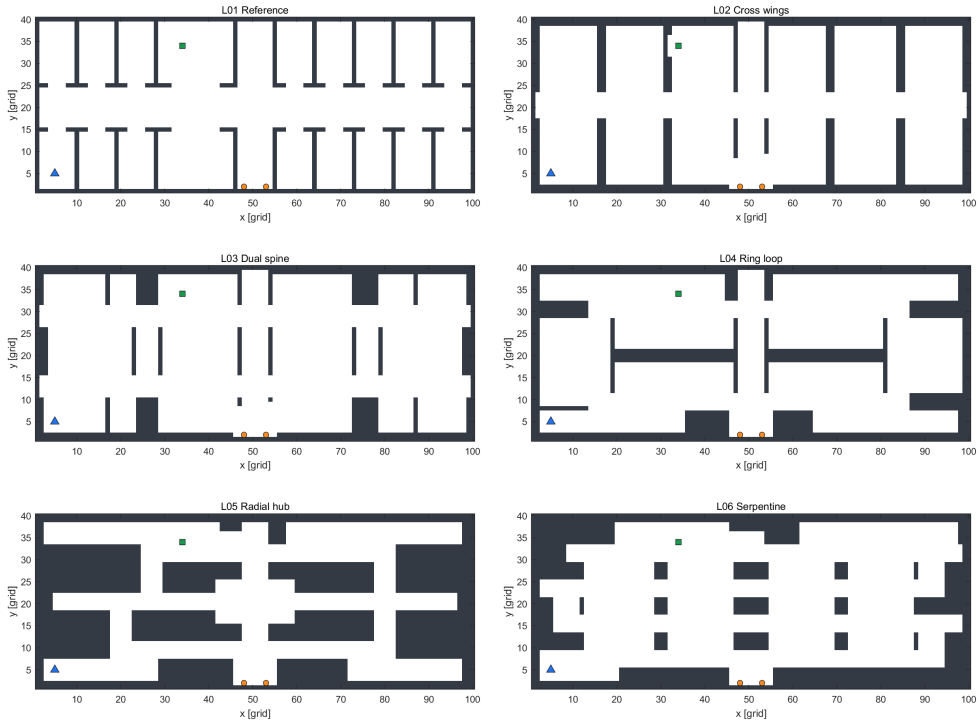


Fig. 3 First-floor layouts of six synthetic hospital environments: L01 Reference, L02 Cross Wings, L03 Dual Spine, L04 Ring Loop, L05 Radial Hub, and L06 Serpentine.

5.6 Scalability with Task Volume and Fleet Size

Scalability was examined separately with respect to task volume and fleet size. The task-volume experiment considers 100, 300, 500, 700, and 900 ordinary tasks. The fleet-size experiment varies the number of robots across 6, 10, 14, 18, and 22. Each setting uses 20 runs. Figures 4 and 5 report six outcomes: nominal travel cost, mean completion latency, allocation runtime, Jain's fairness index, completion ratio, and on-time completion ratio. Seed-indexed runs are matched within each scale study, while each scale level represents its specified workload configuration.

As task volume increases, overall completion and on-time completion decline, while mean completion latency and nominal travel cost increase. Increasing fleet size improves both service ratios, whereas travel cost varies nonmonotonically. Mean cumulative allocation runtime across 100, 300, 500, 700, and 900 ordinary tasks is 2.04, 9.09, 78.45, 418.89, and 700.30 wall-clock seconds, respectively. In the separate fleet-size experiment, mean runtime for 6, 10, 14, 18, and 22 robots is 214.84, 221.84, 82.78, 20.29, and 20.68 seconds. Runtime increases markedly with task volume over the tested range. The nonmonotonic fleet-size response indicates that runtime depends on more than robot count alone, including the number of allocation events and candidate tasks considered. Because task-volume and fleet-size variations were run in separate experimental batches, their absolute runtime levels are not interpreted as a direct cross-study

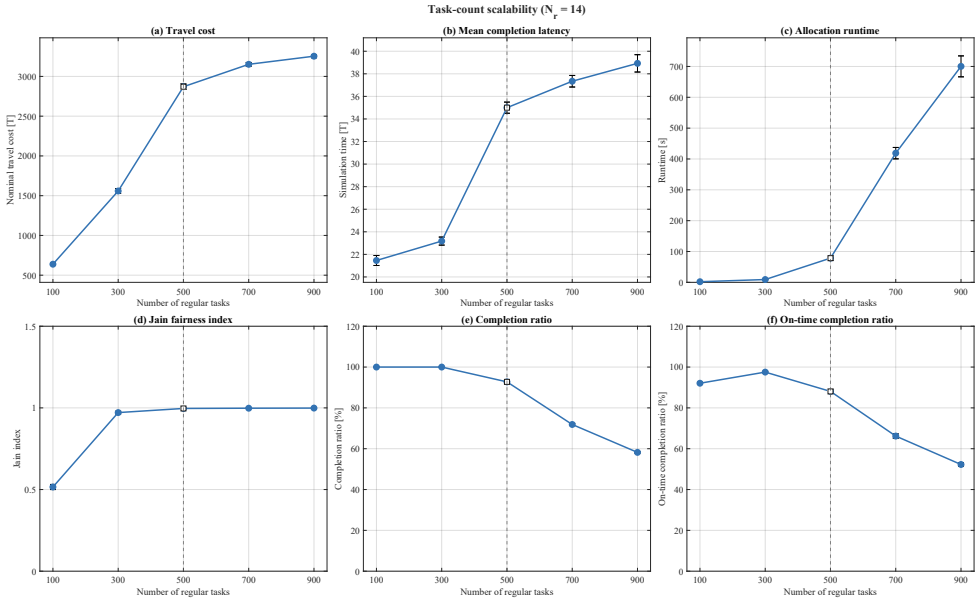


Fig. 4 Task-volume scalability of GCAA across five ordinary-task levels (100, 300, 500, 700, and 900).

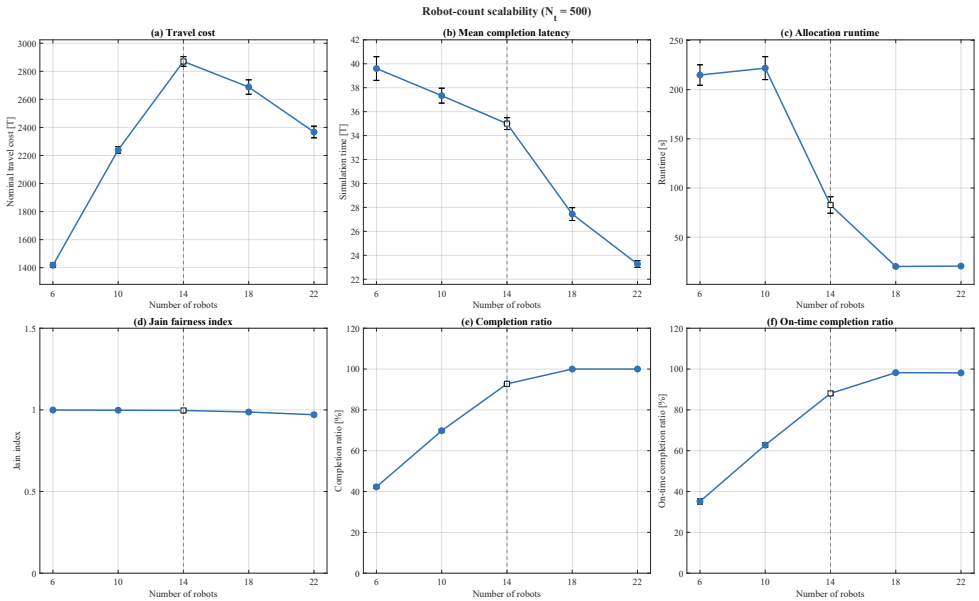


Fig. 5 Fleet-size scalability of GCAA across five robot counts (6, 10, 14, 18, and 22).

comparison. Thus, the scaling observations are confined to the evaluated task volumes and fleet sizes.

5.7 Parameter Sensitivity Analysis

A one-factor-at-a-time sensitivity study was conducted for seven GCAA parameters. Each parameter was evaluated at five levels, including a common reference configuration; this gives 29 distinct settings and 20 paired runs per setting (580 runs in total). The reference setting is shown in each parameter group in Tables 5 and 6; these entries refer to the same 20 reference runs. SD denotes the sample standard deviation across seeds. All reported parameter effects refer to the predetermined tested levels.

Table 5 reports absolute on-time performance, urgent on-time performance, nominal travel cost and conditional mean completion latency for all five tested levels of four scheduling controls. Table 6 reports the remaining bonus and bid factors, with two absolute metrics and two per-seed paired changes relative to the same reference configuration. The paired travel-cost percentage is formed separately for each seed before the mean and SD are calculated. The resulting variation is interpreted within the selected parameter ranges and reference workloads.

Table 5 One-factor-at-a-time sensitivity of GCAA's main scheduling controls (20 paired seeds per setting; mean ± sample SD). Percentage standard deviations are expressed in percentage points. Bold values indicate the shared reference setting.

Value	On-time (%)	Urgent on-time (%)	Travel cost (nominal time)	Mean latency (sim s)
Travel-cost weight λ				
0.06	87.98 ± 1.08	90.33 ± 3.57	2876.88 ± 31.75	35.02 ± 0.47
0.09	87.86 ± 1.08	90.00 ± 3.42	2878.01 ± 33.85	35.03 ± 0.47
0.12 (ref.)	88.08 ± 1.03	90.83 ± 3.73	2870.38 ± 34.12	35.00 ± 0.49
0.15	88.38 ± 1.02	90.83 ± 3.88	2864.81 ± 30.98	34.87 ± 0.43
0.18	88.37 ± 0.74	91.17 ± 3.29	2860.31 ± 32.29	34.82 ± 0.41
Queue capacity L_r				
1	63.78 ± 0.85	91.50 ± 2.53	3399.43 ± 37.41	29.25 ± 0.53
2	79.22 ± 0.92	94.00 ± 2.78	3129.47 ± 39.22	32.74 ± 0.54
3 (ref.)	88.08 ± 1.03	90.83 ± 3.73	2870.38 ± 34.12	35.00 ± 0.49
4	91.29 ± 1.01	89.67 ± 4.31	2622.43 ± 33.22	36.05 ± 0.46
5	91.66 ± 1.31	88.00 ± 3.49	2349.28 ± 48.04	36.67 ± 0.62
Marginal threshold $U_{\min}$				
-15	88.08 ± 1.03	90.83 ± 3.73	2870.38 ± 34.12	35.00 ± 0.49
-10	88.08 ± 1.03	90.83 ± 3.73	2870.38 ± 34.12	35.00 ± 0.49
-5 (ref.)	88.08 ± 1.03	90.83 ± 3.73	2870.38 ± 34.12	35.00 ± 0.49
0	88.05 ± 1.23	91.00 ± 3.44	2880.06 ± 36.10	35.01 ± 0.58
5	87.54 ± 1.07	92.33 ± 3.44	2981.13 ± 33.86	35.03 ± 0.37
Three-task factor $\beta(3)$				
1.09	88.17 ± 1.00	91.00 ± 3.76	2867.86 ± 31.65	34.97 ± 0.49
1.12	88.25 ± 0.97	90.50 ± 3.47	2868.60 ± 31.37	34.98 ± 0.49
1.15 (ref.)	88.08 ± 1.03	90.83 ± 3.73	2870.38 ± 34.12	35.00 ± 0.49
1.18	88.08 ± 1.02	90.83 ± 3.73	2869.44 ± 34.18	34.99 ± 0.51
1.21	88.09 ± 1.03	90.67 ± 3.68	2870.46 ± 32.13	34.99 ± 0.50

Table 6 Sensitivity of the remaining bonus and bid factors (20 paired seeds per setting). Outcome columns show mean ± sample SD; paired changes are computed within seed relative to the shared reference and then summarized as mean ± SD.

Value	On-time (%)	Travel cost (nominal time)	ΔOn-time (pp)	ΔTravel (%)
Two-task factor $\beta(2)$				
1.02	88.08 ± 1.03	2870.38 ± 34.12	0.00 ± 0.00	0.00 ± 0.00
1.05	88.08 ± 1.03	2870.38 ± 34.12	0.00 ± 0.00	0.00 ± 0.00
1.08 (ref.)	88.08 ± 1.03	2870.38 ± 34.12	0.00 ± 0.00	0.00 ± 0.00
1.11	88.08 ± 1.03	2870.38 ± 34.12	0.00 ± 0.00	0.00 ± 0.00
1.14	88.08 ± 1.03	2870.38 ± 34.12	0.00 ± 0.00	0.00 ± 0.00
High-bid factor				
1.00	88.08 ± 1.03	2870.38 ± 34.12	0.00 ± 0.00	0.00 ± 0.00
1.10	88.08 ± 1.03	2870.38 ± 34.12	0.00 ± 0.00	0.00 ± 0.00
1.20 (ref.)	88.08 ± 1.03	2870.38 ± 34.12	0.00 ± 0.00	0.00 ± 0.00
1.30	88.08 ± 1.03	2870.38 ± 34.12	0.00 ± 0.00	0.00 ± 0.00
1.40	88.08 ± 1.03	2870.38 ± 34.12	0.00 ± 0.00	0.00 ± 0.00
Win-bid factor				
1.000	88.08 ± 1.03	2870.38 ± 34.12	0.00 ± 0.00	0.00 ± 0.00
1.005	88.08 ± 1.03	2870.38 ± 34.12	0.00 ± 0.00	0.00 ± 0.00
1.010 (ref.)	88.08 ± 1.03	2870.38 ± 34.12	0.00 ± 0.00	0.00 ± 0.00
1.015	88.08 ± 1.03	2870.38 ± 34.12	0.00 ± 0.00	0.00 ± 0.00
1.020	88.08 ± 1.03	2870.38 ± 34.12	0.00 ± 0.00	0.00 ± 0.00

Queue capacity exhibits the clearest observed trade-off. Relative to the shared $L_r = 3$ setting, $L_r = 1$ lowers overall on-time completion by 24.29 ± 1.27 pp and increases nominal travel cost by $18.45 \pm 1.79\%$ on a paired basis, whereas $L_r = 5$ raises overall on-time completion by 3.58 ± 1.34 pp and reduces travel cost by $18.15 \pm 1.43\%$. However, urgent on-time completion falls from 90.83% at $L_r = 3$ to 88.00% at $L_r = 5$, and the conditional mean completion latency rises from 35.00 to 36.67 simulated seconds. These results show a trade-off between overall timeliness, urgent-task timeliness, travel cost, and completion latency as queue capacity changes.

Travel-cost weight λ and the three-task bonus $\beta(3)$ produce smaller changes within the examined ranges. Setting $U_{\min} = 5$, compared with $U_{\min} = -5$, lowers overall on-time completion by 0.54 ± 1.36 pp and increases travel cost by $3.87 \pm 1.20\%$ on a paired basis. Across the tested ranges, changing the two-task bonus or either bid factor does not alter the recorded overall on-time completion ratio or nominal travel cost for the paired reference workloads (Table 6).

6 Conclusion

This paper has investigated the Greedy Coalition Auction Algorithm (GCAA) for dynamic multi-robot delivery in multi-floor hospitals. The proposed framework constructs batch task queues through marginal-utility-based allocation and adjusts assignments when new requests arrive. Urgent tasks are handled through a separate preemption and rollback procedure.

The experimental results demonstrate the different roles of batch queue operation and urgent-task intervention in hospital delivery scheduling. Under the reference workload, the no-batching configuration reduced overall on-time completion by 24.29 percentage points on average, while disabling urgent rollback reduced urgent on-time completion by 76.33 percentage points. These results highlight the importance of organizing multi-destination deliveries into task queues while accommodating urgent requests through online assignment adjustments. In comparison with the adapted baseline methods, GCAA achieved better performance than CBBA and DSTA on several delivery measures, whereas ADA achieved better overall service and travel outcomes. The relative performance of these methods therefore depends on the scheduling objectives and evaluation criteria considered.

The results are specific to the simulated task streams and hospital layouts examined here. In particular, the model uses nominal elevator transfer times and idealized communication, without explicitly representing robot path conflicts or elevator contention. Evaluation with measured hospital layouts and shared-resource constraints would help establish the applicability of the method beyond these simulated conditions.

Declarations

- **Funding.** This work was supported by the Fundamental Research Funds for the Central Universities (No. 26CX02012A), and the Natural Science Foundation of Shandong Province (No. ZR2022MF291 and No. ZR2024ME244).

- **Competing interests.** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.
- **Ethics approval.** Not applicable. This study did not involve human participants or animals.
- **Consent to participate.** Not applicable.
- **Consent for publication.** Not applicable.
- **Data availability.** Data will be made available on request.
- **Code availability.** The simulation code will be made available on request.
- **Author contribution. Yupeng Zou**: Conceptualization, Funding acquisition, Project administration, Supervision, Writing – review & editing. **Qifan He**: Investigation, Data curation, Writing – original draft, Visualization. **Zhishen Zhou**: Methodology, Software, Formal analysis, Validation. **Xuejin Zhu**: Resources, Writing – review & editing. **Yongfei Ma**: Resources, Writing – review & editing. **Yujie Li**: Resources, Writing – review & editing. **He Zhao**: Resources, Writing – review & editing. **Xutao Chen**: Resources, Writing – review & editing. **Suli Zou**: Conceptualization, Supervision, Writing – review & editing.

References

[1] Camisa, A., Testa, A., Notarstefano, G.: Multi-robot pickup and delivery via distributed resource allocation. IEEE Transactions on Robotics **39**(2), 1106–1118 (2023) https://doi.org/10.1109/TRO.2022.3216801

[2] Afrin, M., Jin, J., Rahman, A., *et al.*: Resource allocation and service provisioning in multi-agent cloud robotics: A comprehensive survey. IEEE Communications Surveys & Tutorials **23**(2), 842–870 (2021) https://doi.org/10.1109/COMST.2021.3061435

[3] Marmaglio, P., Consolati, D., Amici, C., *et al.*: Autonomous vehicles for healthcare applications: A review on mobile robotic systems and drones in hospital and clinical environments. Electronics **12**(23), 4791 (2023) https://doi.org/10.3390/electronics12234791

[4] KA, A., Subramaniam, U.: A systematic literature review on multi-robot task allocation. ACM Computing Surveys **57**(3), 1–28 (2024) https://doi.org/10.1145/3700591

[5] Choudhury, S., Gupta, J.K., Kochenderfer, M.J., *et al.*: Dynamic multi-robot task allocation under uncertainty and temporal constraints. Autonomous Robots **46**, 231–247 (2022) https://doi.org/10.1007/s10514-021-10022-9

[6] Choi, H.L., Brunet, L., How, J.P.: Consensus-based decentralized auctions for robust task allocation. IEEE Transactions on Robotics **25**(4), 912–926 (2009) https://doi.org/10.1109/TRO.2009.2022423

[7] Gerkey, B.P., Matarić, M.J.: A formal analysis and taxonomy of task allocation in multi-robot systems. The International Journal of Robotics Research **23**(9), 939–954 (2004) https://doi.org/10.1177/0278364904045564

[8] Martin, J.G., Frejo, J.R.D., García, R.A., *et al.*: Multi-robot task allocation problem with multiple nonlinear criteria using branch and bound and genetic algorithms. Intelligent Service Robotics **14**(5), 707–727 (2021) https://doi.org/10.1007/s11370-021-00393-4

[9] Nait Chabane, A., Guenounou, O.: An enhanced genetic algorithm for optimized task allocation and planning in heterogeneous multi-robot systems. Complex & Intelligent Systems **11**, 435 (2025) https://doi.org/10.1007/s40747-025-02062-w

[10] Zhuang, Z., Huang, Z., Sun, Y., *et al.*: Optimization for cooperative task planning of heterogeneous multi-robot systems in an order picking warehouse. Engineering Optimization **53**(10), 1715–1732 (2021) https://doi.org/10.1080/0305215X.2020.1821198

[11] Yang, S., Zhang, Y., Ma, L., *et al.*: A novel maximin-based multi-objective evolutionary algorithm using one-by-one update scheme for multi-robot scheduling optimization. IEEE Access **9**, 121316–121328 (2021) https://doi.org/10.1109/ACCESS.2021.3105102

[12] Zhu, A., Yang, S.X.: A neural network approach to dynamic task assignment of multirobots. IEEE Transactions on Neural Networks **17**(5), 1278–1287 (2006)

https://doi.org/10.1109/TNN.2006.875994

[13] Qizilbash, A.A.H., Henkel, C., Mostaghim, S.: Ant colony optimization based multi-robot planner for combined task allocation and path finding. In: 2020 17th International Conference on Ubiquitous Robots (UR), pp. 487–493. IEEE, ??? (2020). https://doi.org/10.1109/ur49135.2020.9144944

[14] Alitappeh, R.J., Jeddisaravi, K.: Multi-robot exploration in task allocation problem. Applied Intelligence **52**, 2189–2210 (2022) https://doi.org/10.1007/s10489-021-02483-3

[15] Ho, T.M., Nguyen, K.K., Cheriet, M.: Federated deep reinforcement learning for task scheduling in heterogeneous autonomous robotic system. IEEE Transactions on Automation Science and Engineering **21**(1), 528–540 (2024) https://doi.org/10.1109/TASE.2022.3221352

[16] Dai, W., Rai, U., Chiun, J., *et al.*: Heterogeneous multi-robot task allocation and scheduling via reinforcement learning. IEEE Robotics and Automation Letters **10**(3), 2654–2661 (2025) https://doi.org/10.1109/LRA.2025.3534682

[17] Yu, Y., Tang, Q., Jiang, Q., *et al.*: A deep reinforcement learning-assisted multimodal multiobjective bilevel optimization method for multirobot task allocation. IEEE Transactions on Evolutionary Computation **29**(3), 574–588 (2025) https://doi.org/10.1109/TEVC.2025.3535954

[18] Quinton, F., Grand, C., Lesire, C.: Market approaches to the multi-robot task allocation problem: A survey. Journal of Intelligent & Robotic Systems **107**, 29 (2023) https://doi.org/10.1007/s10846-022-01803-0

[19] Milot, A., Chauveau, E., Lacroix, S., *et al.*: Market-based multi-robot coordination with htn planning. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2606–2612 (2021). https://doi.org/10.1109/IROS51168.2021.9636286

[20] Dabass, V., Sangwan, S.: Strategic allocation: Exploring optimization techniques in multi-robot systems. International Journal of Intelligent Robotics and Applications **9**, 1279–1301 (2025) https://doi.org/10.1007/s41315-025-00436-4

[21] Bertsekas, D.P.: The auction algorithm for assignment and other network flow problems: A tutorial. Interfaces **20**(4), 133–149 (1990) https://doi.org/10.1287/inte.20.4.133

[22] Berhault, M., Huang, H., Keskinocak, P., *et al.*: Robot exploration with combinatorial auctions. In: Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), pp. 1957–1962 (2003). https://doi.org/10.1109/IROS.2003.1248932

[23] Zhang, K., Collins, E.G., Shi, D.: Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction. ACM Transactions on Autonomous and Adaptive Systems **7**(2), 1–22 (2012) https://doi.org/10.1145/2240166.2240171

[24] Bai, X., Fielbaum, A., Kronmüller, M., *et al.*: Group-based distributed auction algorithms for multi-robot task assignment. IEEE Transactions on Automation Science and Engineering **20**(2), 1292–1303 (2023) https://doi.org/10.1109/TASE.2022.3175040

[25] Mahato, P., Saha, S., Sarkar, C., *et al.*: Consensus-based fast and energy-efficient multi-robot task allocation. Robotics and Autonomous Systems **159**, 104270 (2023) https://doi.org/10.1016/j.robot.2022.104270

[26] Chen, X., Zhang, P., Du, G., *et al.*: A distributed method for dynamic multi-robot task allocation problems with critical time constraints. Robotics and Autonomous Systems **118**, 31–46 (2019) https://doi.org/10.1016/j.robot.2019.04.012

[27] Deng, R., Yan, R., Huang, P., *et al.*: A distributed auction algorithm for task assignment with robot coalitions. IEEE Transactions on Robotics **40**, 4787–4804 (2024) https://doi.org/10.1109/TRO.2024.3475209

[28] Zlot, R., Stentz, A.: Market-based multirobot coordination for complex tasks. The International Journal of Robotics Research **25**(1), 73–101 (2006) https://doi.org/10.1177/0278364906061160

[29] Sandholm, T.W., Lesser, V.R.: Coalitions among computationally bounded agents. Artificial Intelligence **94**(1–2), 99–137 (1997) https://doi.org/10.1016/S0004-3702(97)00030-1

[30] Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. Artificial Intelligence **101**(1–2), 165–200 (1998) https://doi.org/10.1016/S0004-3702(98)00045-9

[31] Dutta, A., Ufimtsev, V., Asaithambi, A.: Correlation clustering based coalition formation for multi-robot task allocation. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, pp. 906–913 (2019). https://doi.org/10.1145/3297280.3297369

[32] Mouradian, C., Sahoo, J., Glitho, R.H., *et al.*: A coalition formation algorithm for multi-robot task allocation in large-scale natural disasters. In: 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 1909–1914 (2017). https://doi.org/10.1109/IWCMC.2017.7986575

[33] Ostergaard, E.H., Matarić, M.J., Sukhatme, G.S.: Distributed multi-robot task allocation for emergency handling. In: Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 821–826 (2001). https://doi.

org/10.1109/iros.2001.976270

[34] Parker, L.E.: ALLIANCE: An architecture for fault tolerant multirobot cooperation. IEEE Transactions on Robotics and Automation **14**(2), 220–240 (1998) https://doi.org/10.1109/70.681242

[35] Verma, A., Gautam, A., Duhan, T., *et al.*: HMR-ODTA: Online diverse task allocation for a team of heterogeneous mobile robots. Intelligent Service Robotics **19**, 1–18 (2026) https://doi.org/10.1007/s11370-025-00666-2

[36] Chen, J., Qing, X., Ye, F., Xiao, K., You, K., Sun, Q.: Consensus-based bundle algorithm with local replanning for heterogeneous multi-UAV system in the time-sensitive and dynamic environment. The Journal of Supercomputing **78**, 1712–1740 (2022) https://doi.org/10.1007/s11227-021-03940-z

[37] Shin, H.-S., Li, T., Lee, H.-I., Tsourdos, A.: Sample greedy based task allocation for multiple robot systems. Swarm Intelligence **16**, 233–260 (2022) https://doi.org/10.1007/s11721-022-00213-0