IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 6 March 2026, accepted 29 March 2026, date of publication 3 April 2026, date of current version 24 April 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3680629

RESEARCH ARTICLE

Hybrid Coordination Framework for Centralized Task Allocation and Execution in Heterogeneous Multi-Robot Systems

GLACE VARGHESE T.[1], (Member, IEEE), SREEJA KOCHUVILA[1], (Senior Member, IEEE), NAVIN KUMAR[1], (Senior Member, IEEE), AND R. R. VENKATESHA PRASAD[2], (Senior Member, IEEE)

[1]Department of Electronics and Communication Engineering, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Bengaluru 560035, India
[2]Department of Electrical Engineering, Mathematics, and Computer Science, TU Delft, 2628 XE Delft, The Netherlands

Corresponding author: Sreeja Kochuvila (k_sreeja@blr.amrita.edu)

ABSTRACT Multi-robot coordination has emerged as a critical enabler for efficient warehouse logistics, yet existing approaches struggle to balance solution quality, computational efficiency, and real-time adaptability in heterogeneous robot environments. This paper presents a novel centralized coordination architecture that integrates a hybrid optimization algorithm combining the shapley value clustering algorithm (SVCA) with the non-dominated sorting genetic algorithm II (NSGA-II) for multi-robot task allocation (MRTA). The proposed system addresses the complex challenge of allocating logistics tasks to heterogeneous mobile robots with varying payload capacities, energy consumption rates, and task deadline while simultaneously optimizing multiple conflicting objectives including makespan, energy consumption, deadline adherence, and workload balance. Unlike traditional static scheduling methods, the proposed system includes dynamic task reallocation capabilities that enable robots to autonomously detect and execute pending tasks nearby upon completion, be it with intelligent battery management that triggers autonomous recharging when energy thresholds are reached. Implemented using robot operating system 2 (ROS2) Humble and validated in Gazebo simulation environments with warehouse scenarios involve five heterogeneous robots and multiple logistics tasks. Experimental results show that, with the hybrid algorithm, 100% task allocation can be achieved with an average allocation time of 0.122 seconds, significantly outperforming state-of-the-art methods including island model genetic algorithm (IMGA), SVCA, particle swarm optimization (PSO), genetic algorithm (GA), and standalone NSGA-II executions. The system effectively manages complex logistics operations to automate the warehouse within the simulated environment, while maintaining required constraint.

INDEX TERMS Multi-robot system (MRS), multi-robot task allocation (MRTA), gazebo, heterogeneous robots, non-dominated sorting genetic algorithm-II (NSGA-II), robot operating system (ROS2), RViz, shapley value clustering algorithm (SVCA), task allocation.

I. INTRODUCTION

Multi-robot coordination has emerged as an important branch of robotics research, driven by the increased demand for efficient, scalable, and intelligent systems that can cooperatively perform complex tasks, which may be difficult or impossible for a single robot to accomplish. Through cooperation, robots are capable of sharing information, distributing workloads, and adapting to dynamic environments to enhance overall system robustness, efficiency, and task completion speed. Multi-robot coordination is expected to have applications in broad areas, including search and rescue, environmental monitoring, transportation, industrial automation, and defense, where decision-making and resource optimization must occur

The associate editor coordinating the review of this manuscript and approving it for publication was Hui Xie.

in real-time [1]. Although there has been significant progress, several open challenges, such as communication constraints, task allocation, collision avoidance, handling uncertainty, and scalability, tend to drive new approaches and frameworks for effective coordination. Within this context, centralized task allocation has been one of the widely explored approaches for multi-robot coordination, wherein one central controller or decision-making unit is assigned the responsibility of task assignment for individual robots with the help of global system knowledge. It enables optimized resource utilization, maintains a balanced distribution of workload, and promotes the efficient scheduling of tasks through consideration of the overall state of the environment and robots. Most centralized methods utilize optimization algorithms, graph-based models, or heuristics to obtain near-optimal allocations and are therefore well-suited for applications with high demands on precision and coordination. However, their dependency on the single point of control brings challenges of scalability, fault tolerance, and communication overhead, underlining the needs for robust design and hybrid strategies beyond the current limitations [2].

Building on this, when the Multi-robot system (MRS) is composed of heterogeneous robots with different capabilities, the centralized coordination system must consider such differences in sensing, mobility, payload capacity, and energy consumption when allocating tasks. In such cases, the process of allocation clearly becomes more complicated, since tasks need to be allocated not only to available but also to well-suited robots for their successful and efficient execution [3]. Comprehensive information on the capabilities of the robots, task requirements, and environmental conditions is concentrated in the centralized controller, ensuring compatibility between tasks and robots. Optimization or auction-based mechanisms ensure minimum overall execution cost and maximum efficiency and reliability of the system. Moreover, in heterogeneous settings, the centralized system ensures continuous and adaptive coordination between diverse robot types through runtime monitoring of ongoing executions, dynamic reallocation in front of failures, and scheduling of recharge or maintenance shifts.

These market-based and optimization-based methods are relevant in the context of heterogeneous multi-robot systems so as to solve the problems in MRTA in an efficient manner by matching the task needs with the various capabilities of the robots. The market-based auctions offer a flexible allocation of tasks dynamically, and optimization-based methods, including genetic algorithm (GA), particle swarm optimization (PSO), shapley value clustering algorithm (SVCA) and non-dominated sorting genetic algorithm II (NSGA-II) give a chance to explore large and complex space of solutions to find near-optimal allocations [4], [5], [6], [7]. Genetic algorithms particularly, are quite appropriate to heterogeneous environments since they will be able to adaptively evolve solutions that consider the different characters of the robots and different environmental constraints in the environment in question. Other improved versions such as the island model genetic algorithm (IMGA) also improve the performance of the system by keeping the diversity in various subpopulations so that the system can escape the effect of premature convergence and can also deal with large scale and complicated situations of task allocation to various subpopulations. These means enable centralized coordination systems to effectively distribute tasks, reassign when failure follows and to carry out effectively in various conditions that are diverse and dynamic. The key contributions are:

- A simple and efficient centralized coordination approach for heterogeneous mobile robots with varying payload capacities, energy consumption, task deadlines, and average speeds.
- A coordination system that schedules and allocates tasks using the proposed hybrid algorithm (SVCA + NSGA-II), which optimizes multiple objectives.
- The coordination architecture with the hybrid scheduler enables the system to operate in an online manner, which is not feasible with existing static scheduling approaches.
- The proposed system with dynamicity and real-time adaptability: once a robot completes a task and returns to its parking station, it can immediately detect and execute nearby pending tasks before returning to parking. Furthermore, the system autonomously manages robot recharging once a predefined energy threshold is reached. This dynamic task reallocation and adaptive energy management highlight the key novelty of the proposed approach in handling real-world logistics scenarios efficiently and reliably.

The remainder of the paper is organized as follows: The section II presents recent and most relevant literature. Section III describes the problem statement and mathematical formulation and the section IV focuses on the system architecture and modeling while the section V discusses the proposed hybrid optimization algorithm. Section VI describes the implementation details and section VII the experimental results and analysis. Finally section VIII concludes the work with overall conclusion and future scope of the work.

II. RECENT AND MOST RELEVANT LITERATURE

A. CLASSICAL AND MARKET-BASED MRTA APPROACHES

Early MRTA research predominantly focused on classical centralized and market-based formulations, where task allocation is performed using auctions, bidding mechanisms, or greedy optimization strategies. Market-based MRTA approaches, such as those proposed by Gerkey and Mataric [37], model robots as rational agents that bid for tasks based on local cost estimates, resulting in low computational complexity and fast allocation decisions. However, these methods generally converge to suboptimal solutions and exhibit limited adaptability when task interdependencies,

TABLE 1. Comparison of state of the art and proposed approach.

Ref.	Problem Formulation	Crossover Operator	Mutation Operator	Generational Structure	Convergence Behavior	Computational Cost	Adaptability
[36]	Evolutionary TSP-based MRTA (single-objective)	Partially Matched Crossover (PMX)	Swap mutation	Standard generational GA	Requires high number of generations (>100) for stability	High	Limited (static task set)
[37]	Market-based MRTA	N/A	N/A	Iterative allocation rounds	Greedy convergence	Low	Low
[18]	Time-extended MRTA	One-point crossover	Random reset	Steady-state GA	Slow convergence	High	Moderate
[21]	Multi-objective MRTA	Uniform crossover	Gaussian mutation	NSGA-II	∼80–100 generations	High	High
[7]	Energy-aware MRTA	Two-point crossover	Bit-flip mutation	NSGA-II	∼100 generations	High	High
Proposed Work	Hybrid MRTA (SVCA + NSGA-II)	Problem-aware crossover preserving task–robot consistency	Adaptive mutation based on population diversity	NSGA-II with elitism	Fast convergence within ∼30 generations	Moderate (balanced)	High (dynamic, scalable)

temporal constraints, or heterogeneous robot capabilities are introduced. Centralized and graph-based planners improve solution optimality but suffer from scalability limitations due to exponential growth in the solution space as the number of robots and tasks increases.

B. EVOLUTIONARY AND GA-BASED MRTA APPROACHES

To overcome the limitations of classical methods, evolutionary and GA–based approaches have been widely explored for MRTA. A representative work is the evolutionary Traveling salesman problem (TSP) formulation by Arif and Haider [36], where task sequences are encoded as permutations and optimized using standard GA operators such as partially matched crossover (PMX) and swap mutation. Although this method shows better solution quality than heuristic approaches, it is still single-objective and requires a high number of generations (usually more than 100) to converge. Later extensions of the GA approach incorporated integer and priority encoding to address time-extended MRTA problems [18]. However, the major challenges in these approaches are still convergence speed and adaptability in dynamic environments.

C. MULTI-OBJECTIVE EVOLUTIONARY MRTA AND BENCHMARKING ALGORITHMS

More recent works have used multi-objective evolutionary approaches like NSGA-II to optimize multiple conflicting objectives like task completion time, energy consumption, and workload balance simultaneously [7], [21]. These methods use dominance-based selection and elitism to enhance Pareto optimality but often use generic crossover and mutation operators, resulting in convergence after 80-100 generations with high computational complexity. In performance analysis, GA, PSO, NSGA-II, IMGA, and SVCA are generally used as comparison benchmarks; Moreover, comparison analysis among existing algorithms reveals significant differences in the parameters and constraints taken into consideration during task allocation. IMGA generally considers weight, battery, and speed [1]; SVCA considers path length, speed, and time; PSO takes into account battery status; GA emphasizes time and path length; and NSGA-II considers makespan and computational time. The percentage of complete solutions (PCS), average rate of task allocation (ARTA), and average time to allocate task (ATAT) are highly affected by the choice of parameters and constraints in these algorithm designs. In addition, each algorithm exhibits distinct limitations IMGA often converges to suboptimal local minima and lacks fairness in coalition formation; SVCA has restricted global optimization ability; PSO [7]can easily become trapped in local optima within discrete spaces; GA performs well for single-objective problems but lacks multi-objective depth; and NSGA-II [4], [8], while multi-objective, suffers from slow convergence. These algorithmic shortcomings collectively indicate the need for a more robust, adaptive, and computationally efficient hybrid framework that can overcome these limitations and improve the overall performance and reliability of MRTA in heterogeneous and dynamic environments [2], [9], [10], [11].

D. COMPARATIVE ANALYSIS OF EXISTING MRTA APPROACHES

Table 1 provides a comparative summary of some of the representative features of the MRTA approaches using some of the important aspects of any algorithm, viz., problem formulation approach, evolutionary operators incorporated into these algorithms, generational characteristics of these MRTAs, convergence behavior of these algorithms with their evaluation of computational costs and their flexibility or adaptability of these approaches. The hybrid approach is based on achieving better quality of solutions with low computational costs and high levels of adaptability, unlike other approaches. Generally, evolution-based MRTAs require more than 80-100 generations for convergence of these approaches. However, with problem-aware crossover and adaptive mutations introduced in this proposed approach, it requires fewer generations of evolution for achieving better quality of solutions.

E. RESEARCH GAP AND MOTIVATION

From the above review, it is evident that existing MRTA solutions face a fundamental trade-off between adaptability and computational efficiency. Classical methods lack flexibility, while GA and multi-objective evolutionary approaches incur high computational cost due to slow convergence and insufficient exploitation of problem-specific structures. Motivated by these gaps, this work proposes a hybrid MRTA framework that integrates SVAC with NSGA-II to achieve fast convergence within approximately 30 generations, explicit constraint satisfaction, and balanced adaptability for dynamic ST–SR task allocation scenarios.

III. PROBLEM STATEMENT AND MATHEMATICAL FORMULATION

A. PROBLEM OVERVIEW

The multi-robot task allocation and coordination problem addressed in this work involves assigning a set of logistic tasks to a heterogeneous team of mobile robots operating within a centralized coordination framework. The system must simultaneously optimize multiple objectives including task completion time, energy consumption, and workload balance, while satisfying various constraints related to robot capabilities, task deadlines, battery levels, payload and operational safety. Unlike traditional MRTA problems that focus on static task allocation, this work considers dynamic task reallocation, autonomous energy management, and real-time system adaptability in heterogeneous robot environments.

B. SYSTEM COMPONENTS AND NOTATION

1) ROBOTS

Let $R = \{r_1, r_2, \ldots, r_n\}$ denote the set of n heterogeneous mobile robots operating in the system. Each robot $r_i \in R$ is characterized by a tuple of attributes that define its operational capabilities:

$$r_i = (mp_i, v_i, E_i^{\max}, E_i^{\text{curr}}(t), bdf_i, dbt_i, p_i(t), p_i^{\text{init}}, t_i^{\text{load}}) \quad (1)$$

where the individual components are defined as follows:

- $mp_i \in \mathbb{N}^+$: Maximum payload capacity (units)
- $v_i \in \mathbb{R}^+$: Average navigation speed (m/s)
- $E_i^{\max} \in \mathbb{R}^+$: Maximum battery capacity (Wh)
- $E_i^{\text{curr}}(t) \in \mathbb{R}^+$: Current battery level at time t (Wh)
- $bdf_i \in \mathbb{R}^+$: Battery discharge factor (%/s)
- $dbt_i \in \mathbb{R}^+$: Discharged battery threshold (%)
- $p_i(t) \in \mathbb{R}^2$: Current position at time t
- $p_i^{\text{init}} \in \mathbb{R}^2$: Initial parking station location
- $t_i^{\text{load}} \in \mathbb{R}^+$: Time required to load/unload payload (s)

The heterogeneous nature of the robot fleet implies that robots possess different capabilities in terms of payload capacity, speed, and energy characteristics [13]. At any given time t, the robot set R can be partitioned into four mutually exclusive subsets based on operational status:

$$R = R_a(t) \cup R_b(t) \cup R_r(t) \cup R_m(t) \quad (2)$$

$$R_a(t) \cap R_b(t) \cap R_r(t) \cap R_m(t) = \emptyset \quad (3)$$

where:

- $R_a(t)$: Set of available robots ready to accept new tasks
- $R_b(t)$: Set of busy robots currently executing tasks
- $R_r(t)$: Set of robots in recharging mode
- $R_m(t)$: Set of malfunctioning or out-of-service robots

2) TASKS

Let $T = \{\tau_1, \tau_2, \ldots, \tau_m\}$ represent the set of m logistic tasks to be executed by the robot team. Each task $\tau_j \in T$ is defined by a tuple of attributes specifying its requirements and constraints:

$$\tau_j = (l_j^{\text{pickup}}, l_j^{\text{delivery}}, w_j, d_j, \delta_j, t_j^{\text{op}}) \quad (4)$$

where:

- $l_j^{\text{pickup}} \in \mathbb{R}^2$: Pickup location coordinates
- $l_j^{\text{delivery}} \in \mathbb{R}^2$: Delivery location coordinates
- $w_j \in \mathbb{N}^+$: Payload weight/units required
- $d_j \in \mathbb{R}^+$: Task deadline (seconds from system start time)
- $\delta_j \in \mathbb{R}^+$: Task priority weight
- $t_j^{\text{op}} \in \mathbb{R}^+$: Operational time at task location (s)

Throughout the system operation, each task transitions through various states forming a lifecycle. The complete task set can be partitioned into five mutually exclusive subsets representing these states:

$$T = T_{wa} \cup T_a \cup T_e \cup T_c \cup T_f \quad (5)$$

$$T_{wa} \cap T_a \cap T_e \cap T_c \cap T_f = \emptyset \quad (6)$$

where:

- T_{wa}: Tasks waiting for allocation
- T_a: Tasks that have been allocated to robots
- T_e: Tasks currently in execution
- T_c: Tasks successfully completed
- T_f: Tasks that have failed during execution

3) ENVIRONMENT AND LOCATION

Let $L = \{l_1, l_2, \ldots, l_k\}$ denote the set of all relevant locations in the operational environment. This set encompasses various location types essential for system operation:

$$L = L_p \cup L_d \cup L_{\text{park}} \cup L_{\text{charge}} \quad (7)$$

where:

- $L_p \subseteq L$: Set of task pickup locations
- $L_d \subseteq L$: Set of task delivery locations
- $L_{\text{park}} \subseteq L$: Set of robot parking stations
- $L_{\text{charge}} = \{c_1, c_2, \ldots, c_s\} \subseteq L$: Set of s charging stations

For any two locations $l_a, l_b \in L$, we define the distance function:

$$d(l_a, l_b) : L \times L \to \mathbb{R}^+ \tag{8}$$

which returns the shortest navigable path distance (in meters) between locations l_a and l_b.

C. DECISION VARIABLES

The multi-robot task allocation and coordination problem requires determining optimal values for the following decision variables:

1) PRIMARY ALLOCATION VARIABLES

The fundamental allocation decision is captured by a binary assignment matrix $X = [x_{ij}]_{n\times m}$, where:

$$x_{ij} = \begin{cases} 1, & \text{if robot } r_i \text{ is assigned to execute task } \tau_j; \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

2) TASK SCHEDULING VARIABLES

For each robot r_i, we define:

$$\sigma_i = [\tau_{i1}, \tau_{i2}, \ldots, \tau_{ik_i}] \tag{10}$$

representing the ordered sequence of k_i tasks assigned to robot r_i, where the order determines the execution sequence. Additionally, for precise temporal scheduling, we define:

$$t_{ij}^{\text{start}} \in \mathbb{R}^+ \tag{11}$$

$$t_{ij}^{\text{end}} \in \mathbb{R}^+ \tag{12}$$

representing the start time and completion time of task τ_j when executed by robot r_i, respectively.

3) CHARGING DECISION VARIABLES

To manage robot energy autonomously, we introduce:

$$y_i^{\text{charge}}(t) = \begin{cases} 1, & \text{if robot } r_i \text{ is scheduled to charge at time } t; \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

$$c_i \in L_{\text{charge}} \tag{14}$$

where c_i represents the assigned charging station for robot r_i when charging is required.

D. DERIVED FUNCTIONS AND PARAMETERS

1) ENERGY CONSUMPTION MODEL

The total energy consumed by robot r_i to execute task τ_j consists of two components:

$$E_{ij}^{\text{task}} = E_{ij}^{\text{travel}} + E_{ij}^{\text{operation}} \tag{15}$$

where the travel energy consumption is given by:

$$E_{ij}^{\text{travel}} = [d(p_i^{\text{current}}, l_j^{\text{pickup}}) + d(l_j^{\text{pickup}}, l_j^{\text{delivery}})] \times bdf_i \tag{16}$$

and the operational energy consumption is:

$$E_{ij}^{\text{operation}} = (t_i^{\text{load}} + t_j^{\text{op}}) \times bdf_i \tag{17}$$

Here, p_i^{current} represents the current position of robot r_i at the time of task allocation.

2) TIME ESTIMATION MODEL

Similarly, the total time required for robot r_i to complete task τ_j is decomposed as:

$$T_{ij}^{\text{task}} = T_{ij}^{\text{travel}} + T_{ij}^{\text{operation}} \tag{18}$$

where

$$T_{ij}^{\text{travel}} = \frac{d(p_i^{\text{current}}, l_j^{\text{pickup}}) + d(l_j^{\text{pickup}}, l_j^{\text{delivery}})}{v_i} \tag{19}$$

and

$$T_{ij}^{\text{operation}} = 2 \times t_i^{\text{load}} + t_j^{\text{op}} \tag{20}$$

The factor of 2 in Equation 20 accounts for both loading (at pickup) and unloading (at delivery) operations.

3) CHARGING TIME ESTIMATION

When a robot requires recharging from energy level E_0 to E_1, the charging time is estimated as:

$$T_i^{\text{charge}}(E_0, E_1) = \frac{E_1 - E_0}{P_{\text{charge}}} \tag{21}$$

where P_{charge} is the charging power rate of the charging station [13].

E. MULTI-OBJECTIVE OPTIMIZATION FUNCTIONS

The problem seeks to simultaneously optimize multiple, potentially conflicting objectives. This multi-objective optimization framework is essential for balancing efficiency, energy conservation, deadline adherence, and workload equity [14].

1) OBJECTIVE 1: MINIMIZE TOTAL TASK COMPLETION TIME (MAKESPAN)

$$f_1(X, \Sigma) = \min \left\{ \max \left(t_{ij}^{\text{end}} \mid i \in \{1, \ldots, n\}, \; j \in \{1, \ldots, m\}, x_{ij} = 1 \right) \right\} \tag{22}$$

This objective minimizes the makespan, defined as the maximum completion time among all tasks, thereby reducing the overall system operational time.

2) OBJECTIVE 2: MINIMIZE TOTAL ENERGY CONSUMPTION

$$f_2(X, \Sigma, Y) = \min \left\{ \sum_{i=1}^{n} \left[\sum_{j=1}^{m} (E_{ij}^{\text{task}} \times x_{ij}) + E_i^{\text{charge}} \right] \right\} \tag{23}$$

where E_i^{charge} represents the total energy consumed by robot r_i for traveling to and from charging stations and during the charging process:

$$E_i^{\text{charge}} = d(p_i^{\text{current}}, c_i) \times bdf_i + \int T_i^{\text{charge}} P_{\text{charge}}\, dt \quad (24)$$

3) OBJECTIVE 3: MINIMIZE WEIGHTED TASK DELAY

$$f_3(X, \Sigma) = \min \left\{ \sum_{j=1}^{m} \delta_j \times \max(0, t_{ij}^{\text{end}} - d_j) \right\} \quad (25)$$

This objective penalizes deadline violations, with penalties weighted by task priorities δ_j, ensuring that critical tasks receive preferential treatment.

4) OBJECTIVE 4: BALANCE WORKLOAD DISTRIBUTION

$$f_4(X, \Sigma) = \min \{\sigma(W_1, W_2, \ldots, W_n)\} \quad (26)$$

where W_i denotes the total workload assigned to robot ri:

$$W_i = \sum_{j=1}^{m} (x_{ij} \times T_{ij}^{\text{task}}) \quad (27)$$

and σ represents the standard deviation operator:

$$\sigma(W_1, \ldots, W_n) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (W_i - \overline{W})^2} \quad (28)$$

where $\overline{W} = \frac{1}{n} \sum_{i=1}^{n} W_i$ representing the mean workload across all robots.

F. SYSTEM CONSTRAINTS

1) TASK ASSIGNMENT CONSTRAINTS

CONSTRAINT C1: SINGLE TASK ALLOCATION

Each task must be assigned to exactly one robot:

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \forall j \in \{1, 2, \ldots, m\} \quad (29)$$

CONSTRAINT C2: SINGLE-TASK ROBOT (ST)

Each robot can execute at most one task at any given time t:

$$\forall i \in \{1, \ldots, n\},\ \forall t \geq 0 : \left| \{j \mid x_{ij} = 1 \wedge t_{ij}^{\text{start}} \leq t \leq t_{ij}^{\text{end}}\} \right| \leq 1 \quad (30)$$

This constraint ensures that the system operates under the Single-Task Robot (ST) paradigm, where robots execute tasks sequentially rather than in parallel

2) ROBOT CAPABILITY CONSTRAINTS

CONSTRAINT C3: PAYLOAD CAPACITY

A robot can only be assigned tasks within its payload capacity:

$$w_j \leq mp_i, \quad \forall i, j \text{ such that } x_{ij} = 1 \quad (31)$$

CONSTRAINT C4: BATTERY FEASIBILITY

A robot must have sufficient battery to complete the assigned task and still reach the nearest charging station:

$$E_i^{\text{curr}}(t_{ij}^{\text{start}}) \geq E_{ij}^{\text{task}} + E_i^{\text{reserve}}, \quad \forall i, j \text{ such that } x_{ij} = 1 \quad (32)$$

where E_i^{reserve} is the minimum energy reserve required:

$$E_i^{\text{reserve}} = \min\{d(l, c) \times bdf_i \mid l \in L, c \in L_{\text{charge}}\} \quad (33)$$

This reserve ensures the robot can reach the nearest charging station from any location in the environment.

3) TEMPORAL CONSTRAINTS

CONSTRAINT C5: TASK DEADLINE ADHERENCE

All tasks must be completed before their specified deadlines:

$$t_{ij}^{\text{end}} \leq d_j, \quad \forall i, j \text{ such that } x_{ij} = 1 \quad (34)$$

CONSTRAINT C6: TASK SEQUENCING

For robot r_i with task sequence $\sigma_i = [\tau_{i1}, \tau_{i2}, \ldots, \tau_{ik}]$, the start time of each subsequent task must occur after the completion of the previous task plus the transit time:

$$t_{i,i(k+1)}^{\text{start}} \geq t_{i,ik}^{\text{end}} + T_i^{\text{transit}}(l_{ik}^{\text{delivery}}, l_{i(k+1)}^{\text{pickup}}) \quad (35)$$

where

$$T_i^{\text{transit}}(l_a, l_b) = \frac{d(l_a, l_b)}{v_i} \quad (36)$$

CONSTRAINT C7: BATTERY RECHARGE SCHEDULING

When a robot's battery level falls below its threshold, it must be scheduled for recharging before it becomes unable to reach a charging station:

$$\text{If } E_i^{\text{curr}}(t) < dbt_i, \text{ then } \exists t' > t : y_i^{\text{charge}}(t') = 1 \wedge t' \leq t + T_i^{\text{max\_reach}} \quad (37)$$

where $T_i^{\text{max\_reach}}$ is the maximum time robot r_i can continue operating before it must begin recharging:

$$T_i^{\text{max\_reach}} = \frac{E_i^{\text{curr}}(t) - E_i^{\text{reserve}}}{bdf_i \times v_i} \quad (38)$$

4) OPERATIONAL SAFETY CONSTRAINTS

CONSTRAINT C8: COLLISION AVOIDANCE

No two robots can occupy the same location simultaneously, maintaining a minimum safe distance:

$$\forall i \neq k,\ \forall t \geq 0 : \|p_i(t) - p_k(t)\|_2 \geq d_{\text{safe}} \quad (39)$$

where d_{safe} is the minimum safe distance between robots and $\| \cdot \|_2$ denotes the Euclidean norm.

CONSTRAINT C9: CHARGING STATION CAPACITY

Each charging station has a limited capacity for simultaneous charging:

$$|\{i \mid y_i^{\text{charge}}(t) = 1 \wedge c_i = c\}| \leq cap_c, \ \forall c \in L_{\text{charge}}, \forall t \geq 0 \tag{40}$$

where $cap_c \in \mathbb{N}^+$ represents the maximum charging capacity of station c.

G. FORMAL PROBLEM FORMULATION

The multi-robot task allocation and coordination problem (MRTACP) with heterogeneous robots can be formally stated as follows:

GIVEN

- A set of n heterogeneous robots $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ with varying capabilities.
- A set of m tasks $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_m\}$ with diverse requirements and deadlines.
- An operational environment $\mathcal{L}$ with known locations and s charging stations.
- Energy consumption model (equations (15)–(17)) and time estimation model (equations (18)–(20)).
- Distance function $d(l_a, l_b)$ for all location pairs $l_a, l_b \in \mathcal{L}$.

FIND

- Task allocation matrix $X = [x_{ij}]_{n\times m}$.
- Task execution sequences $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$.
- Charging schedules:

$$Y = \{y_1^{\text{charge}}(t),\ y_2^{\text{charge}}(t),\ \ldots,\ y_n^{\text{charge}}(t)\}$$

SUCH THAT

- All constraints C_1–C_9 (equations (29)–(40)) are satisfied.
- The multi-objective function $F = [f_1, f_2, f_3, f_4]$ is optimized according to Pareto optimality principles.

OPTIMIZATION PROBLEM:

$$\begin{aligned} \min_{X,\Sigma,Y}\ & F(X,\Sigma,Y) = [f_1(X,\Sigma),\ f_2(X,\Sigma,Y),\\ & f_3(X,\Sigma),\ f_4(X,\Sigma)]\\ & \text{subject to: } C_1,\ C_2,\ C_3,\ C_4,\ C_5,\ C_6,\ C_7,\ C_8,\ C_9 \end{aligned} \tag{41}$$

The solution to this optimization problem provides a Pareto-optimal set of allocations and schedules that balance the competing objectives while satisfying all system constraints.

H. PROBLEM COMPLEXITY ANALYSIS

The MRTACP is computationally intractable for large-scale scenarios due to its NP-hard nature. This complexity arises because the problem generalizes several well-known NP-hard optimization problems.

1) THEOREM 3.1: THE MRTACP IS NP-HARD

Proof: The proof follows by reduction from the vehicle routing problem with time windows (VRPTW), which is known to be NP-hard. When all robots in R are homogeneous ($m_{p_i} = m_p$, $v_i = v$, $bdf_i = bdf$ for all i) and energy constraints are relaxed, the MRTACP reduces to the VRPTW. Since VRPTW is a special case of MRTACP and VRPTW is NP-hard, MRTACP must also be NP-hard [15].

The search space complexity can be analyzed as follows:

a: TASK ALLOCATION COMPLEXITY

The number of ways to assign m tasks to n robots is n^m, giving:

$$\Omega_{\text{allocation}} = n^m \tag{42}$$

b: TASK SEQUENCING COMPLEXITY

For each robot r_i assigned k_i tasks, the number of possible execution orders is $k_i!$. The total sequencing possibilities are:

$$\Omega_{\text{sequencing}} = \prod_{i=1}^{n} k_i! \tag{43}$$

c: TOTAL SEARCH SPACE

The combined search space size is:

$$\Omega_{\text{total}} = \Omega_{\text{allocation}} \times \Omega_{\text{sequencing}} = n^m \times \prod_{i=1}^{n} k_i! \tag{44}$$

In the worst case, where tasks are evenly distributed ($k_i \approx m/n$ for all i), we get:

$$\Omega_{\text{total}} \approx n^m \times \left(\frac{m}{n!}\right)^n = \mathcal{O}((n!)^m \times m!) \tag{45}$$

This exponential growth renders exhaustive search computationally infeasible for realistic problem instances. For example, with $n = 10$ robots and $m = 50$ tasks, the search space exceeds 10^{50} possibilities.

Furthermore, the MRTACP exhibits additional layers of complexity:

- Multi-Objective Complexity: With four competing objectives, the problem demands a Pareto-optimal solution, which is more computationally expensive compared to a single objective optimization problem.
- Dynamic Constraints: The dynamics of the battery state equation (Equation 21) and recharge constraint (Constraint C7) introduce time dependencies, making the search process more complex.
- Heterogeneity Handling: The capability differences among robots lead to $\mathcal{O}(n \times m)$ checks for each robot-task pair during solution evaluation.

Given these complexity characteristics, metaheuristic approaches are essential for obtaining high-quality solutions in polynomial time. This motivates the proposed hybrid algorithm combining the SVCA and the NSGA-II, which will be detailed in the following chapters.

I. DYNAMIC TASK REALLOCATION FORMULATION

Unlike static task allocation approaches, this work incorporates dynamic reallocation capabilities to enhance system adaptability and responsiveness. When a robot r_i completes its assigned task sequence σ_i and returns to its parking station, it can detect and autonomously execute nearby pending tasks before parking.

1) TASK DETECTION MECHANISM

Upon task completion at time t_{complete}, robot r_i scans for nearby pending tasks within a detection radius R_{detect}:

$$T_{\text{detect}}(i, t) = \{\tau_j \in T_{\text{wa}} \mid \|p_i(t) - l_j^{\text{pickup}}\|_2 \leq R_{\text{detect}}\} \quad (46)$$

2) FEASIBILITY EVALUATION

For each detected task $\tau_j \in T_{\text{detect}}(i, t)$, the robot evaluates feasibility based on multiple criteria.

BATTERY FEASIBILITY:

$$E_i^{\text{curr}}(t) \geq E_{ij}^{\text{task}} + E_i^{\text{reserve}} + E_i^{\text{return}} \quad (47)$$

where

$$E_i^{\text{return}} = d(l_j^{\text{delivery}}, p_i^{\text{init}}) \times bdf_i$$

is the energy required for r_i to return to its parking station after task completion.

DEADLINE FEASIBILITY:

$$t + T_{ij}^{\text{task}} \leq d_j \quad (48)$$

PAYLOAD FEASIBILITY:

$$w_j \leq m_{p_i} \quad (49)$$

3) REALLOCATION EXECUTION

Upon selecting τ_{selected}, the robot updates the system state as follows:

$$T_{\text{wa}} \leftarrow T_{\text{wa}} \setminus \{\tau_{\text{selected}}\} \quad (50)$$

$$T_{\text{a}} \leftarrow T_{\text{a}} \cup \{\tau_{\text{selected}}\} \quad (51)$$

$$x_{i,\text{selected}} \leftarrow 1 \quad (52)$$

This dynamic reallocation mechanism enhances system efficiency by opportunistically utilizing available robot capacity without waiting for the next centralized scheduling cycle.

IV. SYSTEM ARCHITECTURE AND MODELING

The overall architecture has been divided into three main entities: the users, coordinator, and robots. Each module performs its specific functions to facilitate the overall operation of the system. The relationships between these modules are depicted in Figure 1 below. The figure presents how data is passed in the system between the different entities. The arrows in the diagram demonstrate which component can transmit or receive data from another component. For instance, the coordinator is able to both transmit and receive data from the robots, enabling real-time synchronization and efficient coordination across the entire system.

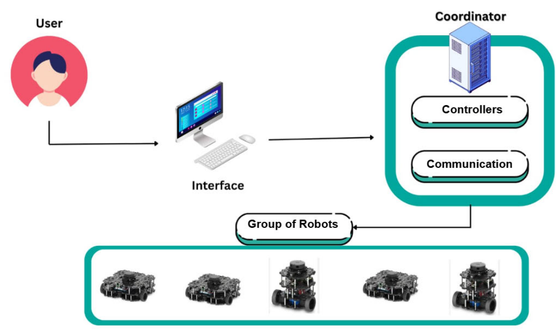


FIGURE 1. Overall architecture of the system consisting of user, interface, coordinator, group of robots.

The overall architecture consists of four main components User, Interface, Coordinator, and a Group of Robots that collaboratively enable efficient task scheduling and coordination within the multi-robot system.

A. USER

The user is the operator or supervisor of the system and he is the one who provides high level inputs like mission parameters, operation constraints or task requests. The interface with the system allows the user to observe the real-time actions of the robots, report progress on the currently being performed tasks, and will send them notifications regarding the finished task or system warnings. The role of the user is mainly supervisory where operation objectives and priorities are established to be used in the coordination system.

B. INTERFACE

The interface acts as the communication tool between the system and the user. It gives an interactive platform which can be a computer terminal or a control panel in which commands are typed and the information displayed. The interface supports two-way data exchange: it will take as input commands or parameters used to set up the system and give back system feedback, robot status and performance measurements in a clear, user-friendly format.

C. COORDINATOR

The coordinator serves as the core decision-making unit of the system. It manages task scheduling, robot assignment, and communication control. Within the coordinator, two main modules are present: controllers and communication.

- The Controllers module deals with the task allocation algorithm processing, robot state management, and task execution optimization according to constraints such as deadlines, weight capacity, and energy usage.

This module guarantees that each robot is allocated appropriate tasks based on their capabilities [2].

- The Communication module deals with the data exchange between the coordinator and the robot system. This module guarantees data synchronization and real-time status updates between all components of the system.

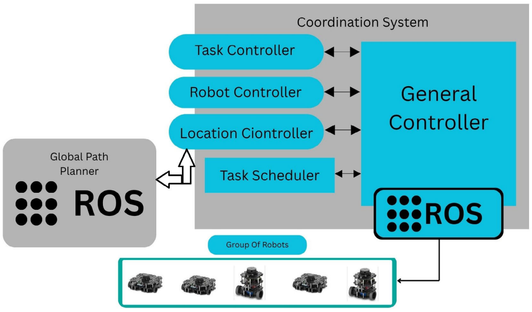


FIGURE 2. **Interconnection diagram of the Centralized coordination system. The components shown in blue and gray represent the parts developed in this study, while the white elements correspond to the ROS navigation stack.**

Figure 2 shows the architectural structure of the centralized coordination system, focusing on the connections between the internal and external coordination components. The coordination system is organized into five key components that work together synergistically to handle task scheduling, allocation, and processing. The general controller is the initialization and control center that handles information flow among all controllers and starts their respective threads during system initialization. The task controller periodically searches for tasks requiring scheduling or cancellation, maintains efficiency by setting upper and lower limits on the number of tasks and robots per scheduling cycle, and monitors task execution through countdown timers that track expected status updates. The robot controller monitors robot status, searches for available robots during allocation, maintains a list of all system robots with countdown timers for status updates, and detects failures when robots become unresponsive. The location controller maintains lists of locations and distance tables used during task allocation, updating these dynamically when locations are added or removed using the global path planner [16]. Finally, the task scheduler represents the optimization core that receives task lists from the task controller, robot lists from the robot controller, and distance tables from the Location Controller, then executes the optimization process to generate scheduled task lists and identify any non-schedulable tasks.

The existing works uses database as a central repository that stores information about the environment, robots, tasks, and system parameters, with bidirectional communication paths connecting it to the various controllers. However, in contrast to these existing works, our proposed system eliminates the dependency on a centralized database block. Instead of relying on database queries for real-time coordination, our architecture leverages direct communication channels and in-memory data structures maintained within the coordinator, reducing latency, eliminating potential database bottlenecks, and enhancing system responsiveness. This modification is particularly beneficial for real-time multi-robot coordination where millisecond-level decision-making is critical and database access overhead could compromise system performance [17].

D. GROUP OF ROBOTS

This element is a group of mobile robots performing the given logistic or operational duties. All robots work independently on navigation and performing their tasks, yet react to the signals that the coordinator sends to them. The system enables dynamically adapted modifications to the robots as they always report their condition like task completion, location, or charge [18]. After a task is done by a robot it also reports back to the coordinator and in case it is required to move on to new tasks it does so or goes back to a parking or charging station according to its energy level or availability.

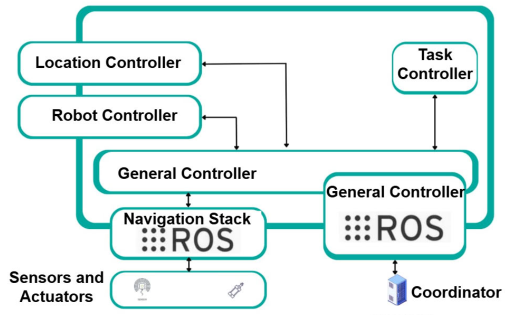


FIGURE 3. **Interconnection diagram of the internal robot system. The sections highlighted in blue and gray indicate the components developed in this work, while the white portions represent the Robotic Operating System (ROS) navigation stack.**

Figure 3 presents the detailed architecture of the internal robot system, depicting how individual robots interface with the coordination system and manage their autonomous operations. The system is organized into three functional layers supported by the ROS navigation stack. At the core, the general controller serves as the main coordination hub, receiving task queues through the communication module, forwarding tasks to the task controller, managing navigation by sending destination coordinates to the ROS navigation stack, and handling all external communication with the central coordinator. The task controller maintains the robot's task queue, adds or removes tasks based on coordinator requests, removes completed tasks from the queue, and forwards the next task to the general controller for execution [19]. The robot controller manages all proprioceptive information including battery level, minimum average speed, current location, and assigned parking/recharging station. It reads robot parameters (minimum average speed, battery discharge factor, battery threshold, payload maximum loading/unloading time, and parking station) from the database during initialization, updates minimum average speed after each task completion, and verifies task feasibility considering time and battery requirements before execution. The Location controller maintains and updates the list of locations and distance table, responding when the robot needs to navigate to unknown locations or parking stations by querying updated information.

The ROS Navigation Stack, shown in white in the figure, handles all autonomous navigation functions including path planning, motion control, and obstacle avoidance. The system employs dijkstra for global path planning, dynamic window approach (DWA) for local path planning, and adaptive monte carlo localization (AMCL) for robot localization. Comparing with the existing architectures, this work considers database as an external component that stores robot parameters, locations, and distance information. In our proposed work, this database dependency is eliminated. Instead of querying a database for location information, distance tables, or robot parameters, our system maintains this information within distributed data structures across the robot's internal memory and the coordinator's shared state space. This architectural change reduces communication overhead, eliminates single points of failure associated with centralized databases, and enables faster decision-making at the robot level. Furthermore, it allows for more dynamic operation where robots can immediately detect and execute nearby pending tasks upon returning to parking stations, without waiting for database synchronization or query responses, thereby enhancing the overall system's adaptability and real-time performance in complex logistic scenarios [20].

V. PROPOSED HYBRID OPTIMIZATION ALGORITHM

A. OVERVIEW OF THE HYBRID APPROACH

The proposed hybrid optimization algorithm, i.e., Figure 4, is a hybridization of the SVCA and NSGA-II to produce efficient and strong assignment of tasks to multi-robotic systems in heterogeneous robots. The process is structured into four primary phases that implement one another and is monitored and adapted dynamically over-the-course.

1) INITIALIZATION PHASE

The algorithm starts with the initiation of ROS2 multi-robot controller node which acts as the main coordinating point of the complete system. At that time, the environment setup is loaded which contains the map of the warehouse with the location of shelves, the delivery goal position, charging stations, and robot parking spots. The system subsequently starts N diverse robots, each with its own peculiarities, e.g., battery capacity, discharge rate, maximum payload capacity, average speed in navigation, and position. At the same time, M logistic tasks are created or accepted by an external task management system and each task has a pickup point (shelf), delivery point (goal), deadline constraint and payload requirement. After the initiation of the physical and logical part the system sets up the ROS2 communication

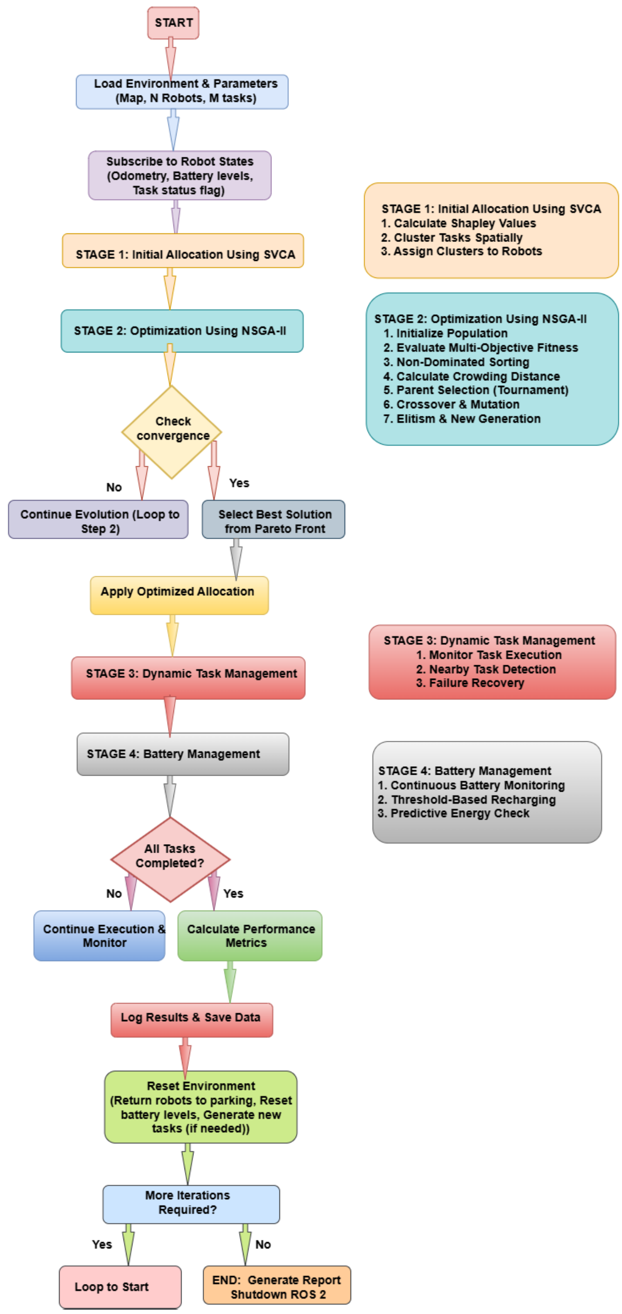


FIGURE 4. Flow of the complete hybrid optimization algorithm working with detailed blocks.

infrastructure by subscribing to the most important robot state topics. The subscribers of odometry, battery, and custom tasks keep information about the actual position of each robot (subscribers of odometry), energy status (subscribers of battery), and task completion status (pending, assigned, in-progress, completed or failed). It underlies this continuous state monitoring to provide adaptive decision-making during the execution of the algorithmic procedure itself [21].

2) INITIAL TASK ALLOCATION USING SVCA

In the first phase of optimization, the SVCA is used to rapidly and effectively produce an initial distribution of tasks.

This method relies on game theory of cooperation, in which robots and tasks are considered as players in a coalition game. The calculation of shapley values for every robot and task is the first step of the algorithm. For robots, the computation of contributing scores takes place by considering the positive aspects such as battery level and navigation speed but also subtract by applying usage penalties that prevent specific robots from being overutilized. Requirement scores are assigned to tasks based on their priority and anticipated completion time, whereby the less demanding tasks are given higher scores.

The individual shapley values calculated for each robot are then applied to assess the robot-task coalitions, where a coalition is defined as the possible combination of robots and tasks. The coalition value function takes into account the total contributions of robots and tasks, it has a penalty for the excessive average distances between robots and assigned tasks, and it requires that the size of the coalition be such that it is still computationally efficient. The algorithm uses monte carlo sampling of coalition orderings to efficiently approximate each player's expected marginal contribution to various coalitions. Once the Shapley values are determined, the algorithm moves to the next step of spatial clustering of tasks via $K-$means, where K is the number of robots available. This clustering method brings together tasks that are close to each other geographically thus it is going to be the case that the inter-task travel distances are minimized when the robots are executing their assigned sequences. The created clusters are then allocated to robots by a matching process that connects high-value robots (those with better capabilities) with demanding task clusters (those requiring more resources), while simultaneously distributing the overall workload across the robot team. The initial allocation matrix X_0 is then generated which indicates the tasks assigned to the robots [5].

The SVCA stage is computationally efficient, providing a good initial solution in polynomial time, which serves two purposes: it allows robots to begin task execution immediately, and it provides a high-quality starting point for the subsequent NSGA-II optimization stage.

3) OPTIMIZATION REFINEMENT USING NSGA-II

While robots execute tasks based on the SVCA allocation, the NSGA-II algorithm runs asynchronously to refine the allocation through evolutionary optimization. Unlike single-objective optimization methods, NSGA-II simultaneously optimizes multiple conflicting objectives without requiring predetermined weights, making it ideal for multi-robot coordination where trade-offs between distance, time, energy, and load balance must be considered. The optimization begins by initializing a population of candidate solutions, where each individual represents a complete task allocation encoded as an assignment matrix and task execution sequences for each robot. The SVCA solution X_0 is included in the initial population to preserve its quality, while additional random valid allocations introduce diversity. Each solution is certified in order to guarantee it. respects constraints of payload (task requirements are no more than). robot capabilities), task differentiation (assigned task is unique) once), and battery viability (enough energy to do what is assigned). tasks). Each solution is then compared to the algorithm. four goals: reducing the total distance covered by all. minimizing makespan or maximum completion (f1), robots. time, all robots (f2) and reduces the energy consumed. Load balance variance to minimize, and allocate load balance. work equitably (f4). The violation of constraints including They are fed by battery thresholds, task deadlines or payload limits. violent punishment that impairs their fitness, drives evolution. to workable distributions.

Non-dominated sorting is a major innovation of NSGA-II. which can put solutions into pareto fronts in terms of domination. relationships. A prevails over B in case there is a solution A. much better than B in at least one objective and not as bad as it is in any objective. All non-dominated are in the first Pareto front F1. solutions- these are the most optimal trades that can be made. Further fronts F2, F3 and on are progressively weaker. dominated solutions. Several highquality is maintained in this ranking. instead of representing trade-offs, solutions. imposing 1 optimal solution. In every Pareto front, the distance of crowding is computed to measure solution density. Solutions in sparse regions are made to get. larger crowding distances, which facilitates variety and eliminates. the algorithm narrows down to a small area of the. solution space. This is a diversity maintenance which is essential to exploration. various allocation plans and shunning premature. narrowing down to inefficient solutions. Binary tournament is used in parent selection of reproduction. selection, two random solutions are selected and the better one (according to Pareto rank and crowding distance) is selected for mating. Crossover is experienced with the selected parents. (mutating sequence pairs of tasks in two parent solutions) and mutation (random choice of tasks to do or in what order) to produce descendants solutions. Invalid offspring are patched using a constraint-checking system that replaces. discomforting tasks or reallocate tasks to meet constraints. The algorithm makes use of elitism combining the current. population containing newly generated offspring, and imposing nondominated. sorting into the combined pool, and picking the optimal pop size solutions on the next generation. This ensures that good solutions are never wasted in the process of introducing continuously. new variations. The evolution process goes on. until convergence criteria, i.e. maximum generations reached, and little betterment observed after consecutive. generations, or a time-restraint overthrown. On convergence, the algorithm removes the last pareto. front and picks the most fit solution with a decision-making. strategy. This could include weighted sum methods (combining. knee point selection, objective weights that are defined by the user), objective with user-defined weights Trying to find the solution that is further far away as possible. user preference (as an ideal point), or an interactive one. The optimized assignments of robot tasks are then updated with allocation, and robots rearrange their navigation strategies [10].

4) DYNAMIC TASK MANAGEMENT

During the execution phase, the system always keeps a close watch on the task progress by receiving feedback from the robot navigation actions and custom status messages. The detection of task failures (robot unable to reach destination, navigation timer expired, path blocked by an obstacle) and execution delays (robot working slower than the schedule due to unexpected conditions) is made possible by this monitoring system.

The dynamic detection of nearby tasks is a major innovative feature of the proposed system. In case a robot has finished its task and is getting ready to go back to the parking station, the system will first look for any pending tasks within a certain distance (for instance, 6m) and then if it finds nothing in that range, it will take the entire area as a potential search space and use Euclidean distance to find the nearest job. If the system finds any pending task nearby, it will instantly reassign the robot thereby letting it do productive work, and the robot won't have to make unnecessary trips to parking. The proposed task allocation shows a substantial increase in system efficiency without the need for a complete re-optimization.

The system executes failure recovery plans as soon as task failures are identified. The failed tasks are put in pending status and are either instantly assigned to another available robot (if there is one) or put in the queue for the next optimization cycle. In case of critical failures affecting a group of tasks (e.g. a robot breakdown), the system can activate a partial re-optimization using NSGA-II over the affected tasks and available robots, keeping the allocation quality while reducing the computational load.

5) BATTERY MANAGEMENT

Energy management is the most important aspect for the continuous running of autonomous robot teams. The system carries out uninterrupted batteries' watching through subscribing to every robot's battery state topic, keeping track of the current charge level, and calculating the rate of discharge based on the recent patterns of power consumption. This monitoring in real-time makes it possible to use both reactive and predictive energy management strategies.

Prompt recharge based on thresholds occurs when the battery of a robot is charged to the level that is below the predetermined level (e.g., 20%). After low battery is detected, the system directs the robot to go the closest available charging station, while the hanging or reassigning of any pending tasks in its queue to other robots is done. The algorithm for choosing a charging station takes into account both distance and current occupancy so that queuing delays in cases of multiple charging stations being available are avoided.

Predictive power checking is an extra safety layer that comes with estimating the power needed for a robot's next duty before assigning it. The estimation considers the distance to travel to the pickup location, the weight of the payload which will change the efficiency of the robot during transport, and the distance to travel to the delivery location. If the robot's present battery level minus the estimated requirement is below the safety threshold, preemptive charging is triggered even if the battery is technically above the threshold. This eliminates the occurrence of situations where a robot commences with a task but exhausts its power midway through the execution.

The battery management system also coordinates multi-robot charging to prevent all robots from simultaneously attempting to recharge, which would halt system productivity. By maintaining awareness of which robots are charging, which are approaching low battery, and which charging stations are occupied, the coordinator can stagger recharging schedules to maintain continuous operational capacity [22].

6) COMPLETION, LOGGING, AND ITERATION

The system is in constant alert of monitoring the completion of tasks, transition. to performance evaluation after all the work is done. The important measures are total distance traveled, makespan. energy usage, load balancing, constraint violation, and runtime (SVCA and NSGA-II). Data, including Definition of movements of a robot, battery energy, work execution, with time. pareto front solutions of NSGA-II, details, and NSGA-II Pareto front solutions are stored under CSV. or tabular format to analyze and visualized with RViz. For re-ExecutionContext after several cycles, the environment will be restored robots will get back to parking. battery statuses and task statuses are recreated, and newtasks. are specified–facilitating statistical analysis of situations. [23]. Upon completion of the last version, a conclusive report sums up. all operates on averages, deviations, and worst/ best-case. results. GRaceful shutdown of ROS 2 nodes occur. resources. This is a workflow that offers centralized solution. allocation of tasks, multi-robot coordination. implementation, malfunction response and power management [24].

B. SHAPLEY VALUE CLUSTERING ALGORITHM

The SVCA begins by combining all robots and tasks into a set $N = R \cup T$ and initializing a counter $s = 1$ and parameter $q = q_{\text{ini}}$. It estimates the Shapley values $\phi_i(N, J)$ for each robot and task, measuring their average contribution to all coalitions. Starting with $N_c = |R|$ clusters, each robot initially forms its own cluster. The algorithm then iteratively pairs tasks with robots: the task with the lowest Shapley value is assigned to the robot with the highest value, forming a cluster with combined Shapley value $\phi_a = \phi_i + \phi_j$. Assigned robots and tasks are removed, and the process continues until all tasks are clustered. Next, if the number of robot clusters exceeds N_c, robots with the lowest Shapley values are merged, summing their values $\phi_b = \phi_i + \phi_j$, until the desired

TABLE 2. Algorithm 1: Shapley value clustering algorithm (SVCA).

Input	Set of robots R. Set of tasks T. Initial parameters $q = q_{\text{ini}}$.
Output	Optimal robot–task clustering configuration C_s.

Algorithm Steps:
1. Let $N = R \cup T$.
2. Initialize $s = 1$ and $q = q_{\text{ini}}$.
3. Compute the estimated Shapley values $\phi_i(N, J)$.
4. Initialize $N_c = R$ clusters, each containing one robot.
5. **While** $N_c \geq 1$ **do**
   a. **While** $T > 0$ **do**
      i. Select task $i \in T$ with the lowest $|\phi_i(N, J)|$.
      ii. Assign it to the cluster containing robot $j \in R_j$ with the highest $|\phi_j(N, J)|$.
      iii. Form a new mixed robot–task player $a = \{i\} \cup \{j\}$.
      iv. Compute the estimated Shapley value: $\phi_a = \phi_i(N, J) + \phi_j(N, J)$.
      v. Remove old players i and j from their respective sets.
   b. **End inner while.**
   c. Save the clusters formed as possible solution C_s.
   d. Increment $s = s + 1$.
   e. Reset R and T to their original sets.
   f. Decrement $N_c = N_c - 1$.
   g. **While** $|R| > N_c$ **do**
      i. Select players $i, j \in R$ with the two lowest $|\phi(N, J)|$ values.
      ii. Form a new robot–robot cluster $b = \{i\} \cup \{j\}$.
      iii. Compute $\phi_b = \phi_i(N, J) + \phi_j(N, J)$.
      iv. Remove players i and j from R.
   h. **End while.**
6. **End outer while.**
7. Compute $V(C_s)$ for all possible clusterings C_s, $s \in [1, N_c]$.
8. Select the clustering C_s with the highest $V(C_s)$ as the final solution.

TABLE 3. Algorithm 2: Non-dominated sorting genetic Algorithm II (NSGA-II).

Input	Population size P, maximum generations $G_{\max}$ Crossover probability P_c, mutation probability P_m Objective functions $f_1, f_2, \ldots, f_k$
Output	Final Pareto-optimal set of MRTA solutions

Algorithm Steps:
1. **Initialization:** Generate an initial population P_0 of size N randomly.
2. **Fitness Evaluation:** Evaluate each individual in P_0 using the objective functions.
3. **Non-Dominated Sorting:** Classify individuals into different fronts $F_1, F_2, \ldots, F_m$ based on dominance rank.
4. **Crowding Distance Calculation:** For each front, compute the crowding distance to preserve diversity among solutions.
5. **Selection:** Apply binary tournament selection based on rank and crowding distance.
6. **Crossover and Mutation:** Generate offspring population Q_t using crossover (probability P_c) and mutation (probability P_m).
7. **Combine Populations:** Merge parent and offspring populations: $R_t = P_t \cup Q_t$.
8. **Sorting and Truncation:** Perform non-dominated sorting on R_t and create a new population P_{t+1} by selecting the best individuals (based on rank and crowding distance) until size N is reached.
9. **Loop:** Repeat Steps 2–8 for all generations until $t = G_{\max}$.
10. **Output:** Return the final set of non-dominated (Pareto-optimal) MRTA solutions representing the trade-off between objectives such as makespan, computational time, and allocation efficiency.

number of clusters is reached. This procedure repeats for all N_c from $|R|$ to 1, generating multiple clustering solutions C_s, which are evaluated using $V(C_s)$. The configuration with the highest $V(C_s)$ is selected as the optimal robot–task clustering based on Shapley value contributions.

C. NON-DOMINATED SORTING GENTIC ALGORITHM

The proposed hybrid MRTA framework (SVAC+NSGA-II) encodes task–robot allocations as chromosomes and employs problem-aware crossover to preserve task–robot consistency, along with adaptive mutation guided by population diversity. Constraint handling mechanisms, including repair or penalization strategies, ensure infeasible solutions are addressed. In each generation, the parent and offspring populations are combined and sorted using NSGA-II with elitism; the next generation is filled front by front, with crowding distance used to maintain solution diversity. This structured approach enables fast convergence within 30 generations while balancing exploration and exploitation. The algorithm outputs a set of non-dominated MRTA solutions forming the Pareto front, from which decision-makers can select allocations that optimally trade off energy consumption, deadlines, and payload requirements. The approach is dynamic, scalable, and preserves diversity throughout the search process [25].

D. ILLUSTRATIVE WALKTHROUGH OF ONE COMPLETE NSGA-II GENERATION

To ensure full reproducibility of the pareto-based NSGA-II within the five-robot, fifteen-task MRTA scenario, one complete generation is illustrated through a representative individual. Each chromosome consists of two coupled components: (i) an allocation vector $A = [\tau_1, \ldots, \tau_{15}]$, where $r_j \in \{r_1, \ldots, r_5\}$ assigns task τ_j to one of the five heterogeneous robots, and (ii) a sequence vector S, which defines the execution priority of tasks within each robot's queue (including the results from the SVCA solution also). For example, a candidate solution D_1 may encode $A = [r_2, r_1, r_3, r_5, r_4, r_2, r_1, r_3, r_5, r_4, r_2, r_1, r_3, r_5, r_4]$, resulting in distributed task queues such as $r_1 = \{\tau_2, \tau_7, \tau_{12}\}$, $r_2 = \{\tau_1, \tau_6, \tau_{11}\}$, etc., with execution order derived from S. The solution quality is evaluated using the four objective formulations defined in Section III-E: (1) makespan as the maximum completion time across the five robots computed via cumulative travel and operation times; (2) total energy consumption derived from motion energy, task operation energy, and battery discharge modeling; (3) weighted deadline violation based on priority-scaled lateness; and (4) workload balance quantified as the standard deviation of robot workloads. Suppose D_1 produces completion times [420, 465, 390, 440, 410] s, yielding $f_1 = 465$ s, total energy $f_2 = 182.6$ Wh, zero deadline violation $f_3 = 0$, and workload deviation $f_4 = 26.3$. After fitness computation for all individuals, non-dominated sorting assigns Pareto ranks and crowding distances. Parent chromosomes are selected via binary tournament, and a problem-aware one-point crossover exchanges allocation genes beyond a cut index (e.g., after τ_8) while preserving task uniqueness; sequence indices are then recomputed per robot queue. Adaptive mutation randomly reassigns a selected task (e.g., τ_{10}) to another robot with probability P_m, increasing exploration when diversity decreases. Each offspring undergoes infeasibility detection: payload constraint (task weight $\leq$ robot capacity), battery sufficiency (available energy $\geq$ task energy + reserve), and deadline feasibility are verified. Violations trigger repair through reassignment or resequencing; if infeasible after repair, a large penalty is added to objective values to prevent Pareto dominance. Finally, elitist environmental selection merges parent and offspring populations, re-sorts by rank and crowding distance, and truncates to maintain constant population size, thereby completing one fully defined and reproducible NSGA-II generation within the 5-robot, 15-task framework.

E. INTEGRATION STRATEGY

The hybrid algorithm that has been suggested marries the SVCA and NSGA-II together in the form of a two-phase sequential optimization framework. While in the first phase, SVCA employs game theory in a cooperative manner to put the large-scale MRTA problem into clusters of smaller and easier-to-manage ones. These clusters are formed by ordering robots and tasks according to Shapley values that reflect their expected marginal contributions to coalition performance, thus, these robots and tasks have very low contribution to the performance of the coalition. However, this process of clustering makes it possible to pair high-capability robots with demanding tasks and as a result, the computational complexity is reduced from exponential to polynomial time. The clusters are then formed containing robot-task pairs that are compatible and forwarded to the second phase where NSGA-II simultaneously performs multi-objective optimization in each cluster. NSGA-II applies non-dominated sorting and crowding distance mechanisms to create Pareto-optimal solutions that minimize both makespan and total computation time while concurrently satisfying constraints on task deadlines, robot payload capacities, and battery levels. This incorporation strategy merges the global task distribution intelligence of SVCA with the local optimization strength of NSGA-II and thus the system will be able to manage heterogeneous robot teams with different capabilities efficiently without sacrificing solution quality even in large-scale, dynamic logistics scenarios. The coordinator is always aware of task execution and can activate rescheduling with the same hybrid method if new tasks arrive or when there are failures, thus, the system operation is adaptive and robust.

F. DYNAMIC TASK REALLOCATION MECHANISM

The hybrid algorithm introduced here includes a dynamic task reallocation approach which allows for real-time adjustments and resource usage in an efficient manner, according to the situation in the operational environment and the unforeseen events that might occur during the execution of the task. When a robot has completed its task successfully and has returned to its charging station, the system will, without any delay, check for unallocated tasks that are pending in the area where the robot is and will judge whether the robot is able to do these tasks considering its current battery level, the load it can carry, and the time limit of the task. If there are nearby tasks that are suitable, the coordinator skips the full rescheduling procedure and assigns the task to the robot using a direct greedy nearest-neighbor strategy, hence minimizing idle time and reducing overall system makespan. When robot failures occur during a task, the system identifies the failure through regular status monitoring and promptly activates the hybrid SVCA+NSGA-II rescheduling algorithm to redistribute the remaining tasks of the non-operational robot among the operational robots while still observing priority constraints and optimizing for the least interruptions to the current schedule. Moreover, if the robotic unit's battery reaches a level below the set limit while it is still functioning, the management system will at once reroute the robot to the closest recharging point and switch its uncompleted tasks over to other robots that have enough energy left, thus making it possible to complete the work uninterrupted and without any manual help. This changing of the task assignments is further supported by the fact that the system can accommodate new tasks being introduced while the execution is in progress: when the queue of tasks gets longer, the coordinator calls on the hybrid method to come up with a new timetable that comprises both the tasks already in progress and the new ones, and at the same time takes into account the positions of the robots, their battery levels, and the sequence of tasks they are committed to, thereby keeping the system efficient and responsive in real-world, time-varying logistics situations.

VI. IMPLEMENTATION DETAILS

This chapter details the implementation framework of the proposed multi-robot coordination system, describing the integration of ROS2 as the primary middleware, Gazebo as the simulation platform, and the supporting system components that enable reliable inter-robot communication and coordination. The design emphasizes architectural choices, communication mechanisms, and visualization tools that collectively ensure robustness and scalability.

The framework uses the ROS2 environment because it offers better real-time capabilities, security features, as well as scalability for the case where more than a single robot is employed in the system, as opposed to the older ROS1. These capabilities allow the communication between the different robots in the system through the "publish-subscribe" service in the ROS2 environment [26]. For the experiments performed in the project, the work employed five open-source robots referred to as the turtleBot3 robots. These robots include three Burger versions as well as two waffle pi versions.

Within each robot namespace, key nodes operate cooperatively, including the `differential_drive_controller` for wheel velocity control and odometry feedback, the `robot_state_publisher` for transformation broadcasting, the `p3d_base_controller` for physics-based motion integration in Gazebo, and static transform publishers for maintaining coordinate frame consistency. All robot nodes are interconnected via the central `/tf` transform tree, supported by infrastructure nodes such as `/gazebo`, `/lifecycle_manager`, `/map_server`, and `/bond`. The `/transform_listener_impl` node aggregates transformation data across robots, providing a unified global view required for collision-free planning and task allocation [27].

This architecture enables fully autonomous low-level robot control while maintaining system-level coordination through shared global information, effectively realizing a hybrid centralized–distributed paradigm suitable for scalable multi-robot logistics operations.

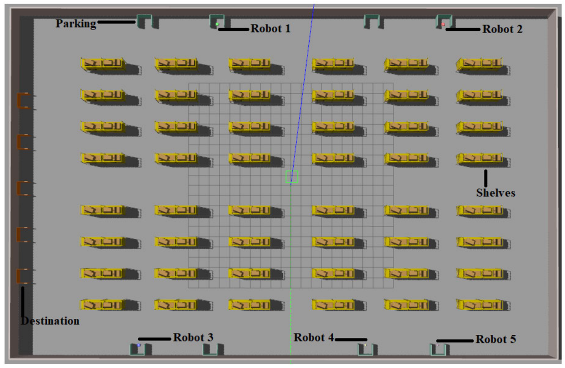


FIGURE 5. Custom gazebo environment created with 48 shelves and five robots.

A. GAZEBO SIMULATION ENVIRONMENT

The Gazebo simulation environment is a physics based simulators of the proposed multirobot. coordination system before implementation in the real world. Gazebo has been chosen due to strong open dynamics. realistic sensor modeling, engine (ODE) physics engine, seamless ROS2, and precision collision support. detection, friction modeling and actuator dynamics. This setup allows the quick prototyping of a system and controlled performance testing, and provides safe testing of impractical scenarios. or hazardous on bodily robots, like battery wear, robots. high-density traffic conditions, failures, and high-density traffic conditions. Figure 5 represents the warehouse logistics space, which covers the world. 1,500 m 2 (30 m × 50 m) and has grid-organized storage. racks that create narrow parallel aisles that add to navigation. and complexity of collision avoidance. A grid-patterned floor supports visual odometry, scale reference, whereas workspaces are marked by enclosing walls. Pickup and delivery stations are situated around the perimeter, and darker. areas that are used to depict loading docks and staging areas. The environment has five turtlebot3 mobile of type differential-drive. robots (3 turtlebot3 waffle robots and 2 tutrtlebot3 burger) No robots) have simulated LiDAR, IMU and wheel. encoders. An active navigation is verified by a highlighted robot trajectory. Such a structure allows thorough assessment. of the hybrid SVCA+NSGA-II algorithm with different task. densities, sizes of teams, heterogeneous payloads, and dynamic. task arrivals, authenticating system scalable, efficacy and robustness [27].

B. ROS2 FRAMEWORK INTEGRATION

The ROS visualization (RViz) interface corresponding to the simulation is shown in Figure 6 and serves as the primary tool for real-time monitoring, visualization, and debugging of the multi-robot system. Figure 6 illustrates the different timing RViz plot and the corresponding python plot created using the coordinate odometry data. Figure 6(a) shows the time T= 5 minutes plots consisting of the RViz plot and python plot. Figure 6(b) depicts the plots at T= 10 minutes, while Figure 6(c) plots are at T= 20 minutes. Finally, Figure 6(d) demonstrates the T= 35 minutes stage. The proposed framework explicitly incorporates collision-free global path planning and local dynamic collision avoidance in multi-robot scenarios. Within the ROS2–Nav2 stack, each robot performs global planning independently while sharing environment information, and at the local planning level, other robots are modeled as dynamic obstacles, enabling real-time avoidance and safe navigation. This ensures coordinated multi-robot operation without inter-robot collisions [28]. In the supplimentary materials added the complete ROS Qt Tool (RQT) Graph (Figure A) for reference. Figure 7 gives the total completed paths for each robots and video link is attached therewith for reference.

C. SYSTEM COMPONENTS AND MODULES

The implemented multi-robot coordination system follows a modular ROS2-based architecture. At the top level, a centralized Coordinator module, implemented as a Python ROS2 node, hosts the hybrid SVCA+NSGA-II task allocation algorithm, manages the global task queue, monitors robot states (position, battery, payload, availability), and performs scheduling and rescheduling in response to dynamic events. Each robot is equipped with a local control module comprising the Nav2 navigation stack for global planning (dijkstra/A*), local obstacle avoidance (DWA), and localization (AMCL), along with a task execution manager that validates and executes assigned tasks and reports status updates [29], [30], [31]. A Battery management module continuously monitors energy levels and triggers task reallocation and autonomous charging when thresholds are reached. Supporting infrastructure includes a SQLite-based Database module for persistent storage of robot, task, environment, and performance data, and a Communication module built on ROS2 DDS middleware for reliable, low-latency message exchange. System monitoring and visualization are provided through RViz and RQT for real-time introspection and performance analysis [32], [33].

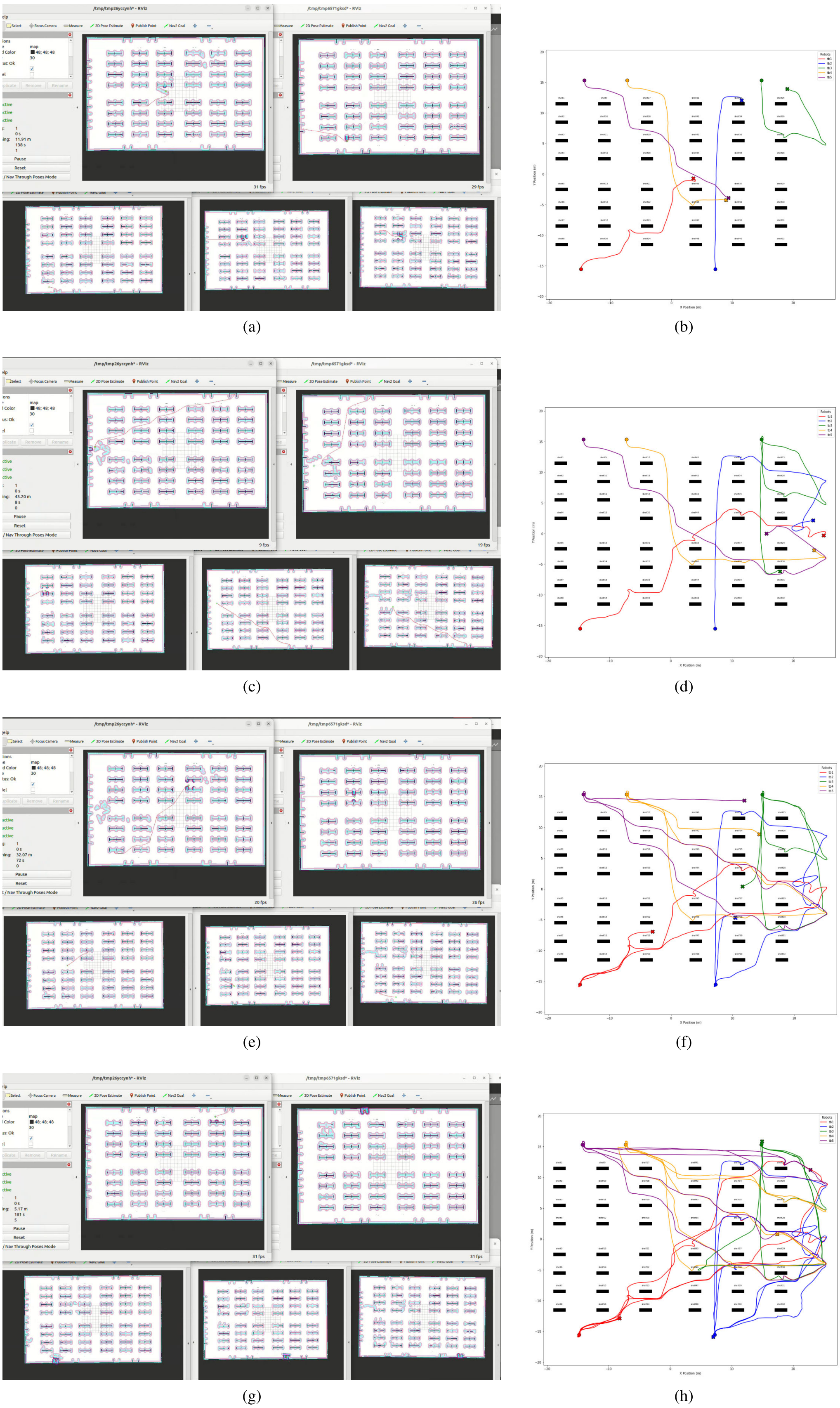


FIGURE 6. RViz and path visualization:(a),(b) at T=5 min,(c),(d) at T=10 min,(e),(f) at T=20 min,(g),(h) at T=35 min.

D. COMMUNICATION PROTOCOLS

Effective communication is the basis of such a multi-robot coordination system, which ensures regular data interchange between the centralized coordinator and the distributed robot agents. It follows a hybrid communication strategy whereby the publish–subscribe model of ROS2 supports continuous streams of data with service-based request–response mechanisms for critical command and control. This way, latency will be minimal, data will be coherent, and the system will be resistant to network interruption.

1) ROBOT STATE PUBLISHING

Each robot continuously publishes its state information at 10 Hz, allowing the coordinator to maintain an up-to-date global view. Published data include pose (x, y, θ), linear and angular velocity, battery level, payload status, and task execution state (idle, navigating, executing, or charging). These messages follow a custom `RobotState.msg` format and are transmitted over robot-specific topics (`/robot_X/state`), to which the coordinator subscribes for scheduling and monitoring purposes [34], [35].

2) TASK ASSIGNMENT PROTOCOL

Task allocation is performed with a service-based protocol that allows for guaranteed acknowledgment. The coordinator requests task assignments through robot-specific services, (`/robot_X/assign_task`), with the task ID, the locations of pickup and delivery, the payload requirements, an estimated travel time, and a deadline. The robots respond with either accept or reject, after checking for feasibility with respect to given battery and payload constraints; denied tasks can be immediately reassigned, if needed.

3) TASK STATUS UPDATES

During the course of task execution, robots periodically and asynchronously send updates at significant points like the beginning of the task, completion of picking up, completion of delivery, and failure. Failure information may include reasons like battery depletion, deadline violation, and failure of environment navigation. Updates are sent to a shared topic denoted by (`/task_status`) and defined by the standardized format of `TaskStatus.msg`.

4) BATTERY AND CHARGING PROTOCOL

When a robot's battery level drops below a predefined threshold, it publishes an urgent alert on `/robot_X/battery_alert` and requests charging through the `/request_charging` service. The coordinator reallocates pending tasks, assigns the nearest available charging station as a high-priority goal, and updates the robot status to *charging*. Upon reaching full capacity, a charging completion message is sent, and the robot becomes available for new assignments [36], [37].

5) DYNAMIC TASK REALLOCATION MESSAGES

To reduce idle time, a lightweight dynamic reallocation mechanism enables fast task reassignment without invoking the full SVCA+NSGA-II pipeline. When suitable nearby tasks are identified, a `fast-assignment` message is sent via the `/quick_assign` topic, containing minimal task information. This enables sub-second reassignment and opportunistic task execution with negligible computational overhead.

6) SYNCHRONIZATION AND TIMINGS

All messages include high-precision timestamps using `builtin_interfaces/Time` to ensure correct event ordering and accurate performance evaluation. The coordinator maintains a synchronized reference clock, while robots align timing using DDS-based synchronization mechanisms. Priority-based QoS profiles ensure reliable delivery for critical messages (task assignments, battery alerts), while best effort delivery is used for periodic state updates to reduce network load and latency [38], [39], [40].

VII. EXPERIMENTAL RESULTS AND ANALYSIS

All experiments were conducted on a computer equipped with an Intel® Core™ i9-13900 processor (32 cores), 32 GiB of RAM, and an NVIDIA T400 4 GB graphics card. The system runs Ubuntu 22.04.5 LTS (64-bit). The simulation environment was developed using ROS 2 Humble and Gazebo version 11.10.2. These specifications provide sufficient computational resources to handle the parallel execution of the island model genetic algorithm and the simultaneous simulation of multiple robots in complex warehouse environments.

A. PERFORMANCE METRICS

To evaluate the effectiveness of the proposed multi-robot task allocation system, we employ several key performance metrics that quantify both the optimization quality and computational efficiency:

1) PERCENTAGE OF COMPLETE SOLUTIONS (PCS)

These metric measures the percentage of optimization runs that successfully allocate all tasks to available robots while satisfying all constraints (deadlines, payload capacity, and battery levels). It is calculated as:

$$\mathrm{PCS} = 100 \times \frac{N_{\text{complete}}}{N_{\text{total}}} \tag{53}$$

where N_{complete} is the number of runs with all tasks allocated and N_{total} is the total number of runs.

2) AVERAGE RATE OF TASKS ALLOCATED (ARTA)

This metric represents the average percentage of tasks successfully allocated across all optimization runs, calculated

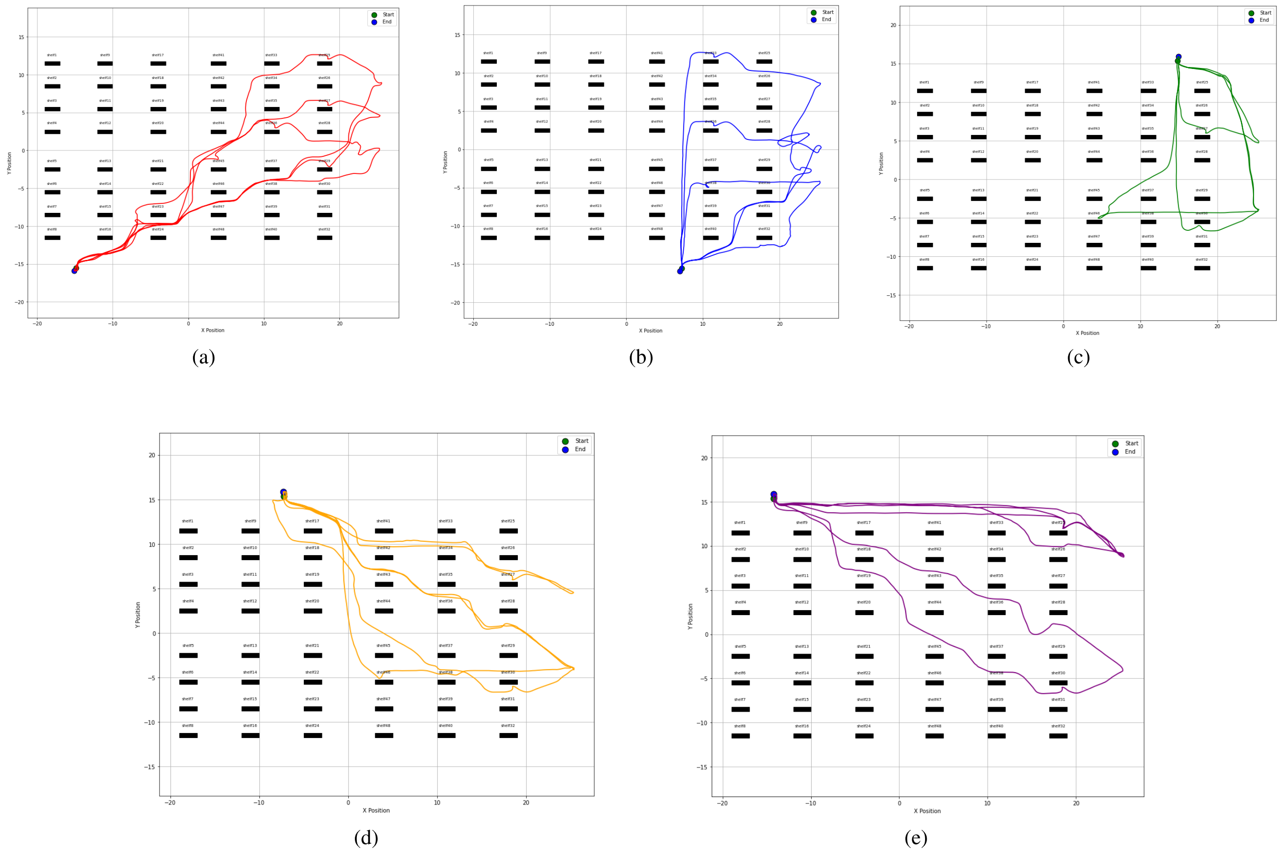


FIGURE 7. Total completed path for:(a) Robot 1,(b) Robot 2,(c) Robot 3,(d) Robot 4,(e) Robot 5. The experimental video is available at Video demonstration.

as:

$$\text{ARTA} = \frac{100}{N_{\text{total}}} \sum_{i=1}^{N_{\text{total}}} \frac{T_{\text{allocated}}}{T_{\text{total}}} \tag{54}$$

where $T_{\text{allocated}}$ is the number of tasks allocated in run i, and T_{total} is the total number of tasks in run i.

3) AVERAGE TIME TO ALLOCATE TASKS (ATAT)

This metric measures the computational efficiency by calculating the average time required to complete the optimization process:

$$\text{ATAT} = \frac{1}{N_{\text{total}}} \sum_{i=1}^{N_{\text{total}}} t_i \tag{55}$$

where t_i is the execution time for optimization run i.
These metrics collectively provide a comprehensive evaluation of the system's performance in terms of solution quality, computational efficiency, energy consumption, and execution reliability.

B. SCALIBILITY

Table 4 shows a quantitative scalability analysis of the proposed hybrid algorithm by analyzing its performance with an increasing number of robots available while keeping the number of tasks constant. With the number of robots scaled up from 3 to 5 for 15 tasks, the proposed algorithm shows consistent and better performance on all critical parameters. The task completion time (TCT) reduces substantially from 1581.06±32.63 s to 1396.25±9.52 s, which shows faster task execution with better parallelism and coordination. Also, with the rise in the number of robots, the total energy consumption (TEC) remains constant, which marginally increases from 39.07±0.12 to 40.20±0.11, thus proving the energy-efficient scalability of the algorithm. Further, the weighted task delay (WTD) also reduces from 7.63 ± 0.27 to 5.27 ± 0.25, which shows better task prioritization and scheduling efficiency. It is worth noticing that the balancing workload distribution (BWD) parameter remains optimal at 100 ± 0 for both scenarios, thus proving that the proposed algorithm performs perfectly on workload balance even at a scaled system.

C. COMPARITIVE ANALYSIS

A comprehensive comparative analysis was conducted to compare the performance of the proposed hybrid algorithm with the five best optimization algorithms: IMGA, SVCA, PSO, GA, and NSGA-II. Each of these algorithms was executed for ten runs, and in each run, the same set of experimental conditions was used, such as the same set of tasks, robot configurations, and constraint parameters. The

TABLE 4. Scalability analysis of the proposed hybrid algorithm under varying robot numbers.

No. of Robots	No. of Tasks	PCS	ARTA	ATAT	TCT	TEC	WTD	BWD
3	15	95 ± 0.63	0.39 ± 0.03	95.60 ± 0.49	1581.06 ± 32.63	39.07 ± 0.12	7.63 ± 0.27	100 ± 0.00
5	15	100 ± 0.00	100 ± 0.00	0.12 ± 0.02	1396.25 ± 9.52	40.20 ± 0.11	5.27 ± 0.25	100 ± 0.00

comparison is based on three major performance metrics: PCS, ARTA, and ATAT that together evaluate the solution quality and computational efficiency of each method. The results indicate that the proposed hybrid algorithm is the best one for optimal task allocation with very little computational overhead.

Table 5 provides a detailed comparative analysis of the proposed hybrid MRTA approach against representative benchmark algorithms in terms of problem formulation, evolutionary operators, constraint handling mechanisms, convergence behavior, adaptability, and computational cost. Conventional GA-based methods such as IMGA and standard GA rely on single-objective formulations and penalty-based constraint handling, which results in slow convergence and high computational overhead, typically requiring more than 100 generations to achieve stable solutions. Although SVCA is computationally efficient owing to its rule-based spatial decomposition approach, it tends to be greedy in convergence and less adaptable in dynamic settings. PSO and NSGA-II are more adaptable and efficient in handling multi-objective problems; however, their dependence on iterative swarm behavior and dominance-based sorting increases computational complexity and convergence time. In contrast, the proposed hybrid approach combines SVCA's spatial decomposition technique with elitist NSGA-II optimization. This synergy significantly reduces the required number of generations to approximately 30 while maintaining fast and stable convergence, high adaptability to dynamic task–robot configurations, and a balanced computational cost. The comparison clearly demonstrates that the proposed method achieves a superior trade-off between solution quality, convergence efficiency, and scalability, thereby addressing key limitations of existing MRTA approaches.

The comparison of PCS values throughout several repetitions for the six different task allocation algorithms hybrid (Proposed), IMGA, SVCA, PSO, GA, and NSGA-II is shown in Figure 8. The proposed hybrid algorithm shines through with the best performance as it gives 100% complete solutions every time, thus demonstrating its strong ability to assign all tasks while the constraints of deadline, payload capacity, and battery level are all satisfied. On the other hand, the IMGA displays uneven performance with PCS values fluctuating between 80% and 100% that indicates in some cases the IMGA is unable to find solutions due to very strict constraints. Both the SVCA and PSO algorithms are stable but not optimal at 90% and 80% respectively with the GA giving fairly constant results of 90-91%. The NSGA-II algorithm is ranked the lowest in performance among all the other ones, even being a multi-objective optimization one, with the majority of its PCS values being around 80% and occasionally 82%. It is evident from the results that the proposed hybrid algorithm has the ability to efficiently integrate the exploration power of the island modeling with problem-specific heuristics to always provide complete task allocation solutions, thus being very reliable for the warehouse logistics applications where all tasks must be assigned to meet operational requirements.

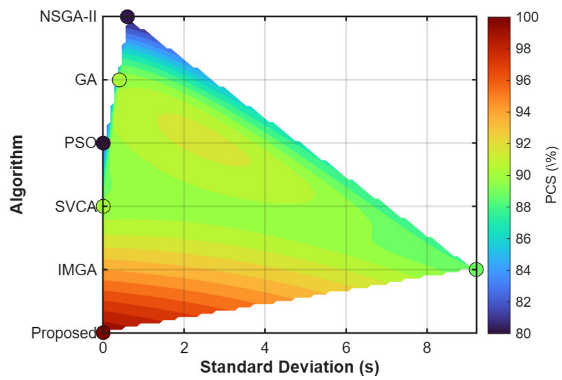


FIGURE 8. Percentage of complete solutions for different algorithm with several repetitions.

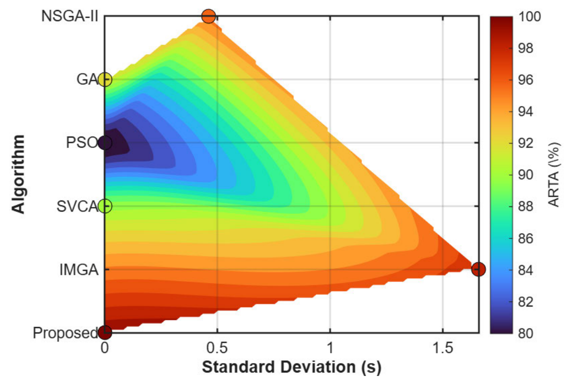


FIGURE 9. Average percentage of allocated tasks for different algorithm with several repetitions.

The picture 9 shows the ARTA metric over several experimental runs and the average percentage of tasks each algorithm has successfully allocated. The proposed hybrid algorithm has a perfect 100% ARTA in all the repetitions and thus it proves its capacity of allocating all the tasks provided without leaving any unassigned. The IMGA algorithm has variable performance that is strong at the same time, and

TABLE 5. Comparative analysis of MRTA baseline algorithms.

Algorithm	Problem Formulation	Crossover Operator	Mutation Operator	Constraint Handling / Repair	Generational Structure (No. of Generations)	Convergence Behavior	Adaptability	Computational Cost
IMGA	Single-objective MRTA formulated as makespan minimization	Partially Matched Crossover (PMX)	Swap mutation	Penalty-based handling of infeasible task–robot assignments	Standard generational GA (∼100–150 generations)	Stable solutions only after a large number of generations	Limited (static task allocation)	High
SVCA	Spatial Voronoi-based coverage allocation	Not applicable (rule-based clustering)	Not applicable	Geometric partitioning inherently enforces feasibility	Iterative spatial reassignment (no fixed generations)	Fast but greedy convergence	Moderate to low (environment-dependent)	Low
PSO	Continuous multi-robot path optimization	Not applicable	Velocity perturbation-based updates	Penalty functions for collision and boundary violations	Iterative swarm optimization (∼80–120 iterations)	Fast initial convergence, oscillatory later	Moderate	High
GA	Single-objective MRTA with path length optimization	One-point crossover	Random swap mutation	Penalization of infeasible chromosomes	Standard generational GA (∼100 generations)	Slow convergence, sensitive to parameter tuning	Low to moderate	High
NSGA-II	Multi-objective MRTA considering energy, time, and coverage	Uniform crossover	Gaussian mutation	Dominance-based constraint handling	NSGA-II with elitism (∼80–100 generations)	Reliable but computationally expensive	High	High
Proposed Work	Hybrid MRTA (SVCA + NSGA-II) integrating spatial decomposition and multi-objective optimization	Problem-aware crossover preserving task–robot consistency	Adaptive mutation guided by population diversity	Implicit constraint satisfaction via SVCA and dominance ranking	Elitist NSGA-II (∼30 generations only)	Fast and stable convergence	High (dynamic, scalable, real-time capable)	Moderate (balanced)

its ARTA values vary between 96% and 100%, meaning that in some cases it can get total allocations but in others it fails to assign 2-4% of tasks. The NSGA-II algorithm displays 95-96% stable performance and thus becomes the second best performer after the Hybrid approach. The GA consistently performs at 92% ARTA, while SVCA and PSO have even lower rates at 90% and 80% respectively. The superior ARTA performance of the Hybrid algorithm is due to the clever combination of several optimization strategies, such as dynamic population management, adaptive genetic operators, and constraint-aware heuristics. The high ARTA throughout is very critical in warehouse logistics cases where the tasks that are not allocated turn into delayed orders and thus reduced customer satisfaction. The results are indicative that the proposed method is capable of taking up the intricate multi-robot task allocation problems with heterogeneous robots and multiple constraints.

At the top of the page, Figure 10 illustrates the execution time of the six algorithms in a form of their ATAT measured in seconds over several trials. The newly suggested hybrid algorithm is this superfast one with ATAT times moving from 0.103 up to 0.147 seconds hence it is indeed the fastest among all algorithms compared. The algorithms do not compromise on solution quality by being extremely fast, which is proved by its perfect PCS and ARTA scores among others. The IMGA takes about 1.0 to 1.1 seconds to allocate the task which means it has approximately 8-10 times longer execution time than the Hybrid method. The NSGA-II algorithm is not that good in terms of speed as it has ATAT ranging from 1.4 to 1.6 seconds. The GA takes around 2.5 to 2.7 seconds while PSO offers inconsistent results from 3.0 to 3.4 seconds. The SVCA algorithm is the worst one regarding speed, as its ATAT time is around 5.1 to 5.2 seconds consistently, thus making it about 40 times slower than the new hybrid method. The Hybrid's superior time performance is a result of the efficient parallel processing

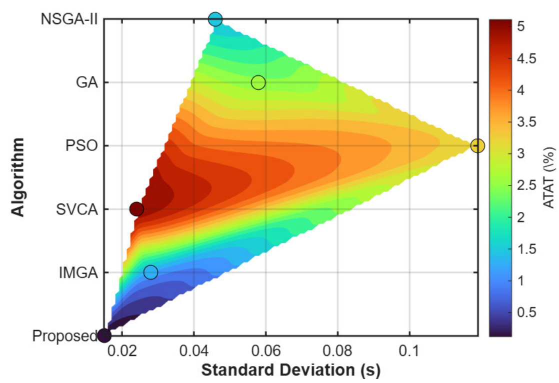


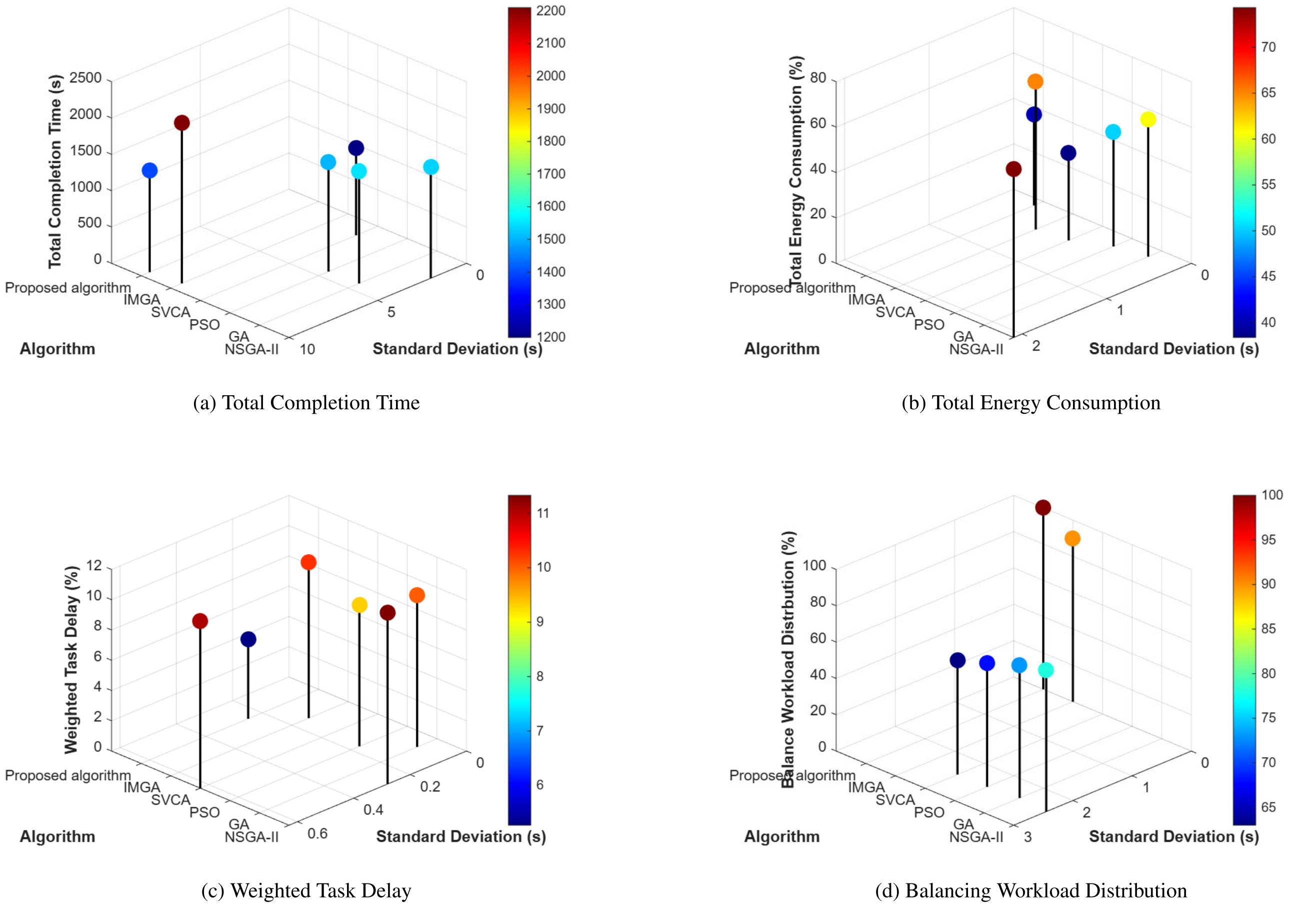
FIGURE 10. Average time to allocate tasks for different algorithm with 10 repetitions.

that uses the 32-core processor, the genetic operators' optimization that eliminates unnecessary iterations, and the early solution termination when the optimal one is found. This computational efficiency is very important in dynamic warehouse settings where task allocations must be done quite frequently in real-time as new orders come in, thus proving that the proposed method is not only accurate but also feasible for large-scale industrial adoption.

TABLE 6. Mean and standard deviation of performance metrics for different algorithms.

Algorithm	PCS Mean	PCS Std.	ARTA Mean	ARTA Std.	ATAT Mean	ATAT Std.
Proposed Hybrid	100.0	0.00	100.0	0.00	0.12	0.02
IMGA	88.8	9.72	98.2	1.75	1.04	0.03
SVCA	90.0	0.00	90.0	0.00	5.11	0.02
PSO	80.0	0.00	80.0	0.00	3.23	0.12
GA	90.2	0.42	92.0	0.00	2.59	0.06
NSGA-II	80.2	0.63	95.7	0.48	1.48	0.05

Mean and standard deviation of performance metrics PCS, ARTA, and ATAT for six different task allocation algorithms



(a) Total Completion Time

(b) Total Energy Consumption

(c) Weighted Task Delay

(d) Balancing Workload Distribution

FIGURE 11. Performance evaluation of baseline and proposed methods:(a) Total completion time,(b) Total energy consumption,(c) Weighted task delay,(d) Balancing workload distribution.

are presented in Table 6. The proposed Hybrid algorithm reached the maximum possible values with 100% PCS and ARTA and the minimum ATAT (0.122 s), which means that it was both precise and speedy. On the other hand, IMGA, SVCA, PSO, GA, and NSGA-II algorithms proved to be less effective as they had significantly lower PCS and ARTA values along with an increased ATAT, which points to a slower or less reliable task allocation performance. IMGA's PCS standard deviation is high (9.7), which means that there is a big variation in the scheduling efficiency across the repetitions, while the Hybrid algorithm kept zero contention, thereby confirming its excellence in stability and robustness [35].

Figure 11 presents a comprehensive performance comparison of the proposed algorithm against baseline methods across four conflicting objectives. As shown in Figure 10(a), the proposed algorithm achieves a total completion time of approximately 1400 s, which is substantially lower than IMGA (≈2100 s) and NSGA-II (≈1600 s), while maintaining a moderate variability of about 10 s, indicating stable execution. In Figure 10(b), the proposed method records the lowest total energy consumption (≈40 %) with a small standard deviation (≈1 s), outperforming PSO (≈65 %) and NSGA-II (≈60 %), demonstrating energy-efficient task execution. Figure 10(c) further shows that the proposed approach attains a weighted task delay of around 8 %, lower than GA (≈10 %) and IMGA (≈11 %), while also exhibiting reduced variability, reflecting improved task prioritization and scheduling fairness. Finally, Figure 10(d) indicates that the proposed algorithm achieves the highest workload balance (≈100 %) with minimal dispersion (≈1 s), whereas baseline methods such as PSO and GA remain below 50 %. Collectively, these results confirm that the proposed algorithm does not optimize a single objective at the expense of others; instead, it consistently provides a well-balanced Pareto-optimal solution, simultaneously minimizing completion time, energy consumption, and task delay while maximizing workload distribution, thereby demonstrating its robustness and suitability for multi-objective MRTA scenarios.

VIII. CONCLUSION AND FUTURE WORK

This research managed to develop and validate a centralized multi-robot coordination architecture which successfully coped with the complex challenge of task allocation and execution in heterogeneous warehouse logistics environments.

The method of combining SVCA with NSGA-II as a hybrid optimization algorithm shows high performance in all the evaluated metrics, scoring 100% in task allocation success rates while incurring the least computational overhead which is 0.122 seconds average allocation time. The dynamic task reallocation mechanisms and autonomous battery management integration remarkably improves system's flexibility and efficiency thus allowing the robots to opportunistically perform nearby tasks and manage their energy needs independently of the human staff.

The system's ability to simultaneously minimize makespan, energy consumption, and deadline violations while keeping a balanced workload distribution across heterogeneous robot teams has been confirmed by the extensive experimental validation in ROS2-based Gazebo simulations. A coordination architecture successfully operates through the task scheduling, allocation, execution monitoring and failure detection in realistic warehouse conditions with the involvement of multiple robots, different payload requirements and stringent deadline constraints. This way the results give not only proof but also confirmation of the proposed approach's validity for the industrial warehouse automation and logistics operations.

The directions for future research will be mainly the transition from simulated to physical hardware implementation of the project. The shift to real-world usage will require coordination of the architecture with the commercially available mobile robot platforms fitted with real sensors (LiDAR, cameras, IMUs), actuators, and embedded computing systems. The hardware implementation will allow validation under real conditions like sensor noise, communication delays, localization inaccuracies, dynamic obstacles, and environmental uncertainties that may not be completely simulated. Besides, we will look at scalability improvements to handle larger fleets of robots in large warehouse spaces, investigate decentralized coordination approaches to avoid single points of failure, involve machine learning techniques for adaptive parameter tuning based on historical performance data, and create sophisticated failure recovery strategies that will be able to reassign tasks and replan trajectories automatically whenever there is a robot malfunction or communication outage. The hardware testing will be a significant source of information for the system's strength, dependability, and real industrial applications versus deployability considerations.

ACKNOWLEDGMENT

The authors gratefully acknowledge Amrita Vishwa Vidyapeetham, India, and Delft University of Technology, The Netherlands, for their support and for fostering a collaborative research environment.

REFERENCES

[1] A. K. Cechinel and E. R. De Pieri, "Centralized multi-robot logistic system: An approach using the island model genetic algorithm as task scheduler," *Int. J. Adv. Robotic Syst.*, vol. 21, no. 5, pp. 1–19, Sep. 2024, doi: 10.1177/17298806241279595.

[2] A. K. Cechinel, E. R. De Pieri, A. L. F. Perez, and P. D. M. Plentz, "Multi-robot task allocation using island model genetic algorithm," *IFAC-PapersOnLine*, vol. 54, no. 1, pp. 558–563, 2021, doi: 10.1016/j.ifacol.2021.08.063.

[3] K. A. Athira, J. D. Udayan, and U. Subramaniam, "A systematic literature review on multi-robot task allocation," *ACM Comput. Surveys*, vol. 57, no. 3, pp. 1–28, Mar. 2025, doi: 10.1145/3700591.

[4] E. D. S. Cardoso, V. P. Bacheti, and M. Sarcinelli-Filho, "A leader-follower control system for a package-delivery drone," *IEEE Access*, vol. 13, pp. 55313–55323, 2025, doi: 10.1109/ACCESS.2025.3555031.

[5] L. F. S. Costa, T. P. Do Nascimento, and L. M. G. Gonçalves, "Online learning and teaching of emergent behaviors in multi-robot teams," *IEEE Access*, vol. 7, pp. 158989–159001, 2019, doi: 10.1109/ACCESS.2019.2951013.

[6] F. Yumbla, M. Fajardo-Pruna, A. Piguave, D. Ronquillo, R. Ortiz, J. B. Choi, G. Díaz, X. G. Pañeda, and H. Moon, "An open-source multi-robot framework system for collaborative environments based on ROS2," *IEEE Access*, vol. 13, pp. 16288–16302, 2025, doi: 10.1109/ACCESS.2025.3530391.

[7] M. Mokhtari, P. H. A. Mohamadi, M. Aernouts, R. K. Singh, B. Vanderborght, M. Weyn, and J. Famaey, "Energy-aware multi-robot task scheduling using meta-heuristic optimization methods for ambiently-powered robot swarms," *Robot. Auto. Syst.*, vol. 186, Apr. 2025, Art. no. 104898, doi: 10.1016/j.robot.2024.104898.

[8] J. G. Martin, F. J. Muros, J. M. Maestre, and E. F. Camacho, "Multi-robot task allocation clustering based on game theory," *Robot. Auto. Syst.*, vol. 161, Mar. 2023, Art. no. 104314, doi: 10.1016/j.robot.2022.104314.

[9] S. Chen, T. X. Lin, and F. Zhang, "Game-theoretical approach to multi-robot task allocation using a bio-inspired optimization strategy," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2023, pp. 5388–5393, doi: 10.1109/IROS55552.2023.10341947.

[10] C. Wu and D. Liu, "Solving the balance optimization problem of makespan and total computation time for cloud computing scheduling based on NSGA-II," in *Proc. Int. Conf. Digit. Anal. Process., Intell. Comput. (DAPIC)*, Feb. 2025, pp. 783–789, doi: 10.1109/DAPIC66097.2025.00149.

[11] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Rel. Eng. Syst. Saf.*, vol. 91, no. 9, pp. 992–1007, Sep. 2006, doi: 10.1016/j.ress.2005.11.018.

[12] J. Lu, L. Wei, M. M. Pour, Y. Mekonnen, and A. I. Sarwat, "Modeling discharge characteristics for predicting battery remaining life," in *Proc. IEEE Transp. Electrific. Conf. Expo (ITEC)*, Jun. 2017, pp. 468–473, doi: 10.1109/ITEC.2017.7993316.

[13] L. Altin, H. R. Topcuoglu, and F. S. Gürgen, "Network congestion aware multiobjective task scheduling in heterogeneous fog environments," *IEEE Trans. Ind. Informat.*, vol. 20, no. 2, pp. 3015–3024, Feb. 2024, doi: 10.1109/TII.2023.3299624.

[14] G. Chen, J. Hou, J. Dong, Z. Li, S. Gu, B. Zhang, J. Yu, and A. Knoll, "Multiobjective scheduling strategy with genetic algorithm and time-enhanced A* planning for autonomous parking robotics in high-density unmanned parking lots," *IEEE/ASME Trans. Mechatronics*, vol. 26, no. 3, pp. 1547–1557, Jun. 2021, doi: 10.1109/TMECH.2020.3023261.

[15] S. Chen and H. Qian, "Task allocation for inspection drone of power transmission lines based on improved NSGA-II," in *Proc. IEEE 7th Adv. Inf. Technol., Electron. Autom. Control Conf. (IAEAC)*, Mar. 2024, pp. 582–588, doi: 10.1109/IAEAC59436.2024.10503586.

[16] A. Miele, M. Lippi, and A. Gasparri, "A distributed framework for integrated task allocation and safe coordination in networked multi-robot systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 11219–11238, 2025, doi: 10.1109/TASE.2025.3532023.

[17] A. Budiman, P. Payeur, E. Lanteigne, and L. E. Garza-Castañón, "A hybrid algorithm for optimized task allocation and coordination among multiple specialized robots," in *Proc. IEEE 20th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2024, pp. 2328–2333, doi: 10.1109/CASE59546.2024.10711462.

[18] E. Nunes and M. Gini, "Multi-robot auctions for allocation of tasks with temporal constraints," in *Proc. AAAI Conf. Artif. Intell.*, vol. 29, 2015, pp. 2110–2116, doi: 10.1609/aaai.v29i1.9440.

[19] M. Li, B. Guo, J. Zhang, J. Liu, S. Liu, Z. Yu, Z. Li, and L. Xiang, "Decentralized multi-AGV task allocation based on multi-agent reinforcement learning with information potential field rewards," in *Proc. IEEE 18th Int. Conf. Mobile Ad Hoc Smart Syst. (MASS)*, Oct. 2021, pp. 482–489, doi: 10.1109/MASS52906.2021.00066.

[20] X. Luo, S. Xu, R. Liu, and C. Liu, "Decomposition-based hierarchical task allocation and planning for multi-robots under hierarchical temporal logic specifications," *IEEE Robot. Autom. Lett.*, vol. 9, no. 8, pp. 7182–7189, Aug. 2024, doi: 10.1109/LRA.2024.3412589.

[21] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 4487–4496, doi: 10.18653/v1/p19-1441.

[22] L. Zhang, J. Zhao, E. Lamon, Y. Wang, and X. Hong, "Energy efficient multi-robot task allocation constrained by time window and precedence," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 1–12, 2025, doi: 10.1109/TASE.2023.3312214.

[23] H. Wu, A. Ghadami, A. E. Bayrak, J. M. Smereka, and B. I. Epureanu, "Evaluating emergent coordination in multi-agent task allocation through causal inference and sub-team identification," *IEEE Robot. Autom. Lett.*, vol. 8, no. 2, pp. 728–735, Feb. 2023, doi: 10.1109/LRA.2022.3231497.

[24] K. Shibata, T. Jimbo, T. Odashima, K. Takeshita, and T. Matsubara, "Learning locally, communicating globally: Reinforcement learning of multi-robot task allocation for cooperative transport," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 11436–11443, 2023, doi: 10.1016/j.ifacol.2023.10.431.

[25] A. Milot, E. Chauveau, S. Lacroix, and C. Lesire, "Market-based multi-robot coordination with HTN planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 2606–2612, doi: 10.1109/IROS51168.2021.9636286.

[26] C. Bonhomme, C. Grand, C. Lesire, J.-L. Dufour, and C. Guettier, "Multi-agent path finding with task assignment and supporting constraints," *Frontiers Artif. Intell. Appl.*, vol. 392, pp. 3581–3588, Jun. 2024, doi: 10.3233/faia240913.

[27] A. Farinelli, N. Boscolo, E. Zanotto, and E. Pagello, "Advanced approaches for multi-robot coordination in logistic scenarios," *Robot. Auto. Syst.*, vol. 90, pp. 34–44, Apr. 2017, doi: 10.1016/j.robot.2016.08.010.

[28] D. Wang and H. Deng, "Multirobot coordination with deep reinforcement learning in complex environments," *Expert Syst. Appl.*, vol. 180, Oct. 2021, Art. no. 115128, doi: 10.1016/j.eswa.2021.115128.

[29] Z. Wu, Z. Li, D. Zhu, Q. Liao, and W. Yao, "A multi-robot task allocation method based on graph attention network and unsupervised learning," in *Proc. IEEE Int. Conf. Unmanned Syst. (ICUS)*, Oct. 2024, pp. 1222–1227, doi: 10.1109/ICUS61736.2024.10840067.

[30] K. J. Prakash, M. Muthukumaran, R. Sandhiya, T. Rajpradeesh, M. A. Deshmukh, and S. Thanikodi, "Swarm intelligence for multi-robot coordination in industrial environments," in *Proc. 9th Int. Conf. Sci. Technol. Eng. Math. (ICONSTEM)*, Apr. 2024, pp. 1–5, doi: 10.1109/ICONSTEM60960.2024.10568689.

[31] T. Borangiu, D.Trentesaux, P. Leitão, A. G. Boggino, and V. Botti, *Service Oriented, Holonic and Multi-Agent Manufacturing Systems for Industry of the Future: Proceedings of SOHOMA 2023* (Studies in Computational Intelligence), vol. 853, Spain, Oct. 2019, doi: 10.1007/978-3-030-27477-1.

[32] F. Zitouni, S. Harous, and R. Maamri, "A distributed approach to the multi-robot task allocation problem using the consensus-based bundle algorithm and ant colony system," *IEEE Access*, vol. 8, pp. 27479–27494, 2020, doi: 10.1109/ACCESS.2020.2971585.

[33] G. Varghese, T. G. C. Reddy, A. K. Menon, A. Paul, S. Kochuvila, R. V. Divya, R. Bhat, and N. Kumar, "Multi-robot system for mapping and localization," in *Proc. 8th Int. Conf. Robot. Autom. Eng. (ICRAE)*, Nov. 2023, pp. 79–84, doi: 10.1109/ICRAE59816.2023.10458504.

[34] T. G. Varghese, S. Kochuvila, N. Kumar, A. Paul, R. V. Divya, and S. Shailendra, "Exploration and coverage path planning of unknown environment using multi-robot system," in *Proc. IEEE Int. Conf. Electron., Comput. Commun. Technol. (CONECCT)*, Jul. 2024, pp. 1–5, doi: 10.1109/CONECCT62155.2024.10677248.

[35] A. A. N. Kumaar and S. Kochuvila, "Mobile service robot path planning using deep reinforcement learning," *IEEE Access*, vol. 11, pp. 100083–100096, 2023, doi: 10.1109/ACCESS.2023.3311519.

[36] M. U. Arif and S. Haider, "An evolutionary traveling salesman approach for multi-robot task allocation," in *Proc. 9th Int. Conf. Agents Artif. Intell.*, 2017, pp. 567–574, doi: 10.5220/0006197305670574.

[37] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, Sep. 2004, doi: 10.1177/0278364904045564.

[38] G. Skorobogatov, C. Barrado, and E. Salamí, "Multi-robot workspace division based on compact polygon decomposition," *IEEE Access*, vol. 9, pp. 165795–165805, 2021, doi: 10.1109/ACCESS.2021.3134760.

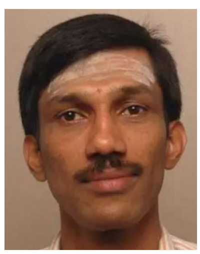
[39] J.-F. Gómez-Hernández, J.-M. Gutiérrez-Hernández, A. Jimeno-Morenilla, J.-L. Sánchez-Romero, and M.-D. Fabregat-Periago, "Development of an integrated robotic workcell for automated bonding in footwear manufacturing," *IEEE Access*, vol. 12, pp. 5066–5080, 2024, doi: 10.1109/ACCESS.2024.3350441.

[40] S.-K. Huang, W.-J. Wang, and C.-H. Sun, "A new multirobot path planning with priority order based on the generalized Voronoi diagram," *IEEE Access*, vol. 10, pp. 56564–56577, 2022, doi: 10.1109/ACCESS.2022.3176713.

GLACE VARGHESE T. (Member, IEEE) received the B.Tech. degree in electronics and communication engineering and M.Tech. degree in VLSI and embedded systems from A. P. J. Abdul Kalam Technological University, Kerala, India, in 2019 and 2021, respectively. She is currently pursuing the Ph.D. degree in robotics and automation with the Department of Electronics and Communication Engineering, Amrita Vishwa Vidyapeetham, Bengaluru, Karnataka, India. She has work experience in C-DAC, Trivandrum, Bosch Bengaluru, and Amrita Vishwa Vidyapeetham, Bengaluru. Her research interests include robot operating systems, multiple robots, accurate mapping, navigation, complex environment, motion planning, and task scheduling.

SREEJA KOCHUVILA (Senior Member, IEEE) received the B.Tech. degree in electrical and electronics engineering from M. G. University, Kerala, India, in 2006, the M.Tech. degree from NIT, Calicut, India, in 2008, and the Ph.D. degree in robotics and control from Amrita Vishwa Vidyapeetham, Bengaluru, Karnataka, India, in 2017. Her research interests include control systems, robotics, signal processing, and AI/ML.

NAVIN KUMAR (Senior Member, IEEE) is currently a Professor with the Department of Electronics and Communication Engineering, School of Engineering, Amrita Vishwa Vidyapeetham, Bengaluru Campus. His research interests include 5G and beyond technology, collaborative robot, drone-to-drone communication, intelligent transportation systems, visible light communication (LiFi), millimeter wave and THz, optical wireless communications, and wireless communications.



R. R. VENKATESHA PRASAD (Senior Member, IEEE) has been an Associate Professor with the Embedded Networked Systems Group, Delft University of Technology (TU Delft), since 2013. From 2005 to 2012, he was a Senior Researcher and an Adjunct Faculty with TU Delft. He has played a key role in bridging research and practice by contributing to standards on cognitive radios and tactile internet. His research interests include the IoT, cognitive radio, energy harvesting, sensors, communications, and robotic design. He is on the editorial board of multiple international transactions and magazines. He is the Vice-Chair of the IEEE Tactile Internet Standardization Group.

• • •