The EALS Framework: A Multi-Objective Approach to Robust Leader Election in Robotic Teams

Arnav Gupta
IIIT Delhi, India
Email: arnav21236@iiitd.ac.in

***Abstract*—In dynamic multi-robot systems, suboptimal leader election can trigger mission failure, communication delays, and catastrophic energy depletion. We introduce EALS, a modular and intelligent leader election framework that adapts to real-world constraints, including battery drain, task overload, spatial dispersion, and communication fragility. At its core, EALS minimizes a quadratic energy inefficiency metric. Adaptive variants Proximity-, Taskload-, Comms-, and Reliability-EALS further enrich the decision-making process with contextual data. By leveraging metric normalization, rank aggregation, and Pareto-optimal tie-breaking, EALS delivers robust, explainable leadership decisions with minimal churn. Validated through a three-pronged strategy of formal verification, rigorous simulation benchmarking, and decentralized deployment in ROS2, EALS and its variants reduce leader churn by up to 87% compared to baseline strategies, significantly improving system stability and mission longevity.**

***Index Terms*—Leader election, multi-robot systems, energy-aware coordination, ROS2, adaptive robotics, optimization, Z3 solver.**

I. Introduction

In dynamic multi-robot systems, the choice of a leader is a high-stakes decision where a single suboptimal choice can cascade into mission failure. This vulnerability exists because classical leader election algorithms, such as the Bully algorithm [1] or ring-based [2] protocols, are often ill-suited for the dynamic realities of robotic deployments. Their foundational assumptions of static, synchronous networks are fundamentally broken by **physical embodiment**, where abstract metrics have tangible consequences. **Energy** is not merely a number but a direct constraint on mission duration; **communication** reliability is dictated by the physics of mobility and environmental obstruction; and **computational load** directly impacts a robot's ability to perform real-time motion and perception tasks. This creates a critical gap for a leadership paradigm that actively confronts the challenges of real-world physical constraints—a domain where naive and classical approaches fall short.

To address this gap, we contend that leader election must be reframed as a multi-objective optimization problem that adapts to system heterogeneity and real-time conditions. We propose the **Energy-Aware Leadership Selection (EALS)** framework, which realizes this philosophy. EALS formalizes the selection process through a rank-based optimization of a quadratic energy inefficiency metric. This formulation strongly penalizes robots with both low battery reserves and high consumption rates, thereby capturing the non-linear risk of mission failure posed by unsustainable nodes. This core metric is complemented by a robust decision-making pipeline using normalization, metric aggregation, and Pareto-optimal tie-breaking to ensure fair and interpretable selections.

The modularity of EALS allows it to be extended with additional context-aware metrics, enhancing its adaptability. We introduce and evaluate four such variants:

- **Proximity-EALS**, which favors spatially central robots to reduce communication delays,
- **Taskload-EALS**, which deprioritizes overloaded agents to avoid computational bottlenecks,
- **Comms-EALS**, which prioritizes robots with high link quality for communication robustness,
- **Reliability-EALS**, which avoids agents with a history of frequent failures.

Each variant integrates domain-relevant metrics into a consistent multi-objective decision-making pipeline.

This work makes the following primary contributions:

1) We propose a unified, modular framework for leader election that incorporates energy efficiency, spatial awareness, workload distribution, communication quality, and historical reliability.
2) We validate our approach across three complementary layers: (i) formal verification using the Z3 SMT solver, (ii) controlled simulation for rigorous quantitative benchmarking, and (iii) a decentralized implementation in ROS2 to prove architectural viability.
3) We demonstrate that EALS and its variants significantly reduce leader churn, improve system robustness, and maintain stable coordination under uncertainty.

Together, these contributions offer a principled yet practical solution for deploying robust, adaptive, and efficient multi-robot teams in the real world.

II. Related Work

A. Classical Strategies in Distributed Systems

Leader election is a foundational problem in distributed computing, with early algorithms like the Bully algorithm [1] and ring-based protocols [2] establishing methods for ensuring a unique leader in static, reliable networks. However, their core assumptions of synchrony and fault-free communication are fundamentally mismatched with the realities of mobile robotics. In robotic deployments, dynamic topologies, frequent disconnections, and asynchronous communication render these classical approaches unstable and inefficient, leading to high re-election costs and system stalls.

B. *Energy and Communication-Aware Methods*

To address the constraints of mobile systems, researchers developed methods focusing on specific physical metrics. In wireless sensor networks (WSNs), protocols like LEACH [3] and KELEA [4] introduced energy-awareness, electing leaders based on residual battery to prolong network lifetime. Similarly, methods like WCA [5] and UAV-based voting schemes [6] prioritize communication awareness by selecting centrally-located agents to minimize latency. While effective, these approaches often optimize for a **single objective** (e.g., energy *or* connectivity), failing to address the multi-faceted nature of robotic missions where these constraints are tightly coupled and must be considered simultaneously.

C. *Optimization and Learning-Based Approaches*

More advanced techniques frame leader election as a formal optimization or learning problem. Methods using Integer Linear Programming (ILP) [7] or Satisfiability Modulo Theories (SMT) [8] can compute optimal leaders but often lack the scalability and real-time performance required for dynamic teams. Conversely, learning-driven approaches using deep reinforcement learning (RL) [9], fuzzy logic [10], or trust-based models [11], [12] offer adaptability but often function as ”black boxes,” lacking the **interpretability and predictability** crucial for safety-critical robotic applications. EALS is designed to bridge this gap, offering a deterministic, explainable, and multi-objective framework that remains computationally tractable for real-time deployment.

III. Motivating Example

Consider a team of five robots ($R_1 - R_5$) executing a collaborative indoor mapping mission, as depicted in Figure 1. The team is heterogeneous: R_3 has a high-drain sensor, making it energy-inefficient; R_4 is at the edge of the WiFi range with an unstable communication link; and R_1 is a spatial outlier, far from the team's center of mass.

A naive leader election strategy, such as Round-Robin or one that only considers battery percentage, would ignore this critical context. The process might first elect R_3 due to its high initial battery. However, its high drain rate would quickly deplete its power, halting its task, and forcing a costly system-wide re-election. In the next round, the algorithm might select R_4, whose unstable link then causes coordination failures and lost data packets. These consecutive, predictable failures, driven by ignoring context, result in significant mission delays and wasted energy.

This scenario highlights the need for an adaptive framework that can holistically evaluate candidates across multiple, often competing, criteria. An ideal leader should be **energy-efficient**, **computationally available**, **centrally located**, and **communicatively robust**. This is precisely the multi-objective optimization problem that our proposed EALS framework is designed to solve.

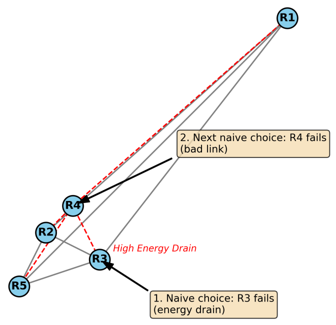
R1
2. Next naive choice: R4 fails (bad link)
R4
R2
High Energy Drain
R3
R5
1. Naive choice: R3 fails (energy drain)

Fig. 1. A motivating scenario for adaptive leader election. Naive strategies sequentially select unsuitable leaders (R_3, R_4), leading to cascading failures. An intelligent framework like EALS would identify the more stable and centrally-located R_2 or R_5 as superior candidates.

IV. Proposed Methodology

This section formulates the Energy-Aware Leadership Selection (EALS) algorithm and its adaptive variants for robust leader election in multi-robot systems. The core method selects the most energy-efficient robot, while variants incorporate metrics like spatial position, task load, communication quality, and reliability. All follow a unified decision framework: compute metrics, normalize, rank, and resolve ties using Pareto-optimality.

A. *Energy-Aware Leadership Selection (EALS)*

Consider a team of N robots indexed by $i \in \{1, 2, \ldots, N\}$. The baseline EALS algorithm selects a leader based on energy efficiency, using the following variables:

- $B_i \in [0, 100]$: Battery level of robot i (percentage).
- $D_i \in \mathbb{R}^+$: Battery drain rate of robot i, defined as $\frac{dB_i}{dt}$.
- $L_i \in \{0, 1\}$: Binary variable indicating leadership; $L_i = 1$ if robot i is selected.

M_i: Energy cost metric for robot i, defined as:

$$M_i = (100 - B_i) \cdot D_i^2 \tag{1}$$

Here, $(100 - B_i)$ penalizes low-battery robots, while the quadratic term D_i^2 imposes a stronger penalty on high-drain nodes, reflecting the non-linear risk of rapid depletion. This design discourages unsustainable leaders and improves long-term stability.

To enable comparison, metrics are normalized as:

$$\tilde{M}_i = \frac{M_i - \min_j M_j}{\max_j M_j - \min_j M_j + \varepsilon} \tag{2}$$

where $\varepsilon \ll 1$ (e.g., 10^{-6}) prevents division by zero and ensures stability.

The leader election problem becomes:

$$\min_i \tilde{M}_i \quad \text{subject to} \quad \sum_{i=1}^{N} L_i = 1, \quad L_i \in \{0,1\} \tag{3}$$

If multiple robots yield the same minimum $\tilde{M}_i$, Pareto-optimal selection is applied to the corresponding $\tilde{M}_i$ values (see Section IV-F for details). If ties remain within the Pareto front, the robot with the lowest $\tilde{M}_i$ is selected.

B. Proximity-Aware Leadership (Proximity-EALS)

In spatially distributed teams, peripheral robots often face higher latency and message loss. To address this, the Proximity-EALS variant prioritizes spatially central robots as leaders, improving coordination efficiency and reducing communication overhead.

For each robot i, the average distance to all other robots is computed as:

$$P_i = \frac{1}{N-1} \sum_{\substack{j=1 \\ j\neq i}}^{N} d(i,j) \tag{4}$$

where $d(i,j)$ is the Euclidean distance between robots i and j. Lower P_i values indicate spatial centrality, making such robots preferable for leadership.

Proximity metrics are normalized via:

$$\tilde{P}_i = \frac{P_i - \min_j P_j}{\max_j P_j - \min_j P_j + \varepsilon} \tag{5}$$

where $\varepsilon \ll 1$ (e.g., 10^{-6}) prevents division by zero and ensures stability.

Final ranking score combines energy and proximity ranks:

$$R_i = \text{rank}(\tilde{M}_i) + \text{rank}(\tilde{P}_i) \tag{6}$$

If multiple robots share the same R_i, Pareto-optimal selection is applied to the tuples $(\tilde{M}_i, \tilde{P}_i)$. Ties within the Pareto front are resolved by selecting the robot with the lowest $\tilde{M}_i$ (see Section IV-F for details).

C. Task Load-Aware Leadership (Taskload-EALS)

To prevent selection of overburdened robots as leaders, we introduce a task load metric T_i for each robot i, representing the extent of computational or functional load. T_i can be measured via indicators such as CPU usage, task queue length, or number of concurrent jobs.

- If a robot is heavily loaded, T_i is high (undesirable for leadership).
- If a robot is underutilized, T_i is low (preferred).

To compare across robots, task loads are normalized:

$$\tilde{T}_i = \frac{T_i - \min_j T_j}{\max_j T_j - \min_j T_j + \varepsilon} \tag{7}$$

where $\varepsilon \ll 1$ (e.g., 10^{-6}) prevents division by zero and ensures stability.

The final score aggregates energy and task load ranks:

$$R_i = \text{rank}(\tilde{M}_i) + \text{rank}(\tilde{T}_i) \tag{8}$$

In the case of tied R_i values, Pareto-optimal selection is applied to the tuples $(\tilde{M}_i, \tilde{T}_i)$, selecting robots not strictly dominated in energy cost and task load. If multiple Pareto-optimal candidates remain, the one with the lowest $\tilde{M}_i$ is chosen (see Section IV-F for details).

D. Communication-Robust Leadership (Comms-EALS)

In lossy or bandwidth-constrained environments, unstable communication links can hinder effective coordination. Comms-EALS prioritizes robots with strong communication quality by introducing a communication cost metric:

$$C_i = \frac{1}{Q_i + \varepsilon} \tag{9}$$

where Q_i denotes a communication quality indicator (e.g., RSSI or packet delivery ratio), and $\varepsilon \ll 1$ (e.g., 10^{-6}) prevents division by zero and ensures stability.

The interpretation is as follows:

- A high Q_i (better communication) yields a low C_i (preferred).
- A low Q_i results in a high C_i (penalized).

Normalized communication scores are computed as:

$$\tilde{C}_i = \frac{C_i - \min_j C_j}{\max_j C_j - \min_j C_j + \varepsilon} \tag{10}$$

where $\varepsilon \ll 1$ (e.g., 10^{-6}) prevents division by zero and ensures stability.

The final selection score combines energy and communication rankings:

$$R_i = \text{rank}(\tilde{M}_i) + \text{rank}(\tilde{C}_i) \tag{11}$$

If multiple robots share the same R_i, Pareto-optimal selection is applied over $(\tilde{M}_i, \tilde{C}_i)$ (see Section IV-F). The tie is resolved by selecting the robot with the lowest $\tilde{M}_i$.

E. Failure-History Weighted Leadership (Reliability-EALS)

In long-running or safety-critical missions, assigning leadership to unstable robots increases coordination risks. Reliability-EALS addresses this by favoring historically stable agents and avoiding nodes with frequent failures or erratic behavior.

Each robot i is assigned a failure score $F_i \in \mathbb{R}^+$, capturing metrics such as:

- Unexpected shutdowns or resets (e.g., watchdog timeouts),
- Battery levels dropping below safe thresholds,
- Disconnection events or loss of heartbeat messages.

Higher values of F_i indicate lower reliability. After normalization, the ranking score is:

$$\tilde{F}_i = \frac{F_i - \min_j F_j}{\max_j F_j - \min_j F_j + \varepsilon} \tag{12}$$

where $\varepsilon \ll 1$ (e.g., 10^{-6}) prevents division by zero and ensures stability.

The final selection score combines energy and failure rankings:

$$R_i = \text{rank}(\tilde{M}_i) + \text{rank}(\tilde{F}_i) \tag{13}$$

If multiple robots share the same R_i, Pareto-optimal selection is applied over $(\tilde{M}_i, \tilde{F}_i)$. Remaining ties are broken by selecting the robot with the lowest $\tilde{M}_i$ (see Section IV-F).

F. Pareto-Optimal Tie Resolution

In cases where multiple candidates share the same aggregated rank score R_i, a tie-breaking mechanism based on Pareto optimality is employed to ensure a fair and robust selection from among the equally-ranked options.

The principle is based on the concept of dominance. A candidate robot j is said to *dominate* another candidate i if it is strictly superior in at least one objective metric and no worse in all others. Let $\tilde{A}_i$ represent the vector of all normalized cost metrics for robot i. Formally, candidate j dominates i if:

$$\tilde{A}_j \preceq \tilde{A}_i \quad \text{and} \quad \tilde{A}_j \neq \tilde{A}_i$$

where the $\preceq$ operator signifies that every element in vector $\tilde{A}_j$ is less than or equal to its corresponding element in $\tilde{A}_i$.

This hierarchical approach ensures that the selection is not only balanced across multiple competing criteria but is also aligned with the system's primary goal of maximizing operational lifetime.

V. Implementation and Validation Methodology

The EALS framework is validated using a rigorous, three-pronged methodology designed to ensure it is theoretically sound, empirically performant, and practically deployable. This strategy encompasses: (i) formal verification of the algorithm's correctness using the Z3 SMT solver; (ii) quantitative benchmarking in a controlled, multi-threaded Python simulator; and (iii) architectural validation in a decentralized ROS2 implementation. While each layer is executed independently, they share a unified decision logic, enabling a comprehensive cross-layer validation of our approach.

A. Formal Correctness Verification with Z3

To formally guarantee the logical correctness of our framework, EALS variants are encoded as Satisfiability Modulo Theories (SMT) problems and solved using Z3 [8]. Robot-specific variables such as the selection indicator L_i, battery level B_i, and task load T_i are defined as assertions. Key logical constraints are then enforced:

- **Unique Leader:** Exactly one leader must be selected: $\sum_{i=1}^{N} L_i = 1$.
- **Safety Constraints:** A robot cannot be a leader if its battery is below a critical threshold: $L_i = 1 \Rightarrow B_i > 10$.
- **Optimality:** The selected leader must have the optimal metric value: $L_i = 1 \Rightarrow \tilde{M}_i \leq \tilde{M}_j, \forall j \neq i$.

This layer acts as a formal oracle, verifying that the core logic of EALS and its variants is correct across all possible configurations.

B. Controlled Simulation for Quantitative Analysis

To rigorously benchmark performance, we developed a multi-threaded Python simulator. This custom environment enables repeatable experiments with precise control over system dynamics and noise injection—conditions that are difficult to isolate and reproduce on physical hardware. Each robot operates as a thread, managing its own state (e.g., battery, load). A central `LeadershipManager` synchronizes states and executes the configured leader election algorithm.

The simulator supports all EALS variants and baselines (Random, Greedy, Round-Robin) and is used to evaluate key metrics like leader churn and stability across various team sizes ($N \in \{5, 10, 50\}$) and adversarial scenarios, such as sudden failures or extreme energy drain profiles.

C. Architectural Validation in ROS2

To confirm EALS's practical deployability in a standard, real-world robotics middleware, we implemented the framework in ROS2. In this architecture, each robot runs an independent EALS node. Nodes publish their local state (battery, task load, etc.) to a shared topic (e.g., `/robot_status`) and subscribe to the states of their peers.

Crucially, the leader election logic is executed locally and deterministically on every robot, ensuring that all nodes converge on the same leader without a central coordinator. The elected leader's ID is then broadcast on a `/leader_id` topic. This decentralized design proves the framework's fault tolerance and real-time operational feasibility.

D. Cross-Layer Validation Strategy

Our methodology ensures comprehensive validation by assigning a distinct role to each layer. The **Z3 layer** provides formal guarantees of logical correctness. The **Python simulator** serves as the platform for rigorous, quantitative performance benchmarking. Finally, the **ROS2 implementation** validates the framework's architectural soundness and feasibility for decentralized, real-world deployment. This integrated strategy ensures that EALS is not only theoretically sound but also empirically robust and practically viable.

VI. Experimental Evaluation

We conduct a series of experiments within our custom Python simulator to validate the EALS framework. Our evaluation is designed to answer three primary questions: **(1)** Does EALS significantly improve leadership stability compared to standard baselines? **(2)** How do its core metric and contextual variants drive intelligent, adaptive decisions? **(3)** Is the framework robust to the noise and uncertainty inherent in real-world deployments?

A. Experimental Setup and Adversarial Scenario

All experiments are run for a simulated duration of 120 seconds, with leader elections occurring every 0.2 seconds. We designed a challenging, heterogeneous five-robot scenario to deliberately stress-test the algorithms' decision-making capabilities. The initial parameters, detailed in Table I, create specific failure modes for naive algorithms:

- **Energy Trap (R3):** A high drain rate makes it an unsustainable leader, despite starting with high battery.
- **Workload Trap (R4):** The highest task load makes it a poor choice for handling leadership overhead.
- **Isolation Trap (R1):** A peripheral location makes it a communication bottleneck.

TABLE I
INITIAL PARAMETERS FOR THE ADVERSARIAL SCENARIO

Robot ID	Battery (%)	Drain Rate	Task Load	Position (x, y)
R0	92	12	5	(0.0, 0.0)
R1 (Outlier)	93	15	3	(20.0, 20.0)
R2 (Stable)	95	10	7	(0.1, 0.3)
R3 (High-Drain)	95	30	2	(0.3, 0.1)
R4 (High-Load)	94	16	8	(0.2, 0.2)

B. EALS Drastically Reduces Leadership Churn

We first evaluate overall system stability by measuring leader churn—the cumulative count of leader changes. High churn indicates instability and wasted energy from frequent re-elections. As shown in Figure 2, the entire EALS family of algorithms maintains a low and stable churn rate. In contrast, the churn from naive baselines like RANDOM and ROUND ROBIN grows excessively, highlighting their unsuitability for dynamic missions.

Table II quantifies this stability. **PROXIMITY-EALS** emerges as the most stable algorithm with only 72 leader changes, an **87% reduction** compared to the 565 changes from ROUND ROBIN. This confirms that context-aware decision-making is critical for stable leadership.

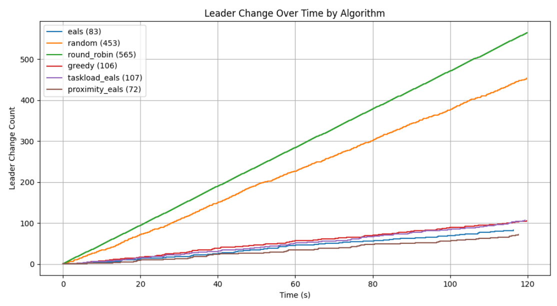

Fig. 2. Cumulative leader changes over time across algorithms. Lower counts indicate better stability.

C. Dissecting the Decision-Making Process

To understand *why* EALS is more stable, we analyze the leader selection frequency for different algorithms. Figure 3 shows how each strategy interacts with our adversarial scenario.

- **EALS** correctly identifies and avoids the **Energy Trap**. It consistently penalizes the high-drain Robot 3, preferring the more sustainable, energy-efficient Robot 2.

TABLE II
COMPARISON OF LEADER ELECTION ALGORITHMS ($N = 5$, DURATION = 120S)

Algorithm	Leader Changes	Leader Bias	Churn	Remarks
EALS	83	High on R2	Low	Avoids high-drain robots like R3.
TASKLOAD-EALS	107	High on R1, R3	Moderate	Avoids high task load robots (e.g. R4); slightly higher churn.
PROXIMITY-EALS	72	High on R2, R4	Very Low	Best stability; avoids spatial outliers (e.g. R1)
GREEDY	106	Balanced	Moderate	Re-elects R3 frequently due to recharge loops.
RANDOM	453	Uniform	Very High	No contextual awareness; highly volatile.
ROUND ROBIN	565	Uniform	Extreme	Worst-case churn; cyclic switching ignores robot state.

- **Taskload-EALS** avoids the **Workload Trap**. It successfully penalizes the overloaded Robot 4 (task load of 8), distributing leadership to less-burdened agents like R1 and R3.
- **Proximity-EALS** avoids the **Isolation Trap**. It heavily penalizes the distant Robot 1, selecting centrally-located robots like R2 and R4 to ensure efficient communication.
- **Greedy**, a myopic baseline, repeatedly falls into the **Energy Trap**. It selects the high-drain Robot 3 whenever its battery is full, which leads to a predictable and inefficient feedback loop:
  1) Robot 3 is elected due to its momentarily full battery.
  2) Its high drain rate causes it to fail quickly.
  3) Upon a simulated recharge, it again becomes the most attractive candidate to the naive algorithm.
  4) The Greedy strategy re-elects it, restarting the cycle.

  This predictable loop highlights the unsustainability of purely myopic selection schemes and underscores the need for predictive metrics like drain rate, which EALS incorporates.

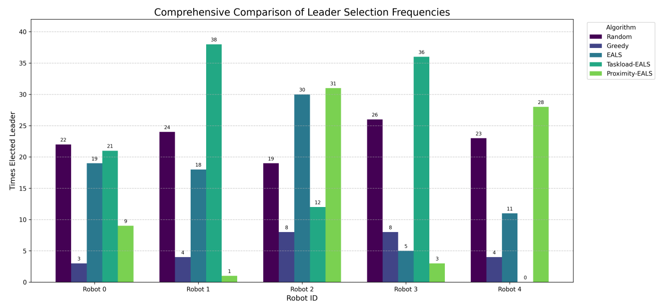

Fig. 3. Leader selection frequency for different algorithms. Each EALS variant intelligently avoids the specific "trap" it is designed for, while the GREEDY baseline repeatedly selects the unsustainable Robot 3.

D. Robustness Under Uncertainty

To validate the framework's performance in non-ideal conditions, we conducted a stress test by introducing environmental uncertainty into the simulation. This test was designed to mimic real-world inconsistencies such as sensor drift, lossy networks, and hardware resets. We applied three types of perturbations: **(i) Sensor Noise**, with Gaussian noise added to battery ($\mathcal{N}(0, 3\%)$) and task load ($\mathcal{N}(0, 1.0)$) telemetry; **(ii) Communication Dropout**, where each robot had a 10% chance of skipping a state update per timestep; and **(iii) Transient Failures**, where robots would randomly fail and recover.

We compare the performance of all algorithms under uncertainty for $N = 5$ and $N = 50$. The main metrics recorded are total leader changes (churn), robustness to missing data, and leader stability.

TABLE III
LEADER CHURN UNDER UNCERTAINTY ($N = 5, T = 120s$)

Algorithm	Churn ↓	Stability Under Noise
PROXIMITY-EALS	76	Highest robustness; avoids spatial outliers
EALS	111	Slightly degraded from clean case (83)
TASKLOAD-EALS	118	More sensitive to task noise and dropouts
GREEDY	127	Over-reacts to noisy battery values
RANDOM	479	Unstable under all conditions
ROUND ROBIN	557	Deterministic cycling unaffected

The results of this evaluation, shown in Table III, reveal the resilience of our context-aware approach. The EALS variants remain highly effective, with **PROXIMITY-EALS** demonstrating remarkable stability (76 changes), as its reliance on spatial centrality is less affected by telemetry noise. While the performance of EALS and TASKLOAD-EALS slightly degraded, they still significantly outperform the baselines. In stark contrast, myopic algorithms like GREEDY and RANDOM become even more unstable, as their decision-making is easily misled by noisy data, confirming that contextual filtering is crucial for robust performance in unpredictable environments.

E. Synthesis of Experimental Results

Our comprehensive evaluation demonstrates that the EALS framework delivers consistently more stable, intelligent, and robust leadership decisions than traditional baselines. The core **EALS** algorithm's quadratic penalization of drain rate successfully prevents the unstable re-election loops that plague myopic strategies like GREEDY. Furthermore, the adaptive variants show the benefit of multi-objective optimization: **PROXIMITY-EALS** achieves the highest stability across all scenarios by leveraging spatial data, while **TASKLOAD-EALS** effectively balances leadership against computational load, albeit with some sensitivity to noisy telemetry. The extreme churn exhibited by **RANDOM** and **ROUND ROBIN** confirms their unsuitability for anything beyond simple scenarios. Overall, the proposed EALS framework—augmented by contextual metrics—delivers a scalable, interpretable, and fault-tolerant solution for leader election in distributed multi-robot systems.

VII. DISCUSSION

The EALS framework reconciles the competing objectives of energy-aware optimization and real-time adaptability within multi-robot systems. Its modular design, which integrates formal verification, simulation, and distributed deployment, facilitates comprehensive validation spanning theoretical guarantees and practical applicability. This section provides a detailed analysis of the framework's metric design trade-offs, its robustness to uncertainty, emergent behaviors observed in simulation, and the implications of its real-time deployment.

A. Metric Design and Trade-offs

EALS employs a computationally simple and interpretable metric to prioritize energy-efficient leadership. The quadratic penalty applied to robots with high energy drain rates effectively stabilizes leadership roles and extends the system's operational lifetime. However, practical deployments often exhibit heterogeneity, which the adaptive variants (Taskload-EALS and Proximity-EALS) address by incorporating supplementary contextual metrics. While these adaptations enhance selection quality, they introduce computational overhead. Proximity-EALS, for instance, incurs an $O(n^2)$ runtime complexity, which presents a scalability challenge for large-scale swarms and motivates future work on approximate centrality estimators.

B. Robustness to Uncertainty

Effective leader election must exhibit tolerance to noise in sensing, communication, and dynamics. The EALS framework maintains robustness through metric normalization, rank-based aggregation, and Pareto filtering. For example, Proximity-EALS mitigates instability by favoring spatially central agents with more stable communication links, while Taskload-EALS avoids overburdened robots, minimizing failure risks.

These design principles were validated in our ablation study under simulated noisy conditions. The results confirm that context-aware variants, most notably Proximity-EALS, maintain a high degree of performance robustness. Even with injected sensor noise and communication dropouts, the frequency of leader transitions remained low. This confirms the framework's operational resilience to real-world uncertainties and that its structured, interpretable metrics outperform myopic policies that are easily misled by degraded telemetry.

C. Behavioral Insights from Simulation

The simulation environment revealed key emergent behaviors that validate our design. A baseline greedy approach resulted in the cyclical re-election of high-drain robots, creating inefficient leadership oscillations. In contrast, the penalty term in the EALS metric effectively suppressed these oscillations, significantly reducing leader transitions. Furthermore, the adaptive variants demonstrated an ability to selectively avoid agents with high task loads or spatial isolation. These behaviors confirm that a principled metric design translates directly into desirable operational characteristics, including lower churn and improved energy balancing.

D. Real-Time Validation via ROS2

To validate practical deployability, EALS was implemented in a decentralized ROS2 architecture. Each robot publishes its state telemetry and independently executes the election logic, ensuring all nodes converge on the same leader without a central bottleneck. This distributed design confers fault tolerance, as the unavailability of a leader triggers a local re-election process, and modularity, as robots can dynamically enter or exit the system. The successful deployment confirms that EALS is suitable for distributed, resilient, real-world multi-robot systems.

E. Limitations and Assumptions

Notwithstanding the promising results, the framework is subject to several limitations. The $O(n^2)$ complexity of Proximity-EALS poses a scalability challenge. The metric weights are currently static and pre-defined; a dynamic adjustment mechanism could enhance adaptability. The framework also operates under the assumption of truthful metric reporting and lacks defenses against malicious agents. Finally, its efficacy is contingent upon the availability of high-fidelity telemetry data. Addressing these limitations is a critical prerequisite for deployment in adversarial or resource-constrained environments.

VIII. Conclusion and Future Work

This work presented the EALS framework, a principled, interpretable, and modular approach to leader election in distributed multi-robot systems. By framing the problem as a multi-objective decision, EALS integrates energy efficiency with task load, spatial proximity, and reliability. Validated through formal methods, simulation, and a ROS2 deployment, EALS consistently outperformed baseline strategies, reducing leader churn and maintaining robustness under uncertainty. The framework's flexibility, robustness, and interpretability position it as a promising candidate for next-generation coordination strategies in mission-critical applications across domains such as UAV swarms and planetary exploration.

Several promising directions can extend this work:

- **Learning-Augmented Adaptation:** The deterministic nature of EALS makes it a suitable foundation for hybridization with machine learning. We plan to replace static rank aggregation with adaptive weight tuning, using reinforcement learning to optimize weights based on mission-level feedback. Supervised models could also predict imminent failures from sensor data to enhance the reliability metric.
- **Scalable Proximity and Trust:** To address computational bottlenecks, we will explore approximate methods for proximity calculation, such as spatial hashing or clustering, to reduce the $O(n^2)$ cost. We will also incorporate robustness to false metric reporting through trust modeling and Byzantine-resilient consensus mechanisms.
- **Physical Hardware Deployment:** The final step is to deploy EALS on physical robot platforms, such as a swarm of UAVs, to evaluate its performance under real-world noise, network latency, and physical control loop constraints.

References

[1] H. Garcia-Molina, "Elections in a distributed computing system," *IEEE Transactions on Computers*, vol. C-31, no. 1, pp. 47–59, 1982.

[2] E. Chang and R. Roberts, "An improved algorithm for decentralized extrema-finding," *Communications of the ACM*, vol. 22, no. 5, pp. 281–283, 1979.

[3] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. 33rd Hawaii International Conference on System Sciences (HICSS)*, 2000, pp. 10–17.

[4] A. N. Alslaity and S. A. Alwidian, "A k-neighbor-based, energy aware leader election algorithm (kelea)," *International Journal of Computer Applications*, vol. 59, no. 19, pp. 38–43, 2012.

[5] M. Chatterjee, S. K. Das, and D. Turgut, "Wca: A weighted clustering algorithm for mobile ad hoc networks," *Cluster Computing*, vol. 5, no. 2, pp. 193–204, 2002.

[6] Y. Zuo, W. Yao, Q. Chang *et al.*, "Voting-based scheme for leader election in lead-follow uav swarm with constrained communication," *Electronics*, vol. 11, no. 14, p. 2143, 2022.

[7] Z. Eskandari, S. A. H. Seno, M. Shenify, and R. Budiarto, "Optimal cluster head selection in wireless sensor networks using integer linear programming techniques," *International Journal of Informatics and Communication Technology*, vol. 3, no. 3, pp. 186–196, 2014.

[8] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. Springer, 2008, pp. 337–340.

[9] F. Zhang, Q. Yang, and D. An, "A leader-following paradigm based deep reinforcement learning method for multi-agent cooperation games," *Neural Networks*, vol. 156, pp. 295–307, 2022.

[10] I. Gupta, D. Riordan, and S. Sampalli, "Cluster-head election using fuzzy logic for wireless sensor networks," in *Proc. 3rd Annual Communication Networks and Services Research Conference*, 2005, pp. 255–260.

[11] S. Kundu, S. Misra, and M. S. Obaidat, "Trust-based leader election for resilient drone swarms," *IEEE Transactions on Mobile Computing*, vol. 20, no. 7, pp. 2636–2649, 2021.

[12] A. S. Bogos, X. Fafoutis, P. Andreou *et al.*, "Adlu: A novel anomaly detection and location-attribution algorithm for uwb wireless sensor networks," *EURASIP Journal on Information Security*, vol. 2014, no. 3, pp. 1–16, 2014.