IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Service Placement in Dynamic Multi-AGV Environments for Minimized Energy Consumption

Claudia Torres-Pérez[1], Estela Carmona-Cejudo[1], Cristina Cervelló-Pastor[2], Maryam Masoumi[3], Estefanía Coronado[1,4], and Muhammad Shuaib Siddiqui[1]

***Abstract*—In multi-automated guided vehicle (AGV) environments, inefficient service placement increases energy consumption, and charging cycles, lowering battery lifespan. Consequently, minimizing energy consumption is key for maintaining operational efficiency and sustainability. Additionally, the unpredictable arrival of service requests in multi-AGV systems can lead to system saturation. However, previous research overlooked the energy costs of on-device computation, especially under dynamic service arrivals. To address these challenges, this work proposes an energy minimization service placement algorithm (EMSPA). The results demonstrate that EMSPA outperforms a baseline random selection (RS) algorithm for different numbers of AGVs, services, and tasks per service, reducing normalized energy consumption by up to 2.34% and improving mean service acceptance rates by up to 16.09% with lineal execution time overhead. Further, EMSPA outperforms a queue-aware scheduling and deadlock mitigation strategy (QASDMS) in terms of processing power ratio by over 58.94%.**

***Index Terms*—Task Planning, Planning, Scheduling and Coordination, Swarm Robotics.**

I. Introduction

In smart manufacturing, industrial devices like automated guided vehicles (AGVs) with embedded computing execute both physical (e.g., navigation) [1] and computational tasks (e.g., decision-making). While this reduces latency and enables new applications, poor energy management in multi-AGV systems often causes disruptive recharging cycles, increased downtime, and significant productivity loss [2]. Moreover, onboard task execution improves responsiveness, but it also consumes significant energy and may overload computational resources.

Furthermore, unpredictable service arrivals in industrial environments [3] exacerbate placement complexity, causing delayed or blocked services when system capacity is exceeded. Thus, placement strategies must consider service and task requirements, AGV capabilities, energy constraints, and system responsiveness. The work in [4] explored AGV task offloading to reduce task completion time. Other works targeted energy savings by optimizing AGV paths, travel distances, and speed [5]. Adaptive planning strategies for energy efficiency were also proposed in [6]. However, most approaches focused on optimizing path planning, neglecting the energy cost of local computation under dynamic service arrivals with varying computational and communication resource needs, location requirements, and tasks per service. Further, most works disregarded communication aspects, which can lead to delays, reduced data rates (DR), and potential service disruptions [7], hindering AGV operation efficiency.

This work addresses these gaps through two research questions: (i) How to optimize service placement among AGVs to balance performance and energy consumption under dynamic workloads and data rate constraints? (ii) How does central processing unit/graphics processing unit (CPU/GPU)-bound processing energy influence placement decisions in resource-constrained AGVs? To answer them, this work proposes an energy minimization service placement algorithm (EMSPA) to optimize service allocation under dynamic arrivals, which jointly considers computational load, energy use, and data rate reliability. Unlike prior works, EMSPA aims to optimize sustainability objectives and support service completion rates by accounting for CPU/GPU-dependent processing costs and bandwidth constraints. Evaluation results show that EMSPA reduces energy consumption by up to 2.34% and increases mean service acceptance by up to 16.09% compared to a baseline random selection (RS) algorithm, and reduces the power processing ratio by over 58.94% in relation to the queue-aware scheduling and deadlock mitigation strategy (QASDMS) [3]. These improvements are consistent across different AGV, service, and task configurations, demonstrating EMSPA's reliability in various scenarios.

The paper is organized as follows. Section II reviews related work. Section III introduces the system model. Section IV formulates the energy minimization problem. Section V presents the proposed solution. Section VI discusses evaluation results and Section VII concludes the paper.

II. Related Work

Multi-AGV systems enhance industrial efficiency, acting as mobile computing platforms to reduce latency and reliance on external servers. However, their limited battery and processing

Manuscript received: June, 2nd, 2025; Revised October, 3rd, 2025; Accepted November, 9, 2025. This paper was recommended for publication by Editor Chao-Bo Yan upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by COALESCE-6G PID2024-163028OB-I00, funded by MICIU/AEI/10.13039/501100011033/FEDER, EU; by the EU "NextGenerationEU/PRTR", MCIN and AEI (Spain) under project IJC2020-043058-I; by MCIN/AEI/10.13039/501100011033 (FEDER "a way of making Europe") under grant PID2022-142332OA-I00; by Spain's MINECO and EU's NextGeneration EU, under UNICO I+D 5G 2021, TSI-063000-2021-9-6GSMART-ICC. The authors would also like to acknowledge CERCA Programme/Generalitat de Catalunya for sponsoring part of this work.

[1]Claudia Torres-Pérez, Estela Carmona-Cejudo, Estefanía Coronado and Muhammad Shuaib Siddiqui are with i2CAT Foundation, Barcelona, Spain claudia.torres@i2cat.net. [2]Cristina Cervelló-Pastor is with Universitat Politècnica de Catalunya, Barcelona, Spain. [3]Maryam Masoumi is with Universidad de Valladolid, Valladolid, Spain. [4]Estefanía Coronado is with Universidad de Castilla-La Mancha, Albacete, Spain.

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

capacity make efficient service placement, based on AGV capabilities and service demands, crucial for performance and fleet coordination under dynamic conditions.

Prior research on service placement in AGV systems has focused on reducing energy consumption and delays while improving throughput. However, many overlooked real-time dynamics, such as unpredictable request arrivals and fluctuating resources. Liu et al. [4] presented a reinforcement learning-based task offloading scheme optimizing execution time and conflict resolution. Although this study neglected dynamic service arrival, which is critical in real-world industrial settings, other works considered this dynamic behavior [3], [8]–[10]. Bergmann et al. [8] introduced a vehicle assignment strategy to minimize completion time and increase throughput under dynamic task flows, but overlooked the computational consumption of AGVs. In [3], [9], dynamic job arrivals were modeled as continuous and unpredictable processes by using a Poisson distribution. A similar distribution was considered in [10], where a centralized system continuously collects real-time information about new tasks. However, these works did not consider energy implications.

TABLE I
PLACEMENT METHODS IN MULTI-AGV ENVIRONMENTS.

Ref.	Task type	Optimization objective	Energy dependency	Dynam. arrival
[3]	movement, processing	completion time	✗	✓
[4]	processing	completion time	✗	✗
[5]	movement, production	makespan, energy, quality	distance, time, energy coefficient	✗
[6]	movement	completion time, energy	distance, load, AGV weight	✗
[8]	movement	machine and AGV utilization, throughput	✗	✓
[9]	production	makespan	✗	✓
[10]	movement	travel, tardiness cost	✗	✓
[11]	movement, production, setup	makespan, idle time, energy	distance, speed, idle time	✗
[12]	movement	energy	distance, speed, load	✗
[13]	production, movement	satisfaction, no. of AGVs, energy	load, time, AGV weight	✗
[14]	movement	makespan	speed	✗
[15]	movement	energy, task deadline	distance	✓
Our work	movement, processing	energy	CPU-GPU consumption	✓

In this regard, the authors of [11] presented a multi-objective placement model for machine processing and AGV transport that jointly optimizes makespan, machine idle time, and energy consumption. Minimizing energy also requires careful task assignment and path selection in real-world scenarios [5], [12]. The work in [5] employed an optimization algorithm to reduce both makespan and energy consumption, while the work in [12] introduced an energy benchmarking framework to evaluate path planning efficiency. Similarly, the paper in [6] employed a grid map-based approach in a two-stage optimization algorithm to minimize task completion time and energy consumption. The authors of [13] used a multi-objective approach to reduce energy consumption and maximize customer satisfaction across production cycles, linking energy consumption to the AGV load transported. Riazi et al. [14] aimed to minimize energy consumption and service completion time by modifying AGV speeds. The research in [15] proposed a dual-class AGV strategy for handling dynamic tasks while conserving energy. However, the focus of these works was mainly related to optimizing energy consumption associated to distance, speed, and load, disregarding the optimization of computational resources. Although the impact of processing was recently considered in [3], [4], the focus of these works was completion time optimization, disregarding the substantial contribution of on-board processing to an AGV's total power consumption [2]. In addition, these works neglected the effect of imperfect communication channels, which is unrealistic in industrial settings. Table I compares the analyzed works.

Unlike previous research, this work proposes a dynamic multi-AGV service placement strategy with the objective of minimizing processing-related energy consumption. A realistic scenario is assumed with communication-related constraints and unpredictable, dynamic service arrival times.

III. SYSTEM MODEL

This section introduces a dynamic service placement model for multi-AGV scenarios. A centralized orchestrator manages AGVs and placement requests [16], with a dedicated module executing the placement algorithm, and an AGV controller issuing navigation and loading/unloading commands[1] [16].

A. Communication Channel Model

Communication in industrial multi-AGV environments is challenging because, unlike fixed computational resources, the data rate of AGVs varies significantly due to vehicle movement line-of-sight obstructions from obstacles links between access points (APs) and AGVs. To model these propagation effects, we adopt the attenuation factor approach from [18], which combines precision and scalability for industrial settings. Under this model, the received power at distance d_x from an AP is calculated using a ray-casting algorithm. Considering a reference path $P_0(d_0)$ representing the power at a nearby reference distance d_0 from the AP, a given path loss exponent η, the number of obstacles of material type e (m_e), a loss factor associated with material e (PF_e), and the number of material types E, the received power at a distance d_x from the AP is expressed as:

$$P_{AF_i}(d_x) = P_0(d_0) - 10\eta \cdot \ln \frac{d_x}{d_0} - \sum_{e=1}^{E} m_e \cdot PF_e. \quad (1)$$

The downlink data rate DR_i of AGV i at a distance d_x from the AP is defined in terms of $P_{AF_i}(d_x)$ as follows:

$$DR_i(d_x) = B \cdot \log_2 \left(1 + \frac{P_{AF_i}(d_x)}{N_0 \cdot B}\right). \quad (2)$$

This calculation accounts for the channel bandwidth B and the noise spectral density N_0, assuming a lossless uplink channel for simplicity as per [7] (note that critical control packets are transmitted in the downlink). The DR capacity C_i^{DR} of AGV i depends on propagation effects along its entire planned path during task execution, as per (2). It is defined as the minimum

[1]The trade-offs between centralized and decentralized control systems are out of the scope of this paper. Interested readers may refer to [17].

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

data rate found by sampling the instantaneous DR at regular intervals along the trajectory, a value used to assess the AGV's ability to host a given task.

B. Service Requests

In this work, a service $s \in \mathcal{S}$, $\mathcal{S} = \{1, \ldots, S\}$ is an ordered set of tasks $\mathcal{T}_s$ with predefined requirements for a multi-AGV environment. A task $t_{s,k} \in \mathcal{T}_s$ is the smallest unit of work an AGV executes as part of service s. The ordered set of sequential tasks $\mathcal{T}_s$ associated with a service s is expressed as $\mathcal{T}_s = \{t_{s,k} \mid k \in \mathcal{K}_s\}, \mathcal{K}_s = \{1, \ldots, K_s\}$, where $t_{s,k}$ represents the k-th task associated with service s, and $\mathcal{K}_s$ represents the ordered set of indices corresponding to all tasks associated with service s. Service requests arrive dynamically at the orchestrator, following a Poisson distribution with arrival rate λ_s [3]. The service placement algorithm allocates tasks based on AGV positions and resource capacity, ensuring that computational and communication demands are met. It is assumed that this decision is communicated to a central controller, which commands service execution orders to AGVs, as per [16].

Two task types are considered: movement-related (e.g., goods transportation) and processing-only (e.g., data processing, collaborative collision avoidance). It is assumed that both movement-related and processing-only tasks require computational resources. A binary parameter $\zeta_{s,k}$ represents the task type, being $\zeta_{s,k} = 0$ for processing tasks and $\zeta_{s,k} = 1$ for loading tasks. The requirements of each task $t_{s,k}$ are defined as the tuple $R_{s,k}$:

$$\begin{aligned} R_{s,k} = \{&R^{ST}_{s,k}, R^{RAM}_{s,k}, R^{CPU}_{s,k}, R^{GPU}_{s,k}, R^{DR}_{s,k}, \\ &R^{\phi}_{s,k}, x_{src}, y_{src}, x_{dst}, y_{dst}, \tau^{max}_{s,k}\}. \end{aligned} \tag{3}$$

Parameters $R^{ST}_{s,k}, R^{RAM}_{s,k}, R^{CPU}_{s,k}, R^{GPU}_{s,k}$ define the computational requirements of each task in terms of storage (ST), random access memory (RAM), CPU, and GPU, respectively. $R^{DR}_{s,k}$ is the minimum DR requirement for task $t_{s,k}$. For loading tasks, source and destination positions are given by x_{src}, y_{src} and x_{dst}, y_{dst}, and the load to be carried by $R^{\phi}_{s,k}$.

AGVs are assumed to move at a known and constant average speed v_i, following a pre-planned route. Without loss of generality, for the purpose of service placement evaluations that is of interest in this work, AGVs are assumed to follow a shortest path routing algorithm [19]. The placement algorithm presented in Section V must ensure that the AGVs meet the computational and communication requirements for each task, considering both the computational availability of AGVs and the quality of the communication channel along the route. The completion time for each task depends on its nature. For loading tasks, the time is calculated based on the path traversed by the AGV and its velocity to the source and, subsequently, to the destination. For processing tasks, there is no source and destination node, and AGVs can process the data at any location. Both types of tasks have a maximum time requirement for task completion, defined as $\tau^{max}_{s,k}$.

C. Autonomous Guided Vehicles (AGVs)

In the system model, a set $\mathcal{I}$ of AGVs is considered for task allocation. The normalized capacity of AGV i, $1 \leq i \leq I$, is represented by the vector $\mathbf{c}_i$ of initial available resources, namely ST, RAM, CPU, GPU, load weight, and DR. Formally, this vector is expressed as:

$$\mathbf{c}_i = \{C^{ST}_i, C^{RAM}_i, C^{CPU}_i, C^{GPU}_i, C^{\phi}_i, C^{DR}_i\}. \tag{4}$$

The current availability vector $\mathbf{a}_i$ of AGV i is defined as:

$$\mathbf{a}_i = \{A^{ST}_i, A^{RAM}_i, A^{CPU}_i, A^{GPU}_i, A^{\phi}_i, A^{DR}_i\}, \tag{5}$$

where A^{ST}_i, A^{RAM}_i, A^{CPU}_i, A^{GPU}_i, and A^{DR}_i represent the normalized availability in terms of storage, RAM, CPU, GPU, and data rate on AGV i, respectively. The AGV controller provides the AGV's velocity v_i and its current location (x_i, y_i) at any given time. The load availability of an AGV is denoted by A^{ϕ}_i. The time needed for an AGV to complete a task, τ_i, depends on the task type as follows:

$$\tau_i(t_{s,k}) = \begin{cases} \beta \cdot F_{s,k}, & \text{if } \zeta_{s,k} = 0, \\ \frac{d_{i,src}}{v_i} + \frac{d_{src,dst}}{v_i}, & \text{if } \zeta_{s,k} = 1, \end{cases} \tag{6}$$

where $F_{s,k}$ is the computational complexity of task $t_{s,k}$ with respect to a benchmark task β. The execution duration of processing tasks depends on the task computational complexity and on the computational resources of AGV i, and that of movement-related tasks depends on the time to move AGV i from location to source and from source to destination.

D. Energy Consumption

This paper follows the computational energy model from [2], where the power consumption PW_i of the computing platform of AGV i depends on its CPU and GPU usage, even when idle. The relationship between CPU and power consumption is modeled as:

$$PW_{cpu_i} = \begin{cases} 34.9, & p_{cpu_i} \geq 0.75, \\ -56.2p^2_{cpu_i} + 88p_{cpu_i} & \\ \quad +0.51, & p_{cpu_i} < 0.75, \end{cases} \tag{7}$$

where p_{cpu_i} is the normalized CPU consumption of AGV i, $\forall i$, i.e. $p_{cpu_i} = C^{CPU}_i - A^{CPU}_i$. Further, the relationship between the GPU and power consumption is given by:

$$PW_{gpu_i} = \begin{cases} 7.6 \cdot p^2_{gpu_i} + 24p_{gpu_i} + 0.006f_{g_i} & \\ \quad + 0.3t_{g_i} + 11, & f_{g_i} < 800, \\ 143p^2_{gpu_i} + 23p_{gpu_i} + 0.04f_{g_i} & \\ \quad + 2t_{g_i} - 66, & f_{g_i} \geq 800, \end{cases} \tag{8}$$

where p_{gpu_i} is the normalized GPU consumption of AGV i, $\forall i$, i.e. $p_{gpu_i} = C^{GPU}_i - A^{GPU}_i$. Finally, the overall power consumption of AGV i is given by:

$$PW_i = \left(PW_{cpu_i} + PW_{gpu_i} + P_{others_i}\right)/\eta_{c_i} + P_{off_i}, \tag{9}$$

where P_{others_i} is the power consumption of other components, η_{c_i} is the energy transmission efficiency, and P_{off_i} is the power consumption of the platform in standby state. For simplicity, and without loss of generality, constant average AGV weight and speed with no stops are assumed. Although the relevance of energy consumption associated to mechanical-related parameters such as weight and speed is acknowledged, these factors fall outside of the scope of this paper. Interested readers may refer to [2] for an overview of the effect of these parameters on AGV power consumption.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

IV. Service Placement Problem with Energy Minimization Objective

This work aims to minimize energy consumption while meeting all task requirements in the dynamic environment described in Section III. Upon the arrival of a new service placement request, the service placement module must select the most energy-efficient AGV for each task in the service. The energy minimization process is triggered upon the dynamic arrival of a service's first task, $t_{s,1}$, or when a subsequent task $t_{s,k+1}$ becomes available after its predecessor completes. Each task placement is treated as an independent optimization iteration, where the algorithm allocates resources efficiently while minimizing energy consumption. To support this process, an optimization problem is formulated to determine the most energy-efficient AGV for task placement.

Let $\varrho = [\rho_1, \rho_2, \cdots, \rho_{R-1}]$ denote the ordered set of currently running task requests, where each ρ_r, $1 \leq r \leq R-1$, corresponds to a served request for task $t_{s,k}$, and ρ_R represents the next request to be served. The decision variable $\chi_{\rho_R,i}$ is defined as:

$$\chi_{\rho_R,i} = \begin{cases} 1, & \text{if } t_{s,k} \text{ from request } \rho_R \text{ is assigned to AGV } i, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

Considering that $R-1$ requests have been successfully allocated and are running in the system, the resource availability of AGV i can be expressed as a function of its capacity $\mathbf{c}_i$ and of the requirements of request ρ_R associated to $t_{s,k}$, $\mathbf{r}_{s,k} = \{R^{ST}_{s,k}, R^{RAM}_{s,k}, R^{CPU}_{s,k}, R^{GPU}_{s,k}, R^{\phi}_{s,k}, R^{DR}_{s,k}\}$ as:

$$\mathbf{a}_i = \mathbf{c}_i - \sum_{r=1}^{R-1} \mathbf{r}_{s,k} \cdot \chi_{\rho_R,i}, \quad \forall i \in \mathcal{I}. \tag{11}$$

On the basis of the definitions in (7) and (8), the utility function U is expressed as a function of variable CPU and GPU consumption[2], as follows:

$$U = \sum_{i=1}^{I} PW_i \left(p_{cpu_i} + p_{gpu_i} + \chi_{\rho_R,i} \cdot \left(R^{CPU}_{s,k} + R^{GPU}_{s,k} \right) \right). \tag{12}$$

The target energy minimization optimization problem results:

$$\underset{\chi_{\rho_R,i}}{\text{minimize}} \quad U \tag{13a}$$

$$\text{subject to} \quad \chi_{\rho_R,i} \cdot R^{ST}_{s,k} \leq A^{ST}_i, \quad \forall i \in \mathcal{I}, \tag{13b}$$

$$\chi_{\rho_R,i} \cdot R^{RAM}_{s,k} \leq A^{RAM}_i, \quad \forall i \in \mathcal{I}, \tag{13c}$$

$$\chi_{\rho_R,i} \cdot R^{CPU}_{s,k} \leq A^{CPU}_i, \quad \forall i \in \mathcal{I}, \tag{13d}$$

$$\chi_{\rho_R,i} \cdot R^{GPU}_{s,k} \leq A^{GPU}_i, \quad \forall i \in \mathcal{I}, \tag{13e}$$

$$\chi_{\rho_R,i} \cdot R^{\phi}_{s,k} \leq A^{\phi}_i, \quad \forall i \in \mathcal{I}, \tag{13f}$$

$$\chi_{\rho_R,i} \cdot R^{DR}_{s,k} \leq A^{DR}_i, \quad \forall i \in \mathcal{I}, \tag{13g}$$

$$\tau_i(t_{s,k}) \leq \tau^{max}_{s,k}, \quad \forall i \in \mathcal{I}, \tag{13h}$$

$$\chi_{\rho_R,i} \leq 1 - l_i \cdot \zeta_{s,k}, \quad \forall i \in \mathcal{I}. \tag{13i}$$

The optimization objective in (13) minimizes the average power consumption across AGVs during the placement of task request ρ_R. Constraints ensure that the capacity of AGV i meets the requirements for storage (13b), RAM (13c), CPU (13d), GPU (13e), load weight (13f), and DR (13g) for any given task $t_{s,k}$. Constraint (13h) enforces that the task is completed within the task's maximum allowable time. An additional constraint specifies that AGVs can host only one loading task simultaneously (13i). A parameter l_i is introduced, taking the value 1 if an AGV i has an active loading task at the time of request ρ_R placement, and 0 otherwise.

[2]Note that frequency f_{g_i} and temperature t_{g_i} in (8), along with P_{others_i}, η_{c_i}, and P_{off_i} in (9), are considered fixed constants values. For a detailed analysis of PW_{gpu_i}, interested readers may refer to [2].

V. Energy Minimization Service Placement Algorithm

The dynamic nature of task arrivals and fluctuating channel conditions in multi-AGV environments introduces significant complexity, resulting in a mixed-integer optimization problem that is NP-hard [20]. To solve the problem in (13), this work proposes a heuristic, named EMSPA, to reduce computational energy consumption by assigning each task to the AGV that results in the lowest system-wide power consumption, while satisfying all constraints. EMSPA adopts a constrained exhaustive search over feasible AGVs. By assessing only AGVs that meet placement criteria, EMSPA identifies the assignment with the lowest incremental energy cost. This low-complexity approximation of the optimal solution guarantees a local optimum for each task allocation.

Algorithm 1 EMSPA

Input: $\mathcal{A}, t_{s,k}$ **Output:** `best_AGV`, `min_PW`
`best_AGV` ← None; `min_PW` ← 0; `total_PW` ← 0; `candidate_AGV_array` ← {}
for $i \in \mathcal{I}$ **do**
  if Constraints (13b)-(13h) hold **then**
    if $\zeta_{s,k} = 0$ **or** ($\zeta_{s,k} = 1$ **and** $l_i = 0$) **then**
      `candidate_AGV_array` ← i
    end if
  end if
end for
if `candidate_AGV_array` is empty **then**
  $t_{s,k}$ is rejected; `best_AGV` ← None
else
  for $i \in$ `candidate_AGV_array` **do**
    for $j \in \mathcal{I}$ **do**
      $p'_{cpu_j} \leftarrow p_{cpu_j} + R^{CPU}_{s,k}$; $p'_{gpu_j} \leftarrow p_{gpu_j} + R^{GPU}_{s,k}$ (if $j = i$; else $p'_{cpu_j} \leftarrow p_{cpu_j}$, $p'_{gpu_j} \leftarrow p_{gpu_j}$)
      `total_PW` ← `total_PW` + $PW_j(p'_{cpu_j}, p'_{gpu_j})$
    end for
    if `min_PW` = 0 **or** `total_PW` ≤ `min_PW` **then**
      `min_PW` ← `total_PW`; `best_AGV` ← i
    end if
  end for
end if
return `best_AGV`, `min_PW`

EMSPA is presented in Algorithm 1. EMSPA maintains an AGV availability matrix $\mathcal{A} = \{\mathbf{a}_1, \mathbf{a}_2, \cdots\}$, which tracks the current resources and location. Upon receiving a new event, EMSPA distinguishes between two types: arrival (i.e.,

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

a request to place a task) and finish (i.e., a notification that a task has been successfully finalized). Because service tasks are sequential and interdependent, a finish event for one task makes the next task eligible for placement. For arrival events, EMSPA updates the availability matrix $\mathcal{A}$ with the latest AGV resource availability from the orchestrator, and location data from the AGVs controller. If the preceding task is completed, EMSPA calculates the additional power consumption associated with placing the new task on each eligible AGV. It then selects the AGV that minimizes total system consumption and updates the AGV availability vector accordingly. Once the task is executed, a finish event is issued, and AGV resource availability and location information are updated again.

The algorithm operates iteratively to account for the dynamic nature of asynchronous incoming service requests. E.g., during the time between the completion of task $t_{s,k}$ and the start of the next task $t_{s,k+1}$ in service s, a new service $s+1$ may arrive. In such cases, the first task of the new service $s+1$, i.e., $t_{s+1,1}$, must also be placed. While tasks from different services can run in parallel, tasks from the same service are assumed to be interdependent and thus are managed sequentially [5], [11]. EMSPA handles such interleaved task arrivals seamlessly. Note that EMSPA follows a greedy, task-by-task placement strategy, which may not ensure global energy optimality over time. However, this limitation can be overcome by applying predictive or batch-based scheduling to improve long-term performance.

VI. Performance Evaluation

A. Methodology

EMSPA is evaluated in a Python-based simulation of a multi-AGV setting with a heterogeneous fleet and varying service configurations. AGVs are equipped with different commercially available computing platforms. [3] AGV specifications are randomly selected among the aforementioned configurations to reflect realistic deployment diversity. Constant speeds are sampled from a uniform distribution in the range $[5, 10]$ m/s, while their load capacities are 100 kg. AGVs communication and obstacles characteristics follow the specifications in [24]. A virtual factory environment of 100x100 meters is considered with six fixed obstacles, whose positions remain static throughout the experiments. Service requests arrive following a Poisson process with a fixed rate $\lambda_s = 0.1$, reflecting the stochastic nature of industrial tasks. Each service consists of a sequence of tasks with requirements drawn from uniform distributions. The CPU utilization modeling for movement tasks follows a benchmark profile from [25], with a mean of 2.56%, standard deviation of 3.92%, and values ranging from [0, 24.87%]. For processing tasks, average CPU utilization is 1.56% with a standard deviation of 3.02%, and values are distributed between [0, 16.62%]. It is assumed that the normalized consumption values for RAM, storage, and GPU are of similar magnitude to CPU consumption for movement tasks, across both types of tasks. The normalized consumption profile of the JETSON ORIN NANO SUPER platform [25] is used as a baseline to map resource consumption in the heterogeneous fleet of AGVs considered in this work. Task execution times vary uniformly between $[8, 72]$ seconds, reflecting realistic variability as in [26]. DR requirements are sampled from a uniform distribution in the range $[1, 2]$ Mbps and for load weight between $[1, 2]$ kg. Each simulation is repeated 30 times, with results reported using 95% confidence intervals. AGV configurations, task arrivals and requirements vary across repetitions, allowing analysis of EMSPA's performance under diverse operational conditions. EMSPA is evaluated against two baselines: a RS algorithm that assigns tasks randomly among AGVs that satisfy constraints (13b)-(13i), as in [27], [28]; and the QASDMS [3], that minimizes the total time required to complete tasks.

[3]The selected platforms and their thermal design power (TDP) are Intel(R) Core(TM) i7-8700 CPU (65W TDP) [21], GeForce RTX 3090 GPU (350W TDP) [22], and NVIDIA Jetson Xavier modules (AGX Xavier Industrial (20W-40W TDP), AGX Xavier 32GB (10W-30W TDP), Xavier NX 16GB (10W-20W TDP), and Xavier NX 8GB (10W-20W TDP) [23]).

TABLE II
Scenario Simulation Parameters.

Scenario	No. AGVs	No. services	Tasks per service
1	5	20, 30, 40, 50, 60	3, 5, 7
2	3, 5, 7, 9, 11	40	3, 5, 7

Two experimental scenarios are defined. *Scenario 1* evaluates EMSPA's performance with respect to RS's under increasing workloads by varying the number of services and tasks per service. *Scenario 2* analyzes the performance of EMSPA with respect to RS for varying number of AGVs. Scenario settings are detailed in Table II. The following performance metrics are evaluated: completion time (CT), total energy consumption (EC), service acceptance rate (SAR), and performance efficiency (Γ). CT is the total time to execute all services, indicating placement efficiency and execution speed. Total energy consumption refers to the cumulative energy used by all AGVs over time, i.e.:

$$EC = \sum_{\tau=0}^{CT} \sum_{i=1}^{I} PW_i \cdot \Delta\tau, \tag{14}$$

where PW_i is the power consumption of the i-th AGV at time τ. The performance efficiency parameter provides the algorithm efficiency in terms of normalized energy savings with respect to service throughput:

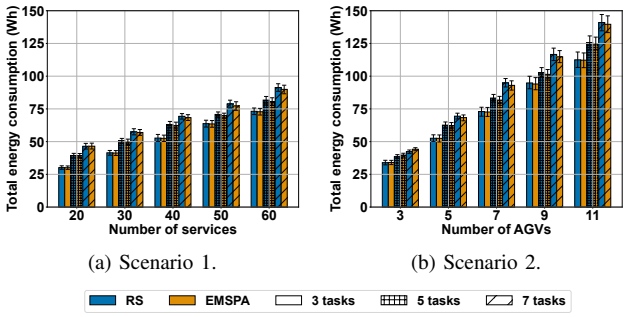
$$\Gamma = (1 - \overline{EC}) \cdot SAR, \tag{15}$$

where $\overline{EC} = EC/BEC$, and BEC is the baseline energy assuming all AGVs operate at full CPU and GPU load. The service acceptance rate refers to the ratio of accepted service requests, indicating system throughput.

B. Experimental Results

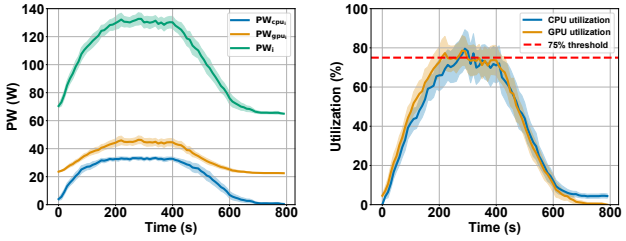
In both *Scenario 1* and *Scenario 2*, total EC predictably increases with system load (i.e., more services, tasks or AGVs), as shown in Fig. 1. In *Scenario 1*, for a fixed number of AGVs, EMSPA consistently achieves lower energy consumption than RS (Fig. 1a). This trend highlights EMSPA's effectiveness in

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.



(a) Scenario 1. (b) Scenario 2.

Fig. 1. Total energy consumption.

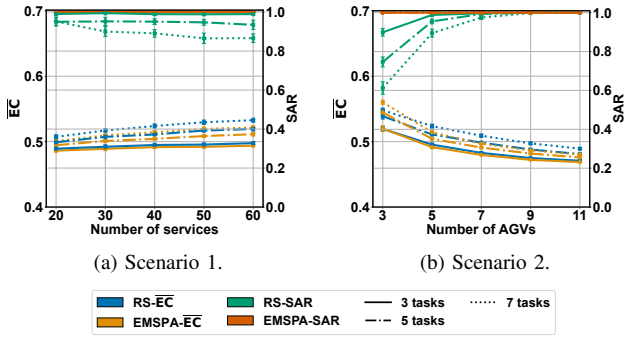


(a) Power consumption. (b) CPU and GPU utilization.

Fig. 2. Consumption for one AGV (40 services, 7 tasks per service).

enhancing energy efficiency by strategically placing tasks in the AGV that minimizes computational energy consumption. For *Scenario 2*, Fig. 1b demonstrates that total energy consumption (EC) also increases with the number of AGVs, as expected due to the lager number of active computing AGVs. However, as the number of AGVs increases, the performance gap in terms of energy consumption also increases between EMSPA and RS. This is attributed to the increased resource diversity which, in turn, enhances EMSPA's ability to find lower-power task placement solutions. However, for scenarios with over 7 AGVs, the perceived energy consumption difference for all task configurations starts decreasing because the consumption of the services decreases with respect to available resources, and AGVs have minimal resource occupation for both algorithms. Nevertheless, EMSPA outperforms RS in configurations with 5, 7, 9, and 11 AGVs. Note, however, that an exception is observed in the 3-AGV configuration with five and seven tasks per service. In this configuration, resource saturation causes RS to frequently fail constraint (13i) and reject more services. This lower SAR results in lower total energy consumption for RS in this specific high-saturation case, highlighting a trade-off.

Fig. 2 illustrates the direct relationship between the computational workload (Fig. 2b) and the instantaneous power consumption (Fig. 2a) of an AGV's computing platform. The period of peak utilization, which corresponds to maximum task concurrency in the simulation, directly translates to a peak power consumption of nearly 140W. This increased utilization results in a corresponding increase in power consumption for each component, as depicted in Fig. 2a. It shows a trend where power consumption is a non-linear function of workloads. This approach accounts for the significant power consumption of the computing platform, a critical optimization factor often overlooked in the analyzed related works. Specifically, for the



(a) Scenario 1. (b) Scenario 2.

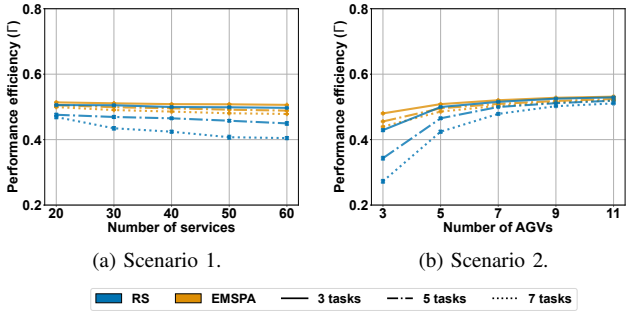
Fig. 3. Performance for different configurations.

configuration with 40 services and 7 tasks per service in *Scenario 1*, the power consumed by the CPU and GPU averages 50.4% of the total computational platform power consumption, demonstrating the importance of these parameters.

Fig. 3 includes performance for different configurations. The trade-off between $\overline{EC}$ and SAR is depicted in Fig. 3a for *Scenario 1*, where it can be seen that EMSPA outperforms RS for both metrics. Specifically, EMSPA achieves mean $\overline{EC}$ reductions of 0.68%, 1.31%, and 1.72% for 3, 5 and 7 task per service, respectively, with values up to 2.09% in the configuration with 5 AGVs, 60 services and 7 tasks per service, demonstrating an increasing trend with the number of tasks per service due to order statistics effect. The significance of this improvement is particularly relevant when considered in absolute terms, given the relatively large share of power consumption associated to processing in AGV platforms. Fig. 3a also depicts the performance in terms of SAR. For EMSPA, the service acceptance rate is 100% across all configurations, while RS suffers from resource fragmentation and constraint violations, particularly under high load. As system load increases, the probability of all AGVs being occupied with loading tasks is higher due to resource fragmentation. Thus, constraint (13i) cannot be met, since AGVs can only host one loading task at a time, leading to a lower SAR for RS. This finding is consistent with results in Fig. 1a. On average, EMSPA outperforms RS in terms of SAR by 0.98%, 5.61% and 11.89% for 3, 5 and 7 tasks per service, respectively.

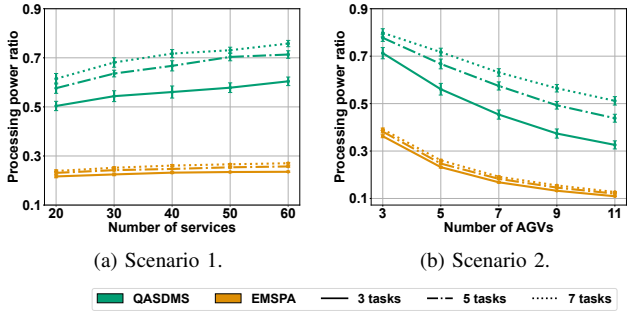
Fig. 3b presents $\overline{EC}$ and SAR evaluations, respectively, for *Scenario 2*. According to this figure, $\overline{EC}$ decreases as the number of AGVs increases. This indicates that EMSPA's power-saving decisions become more effective with system scaling, since the availability of additional AGVs provides EMSPA greater flexibility in placing new tasks to minimize overall energy use. Thus, EMSPA becomes increasingly successful in finding energy-efficient placements as system complexity and scale grow, resulting in a less than proportional increase in total $\overline{EC}$. EMSPA achieves mean $\overline{EC}$ savings over RS of 0.45%, 0.80%, and 1.16% for 3, 5, and 7 tasks per service, respectively, with differences up to 2.34% in the configuration with 7 AGVs, 40 services and 7 tasks per service. Consistently with the results in Fig. 1b, for a scenario with 3 AGVs, $\overline{EC}$ is lower for RS than for EMSPA, when resource saturation is higher, as seen in configurations with 5 and 7 tasks per service. For other configurations, the positive $\overline{EC}$ gap

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.



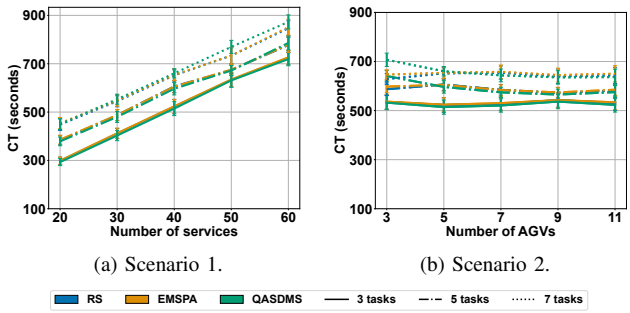
(a) Scenario 1. (b) Scenario 2.

Fig. 4. Performance efficiency (Γ) evaluation.



(a) Scenario 1. (b) Scenario 2.

Fig. 5. Processing power ratio evaluation.



(a) Scenario 1. (b) Scenario 2.

Fig. 6. Completion time evaluation.

between EMSPA and RS decreases for an increasing number of AGVs, as total energy consumption becomes dominated by AGV idle power consumption. Since excess energy required to execute tasks is relatively small compared to their baseline consumption, both algorithms produce increasingly similar $\overline{EC}$ for configurations with a higher number of AGVs. Similar to *Scenario 1*, potential $\overline{EC}$ savings increase with the number of tasks per service.

As the number of AGVs increases, SAR also improves for EMSPA and RS, as depicted in Fig. 3b. However, the performance gap between EMSPA and RS decreases as the number of AGVs increases. For a low number of AGVs, EMSPA achieves load balancing, yielding additional free space across AGVs to place the services, in contrast to RS, which leads to blocking with higher probability. However, as the number of AGVs increase, RS achieves a similar SAR compared to EMSPA for over 7 AGVs configurations, thanks to the additional resource availability to host the services. Similar to *Scenario 1*, in *Scenario 2*, the performance gap between EMSPA and RS increases with the number of tasks per service. On average, EMSPA outperforms RS in terms of SAR by 2.70%, 8.02%, and 16.09% for 3, 5, and 7 tasks per service, respectively.

Performance efficiency (Γ) evaluations are given in Fig. 4a and Fig. 4b for *Scenario 1* and *Scenario 2*, respectively, demonstrating that EMSPA outperforms RS for all configurations. In *Scenario 1*, EMSPA obtains higher performance efficiency values than RS, showing higher joint efficiency related to SAR and $\overline{EC}$, across all the configurations. However, performance efficiency slowly decreases with the number of services; intuitively, this is due to an increase in $\overline{EC}$ and a decrease in SAR for a higher number of services relative to available resources, according to the results in Fig. 3a, respectively. In *Scenario 2*, EMSPA also outperforms RS in terms of performance efficiency. This result is consistent with the findings in Fig. 3b, where it can be observed that the $\overline{EC}$ decreases with an increasing number of AGVs; and SAR converges to its maximum as the number of AGVs increases. In the 3 AGVs setup with 5 and 7 tasks per service, despite the enhanced $\overline{EC}$ performance of RS over EMSPA, the performance efficiency in Fig. 4b indicates a better performance of EMSPA. This result holds for configurations with five and seven AGVs. However, for configurations of over seven AGVs, EMSPA and RS yield comparable values of Γ, due to similar outcomes in terms of SAR and $\overline{EC}$. However, EMSPA still achieves and improved Γ over RS.

Fig. 5a and Fig. 5b evaluate EMSPA against QASDMS in terms of processing power ratio for a variable number of services and AGVs, respectively. The processing power ratio measures the fraction of the total power consumed by the AGVs dedicated to active processing tasks. For both scenarios, EMSPA consistently outperforms QASDMS, as it activates fewer AGVs, reducing idle power consumption. This makes EMSPA more efficient in reducing processing power and optimizing AGV utilization. The trends in processing power ratio for EMSPA are consistent with the $\overline{EC}$ shown in Fig. 3a and Fig. 3b, as both metrics are driven by the system workload and AGVs computational capacity. In *Scenario 1*, higher workloads lead to increased processing power and total power consumption, exhibiting a similar trend. In *Scenario 2*, although processing power consumption decreases with the number of AGVs for both EMSPA and QASDMS, QASDMS activates more AGVs than EMSPA, leading to a higher processing power ratio. Specifically, EMSPA's processing power ratio is on average 58.94%, 62.52%, and 63.14% lower than QASDMS in *Scenario 1* for 3, 5, and 7 tasks per service, respectively. In *Scenario 2*, EMSPA outperforms QASDMS by an average of 60.33%, 65.0%, and 66.37%, respectively, for the same number of tasks per service.

C. Execution Time Costs

The performance of EMSPA is computed with respect to RS through numerical evaluation in terms of total execution time, which is presented in Fig. 6a and Fig. 6b for *Scenario 1* and *Scenario 2*, respectively. Both EMSPA and RS exhibit similar completion times across configurations.

This is expected, as EMSPA primary objective is not to minimize completion time, and the inherent randomness in

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

task processing durations and AGV movement for loading operations introduces variability in the completion times for both algorithms. However, the results indicate an average increase in EMSPA with respect to RS of 0.34% for *Scenario 1*, attributed to the additional processing overhead required by EMSPA for each task placement decision. Results from *Scenario 1* show a lineal increase in total execution time as the number of services is scaled, which demonstrates EMSPA's low complexity. Fig. 6b shows that *Scenario 2* also exhibits an increasing trend in completion time with a greater number of tasks per service, although the number of AGVs does not significantly impact the overall simulation completion time. Although a higher number of AGVs leads to a higher service acceptance rate, the dynamic arrival of services can still result in increased completion times. Consequently, the completion times remain quite similar for simulations with the same number of tasks per service, regardless of the number of AGVs. For *Scenario 2*, the time taken by EMSPA is on average 0.63% higher than for RS. QASDMS outperforms EMSPA in completion time under non-saturated scenarios, with mean and maximum reductions of 0.25% and 2.29%, respectively, in *Scenario 1*; and of 0.12% and 2.20%, respectively, in *Scenario 2*, demonstrating the favorable trade-off between energy and time efficiency achieved by EMSPA. QASDMS incurs higher times when the system reaches saturation, in exchange for increased SAR; in such scenarios, EMSPA achieves the most favorable efficiency in terms of energy and time efficiency.

VII. Conclusion

This work proposed EMSPA, a service placement approach for multi-AGV systems considering loading and processing tasks under dynamic task arrivals. The algorithm was designed to enhance energy efficiency while satisfying computational and communication constraints. Simulation results show that EMSPA outperform a RS method, achieving up to 2.34% lower normalized energy consumption compared to RS and up to 16.09% improvement in service acceptance rate. Further, compared to the QASDMS approach, EMSPA reduced the processing power ratio by over 58.94% at a cost of less than 2.29% increase in completion time. These results validate EMSPA as an effective energy-aware service placement solution for heterogeneous multi-AGV environments. Future work will extend EMSPA toward global optimization, by incorporating batch-based task placement as well as predictive knowledge about future task arrivals, leading to an increased global system performance.

References

[1] H. Fan *et al.*, "Improving scheduling in multi-AGV systems by task prediction," *J Scheduling*, vol. 27, no. 3, pp. 299–308, Jun 2024.

[2] J. Leng *et al.*, "Profiling power consumption in low-speed autonomous guided vehicles," *IEEE Rob. Autom. Lett.*, vol. 9, no. 7, pp. 6027–6034, Jul. 2024.

[3] M. Masoumi *et al.*, "Dynamic joint scheduling of movement and data processing tasks using extreme-edge computing in multi-AGV scenarios," *IEEE Open J. Ind. Electron. Soc.*, vol. 6, pp. 1312–1334, 2025.

[4] P. Liu *et al.*, "Task offloading optimization for AGVs with fixed routes in industrial IoT environment," *China Commun.*, vol. 20, no. 5, pp. 302–314, 2023.

[5] T. Zhang *et al.*, "Modeling an optimal environmentally friendly energy-saving flexible workshop," *Appl. Sci.*, vol. 13, no. 21, p. 11896, 2023.

[6] Z. Jiang *et al.*, "Grid-map-based path planning and task assignment for multi-type AGVs in a distribution warehouse," *Mathematics*, vol. 11, no. 13, p. 2802, 2023.

[7] Y. Qiao *et al.*, "Communication–control co-design in wireless networks: A cloud control AGV example," *IEEE Internet Things J.*, vol. 10, no. 3, pp. 2346–2359, Feb. 2023.

[8] J. Bergmann *et al.*, "Adaptive AGV fleet management in a dynamically changing production environment," *Procedia Manuf.*, vol. 54, pp. 148–153, Jan 2021.

[9] D. Johnson *et al.*, "Multi-agent reinforcement learning for real-time dynamic production scheduling in a robot assembly cell," *IEEE Rob. Autom. Lett.*, vol. 7, no. 3, pp. 7684–7691, 2022.

[10] Y. Li *et al.*, "Efficient task planning for heterogeneous AGVs in warehouses," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–15, 2024.

[11] L. He *et al.*, "A multiobjective evolutionary algorithm for achieving energy efficiency in production environments integrated with multiple automated guided vehicles," *Knowl-based syst.*, vol. 243, p. 108315, 2022.

[12] L. Hu *et al.*, "Energy benchmark for energy-efficient path planning of the automated guided vehicle," *Sci. Total Environ.*, vol. 857, p. 159613, 2023.

[13] W.-Q. Zou *et al.*, "Efficient multiobjective optimization for an AGV energy-efficient scheduling problem with release time," *Knowl-based syst.*, vol. 242, p. 108334, 2022.

[14] S. Riazi *et al.*, "Energy optimization of large-scale AGV systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 2, pp. 638–649, 2021.

[15] G. Bohács *et al.*, "Integrating scheduling and energy efficiency aspects in production logistic using AGV systems," *IFAC-PapersOnLine*, vol. 54, no. 1, pp. 294–299, 2021.

[16] J. Palomares *et al.*, "Toward Field-Level Device Orchestration in Industrial Multiaccess Edge Computing Deployments: A Unified IT–OT Framework," *IEEE Ind. Electron. Mag.*, pp. 2–12, 2025.

[17] M. De Ryck *et al.*, "Automated guided vehicle systems, state-of-the-art control algorithms and techniques," *J. Manuf. Syst.*, vol. 54, pp. 152–173, 2020.

[18] C. Sauer *et al.*, "Testing AGV mobility control method for MANET coverage optimization using procedural simulation," *Comput. Commun.*, vol. 194, pp. 189–201, Oct. 2022.

[19] L. Qiu *et al.*, "Scheduling and routing algorithms for agvs: A survey," *Int. J. Prod. Res.*, vol. 40, no. 3, pp. 745–760, 2002.

[20] S. Boyd *et al.*, *Convex Optimization*. Cambridge University Press, 2004.

[21] Intel, "Intel® Core™ i7-8700 Processor (12M Cache, up to 4.60 GHz) - Product Specifications," accessed: 2025-09-02. [Online]. Available: https://www.intel.com/content/www/us/en/products/sku/126686/intel-core-i78700-processor-12m-cache-up-to-4-60-ghz/specifications.html

[22] NVIDIA, "Gama GeForce RTX 3090," accessed: 2025-09-02. [Online]. Available: https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/rtx-3090-3090ti/

[23] ——, "NVIDIA Jetson Xavier," accessed: 2025-14-04. [Online]. Available: https://www.nvidia.com/en-gb/autonomous-machines/embedded-systems/jetson-xavier-series/

[24] A. Cavilla *et al.*, "Simplified simulation models for indoor MANET evaluation are not robust," in *Proc. of IEEE SECON*, Santa Clara, CA, USA, 2004, pp. 610–620.

[25] N. Corporation, "Isaac ROS benchmark results," 2024, accessed: 2025-12-05. [Online]. Available: https://github.com/NVIDIA-ISAAC-ROS/isaac_ros_benchmark/tree/main/results

[26] Y. Fang *et al.*, "Digital-twin-based job shop scheduling toward smart manufacturing," *IEEE Trans. Ind. Inform.*, vol. 15, no. 12, pp. 6425–6435, 2019.

[27] X. Gong *et al.*, "Toward safe and efficient human–swarm collaboration: A hierarchical multi-agent pickup and delivery framework," *IEEE Trans. Intell. Veh.*, vol. 8, no. 2, pp. 1664–1675, 2023.

[28] F. Xu *et al.*, "Multi-objective fog node placement strategy based on heuristic algorithms for smart factories," *Wirel. Netw.*, vol. 30, no. 6, pp. 5407–5424, 2024.