Breaking Randomness: An Intelligent, Adaptive Leadership Algorithm for Energy-Efficient Multi-Robot Teams

Arnav Gupta*, [Add other authors here]
*IIIT Delhi, India
Email: arnav21236@iiitd.ac.in

***Abstract*—In dynamic multi-robot systems, choosing the wrong leader can mean mission failure, communication delays, or energy collapse. We present EALS—a modular, intelligent framework for leader election that adapts to real-world constraints like battery drain, task overload, spatial dispersion, and communication fragility. At its core, EALS minimizes a quadratic energy inefficiency metric, while adaptive variants—Proximity-, Taskload-, Comms-, and Reliability-EALS—infuse awareness of context into the decision-making. Through metric normalization, rank aggregation, and Pareto-optimal tie resolution, EALS delivers robust, explainable leadership decisions with minimal churn. Whether in simulation or real-world deployment, EALS consistently outperforms traditional strategies in stability, fairness, and adaptability.**

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

I. Introduction

Leader election plays a central role in coordinating task allocation, communication schedules, and consensus in distributed multi-robot systems. However, traditional algorithms—such as Bully [1], ring-based [2], or round-robin schemes—are designed for static, synchronous networks and fail to account for the challenges inherent in robotic deployments.

In particular, electing a robot with low residual energy or a high drain rate can lead to frequent re-elections, degraded performance, and reduced mission lifespan. To address these limitations, leader election must be treated as a multi-objective optimization problem that adapts to system heterogeneity and real-time constraints.

We propose the **Energy-Aware Leadership Selection (EALS)** framework, which formalizes leader election as a rank-based optimization process centered on a quadratic energy inefficiency metric. This formulation penalizes robots with both low battery and high energy consumption, capturing the non-linear risk posed by unsustainable nodes. The decision process includes normalization, metric aggregation, and tie-breaking via Pareto-optimality, enabling fair and interpretable selections.

To enhance adaptability, we also introduce four EALS 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.

Contributions. The key contributions of this work are:

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 via a multithreaded Python environment, and (iii) real-time deployment within a distributed ROS2 architecture.
3) We demonstrate that EALS and its variants significantly reduce leader churn, improve system robustness, and maintain stable coordination under uncertainty.

Together, these contributions provide a practical, extensible solution for intelligent leader election in real-world multi-robot deployments.

II. Related Work

A. Classical Strategies in Distributed Systems

Leader election is a foundational problem, with early algorithms like Bully [1] and ring-based protocols [2] ensuring uniqueness in static networks. However, their assumptions of synchrony and reliability do not hold in mobile robotic systems, where frequent disconnections and dynamic topologies lead to instability and high re-election costs.

B. Energy and Communication-Aware Methods

In energy-constrained systems, WSN protocols like LEACH [3], HEED, and KELEA [4] elect leaders based on residual energy or priority scores. For mobile agents, these ideas reduce failures by offloading leadership from depleted robots. Communication-aware methods such as WCA [5], ZePoP, and UAV-based voting [6] select central nodes to improve latency and coordination.

C. Optimization and Learning-Based Approaches

Optimization-based methods (e.g., ILP [7], SMT [8]) offer optimality but often lack scalability. Heuristic, evolutionary, and learning-driven approaches—such as deep RL [9], fuzzy logic [10], and trust-based models [11], [12]—aim to approximate solutions and adapt under uncertainty or adversarial settings.

III. Motivating Example

Consider five robots performing indoor mapping with limited energy and unreliable WiFi. When leadership rotates blindly, R_3 (low battery, high load) and R_4 (unstable link) fail consecutively, delaying progress and wasting energy. Such failures underscore the need for a context-aware, adaptive leader election framework.

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 (Task-EALS)

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

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

To compare across robots, task loads are normalized:

$$\tilde{L}_i = \frac{L_i - \min_j L_j}{\max_j L_j - \min_j L_j + \varepsilon} \tag{7}$$

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

The final score aggregates energy and task load ranks:

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

In the case of tied R_i values, Pareto-optimal selection is applied to the tuples $(\tilde{M}_i, \tilde{L}_i)$, 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

When multiple robots have the same final score R_i, we use Pareto-optimality to break ties fairly. A robot is Pareto-optimal if no other robot is better in all relevant metrics (e.g., energy, proximity, task load).

Formally, robot i is Pareto-optimal if no robot j has lower or equal values for every metric and strictly better in at least one:

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

From the set of Pareto-optimal candidates, we select the robot with the lowest normalized energy cost $\tilde{M}_i$. This method ensures balanced, interpretable, and robust leader selection across multiple competing criteria.

V. Implementation

EALS is implemented using a tri-layer architecture to ensure theoretical soundness, behavioral benchmarking, and real-world viability. The architecture includes: (i) formal verification using the Z3 SMT solver, (ii) a Python-based simulation platform for performance analysis, and (iii) ROS2-based distributed deployment. Each layer operates independently while sharing a unified metric computation and decision logic.

A. Z3-Based Formal Verification

EALS variants are encoded as SMT problems using Z3. Robot-specific variables include the selection indicator L_i, battery level B_i, drain rate D_i, task load T_i, and coordinates (x_i, y_i).

Key constraints include:

- **Unique leader:** $\sum_{i=1}^{N} L_i = 1$
- **Battery safety:** $L_i = 1 \Rightarrow B_i > 10$
- **Metric optimality:** $L_i = 1 \Rightarrow \tilde{M}_i \leq \tilde{M}_j, \forall j \neq i$

Multi-metric variants apply min-max normalization and rank aggregation. A Python–Z3 interface dynamically generates variant-specific constraints. This layer acts as a correctness oracle across diverse configurations.

B. Python-Based Simulation Platform

A multithreaded Python simulator evaluates runtime behavior under controlled, noise-free conditions. Each robot runs as a thread, updating battery, load, and drain rate. A central `LeadershipManager` collects states and applies the configured selection logic.

The simulator supports all EALS variants and baselines (Random, Greedy, Round-Robin). It evaluates scenarios across team sizes ($N \in \{5, 10, 50\}$) and durations (up to 120 seconds), and includes stress-testing with adversarial drain profiles and failure injection. It records metrics such as leader churn, bias, and stability.

C. ROS2-Based Distributed Execution

To test decentralized deployment, EALS is implemented using ROS2. Each robot node publishes its metrics (battery, drain, load, communication) to a shared topic (e.g., `/robot_status`) and subscribes to peer data.

Leader election logic is executed locally and deterministically, ensuring all nodes reach the same decision. The elected leader is broadcast via `/leader_id`, enabling coordinated behavior. Timeout-based fallback handles tie or dropout scenarios. This design removes central dependencies, enabling fault-tolerant, real-time operation.

D. Layered Integration and Validation

Though independently executed, all layers share a common decision pipeline: normalize metrics, aggregate ranks, and resolve ties via Pareto-optimality. The Z3 layer checks constraint logic, the simulator assesses performance and fairness, and the ROS2 system validates distributed feasibility. This integrated design enables rigorous cross-domain validation of EALS.

VI. Experiments and Results

We evaluate the proposed EALS framework and its adaptive variants using a custom Python-based multi-threaded simulator. Each robot is modeled as a thread with synchronized access to a shared state. The simulator emulates battery dynamics, spatial distribution, task loads, and re-election cycles every 0.2 seconds for a total duration of 120 seconds. The testbed includes five robots with diverse energy profiles, workload conditions, and spatial locations.

A. Robot Initialization Parameters

To simulate realistic heterogeneity in team behavior, the following initial parameters were assigned:

TABLE I
Initial Parameters of Each Robot

Robot ID	Battery (%)	Drain Rate	Task Load	Position (x, y)
0	92	12	5	(0.0, 0.0)
1	93	15	3	(20.0, 20.0)
2	95	10	7	(0.1, 0.3)
3	95	30	2	(0.3, 0.1)
4	94	16	8	(0.2, 0.2)

These values were chosen to reflect operational diversity:

- Robot 3 has the highest drain rate ($D_3 = 30$) and lowest task load.
- Robot 4 is the most heavily loaded but centrally located.
- Robot 1 is a spatial outlier, positioned far from the rest of the team.

B. Overall Stability Comparison Across Algorithms

We first analyze the cumulative number of leader changes over time, a proxy for system stability. Frequent leader switches imply instability and energy inefficiency.

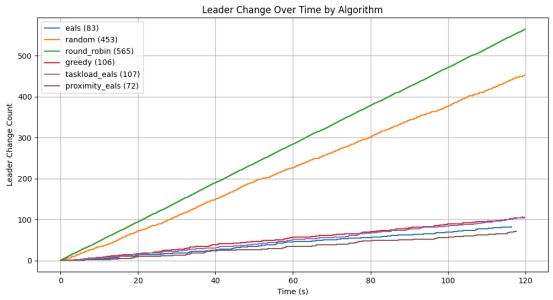

Fig. 1. Cumulative leader changes over time across algorithms. Lower counts indicate better stability.

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.

a) Observation.: Adaptive variants of EALS outperform all baselines in minimizing leader changes. Proximity-EALS offers the highest stability, followed closely by EALS and Taskload-EALS. In contrast, Round Robin and Random result in excessive churn due to their non-contextual nature.

C. Effect of EALS Metric on Leader Fairness

EALS uses the quadratic metric:

$$M_i = (100 - B_i) \cdot D_i^2$$

which penalizes high-drain robots more strongly. This effectively biases against robots with unsustainable energy usage patterns.

a) Analysis.: Figure 2 shows that Random uniformly selects all robots. In contrast, Figure 3 demonstrates that EALS systematically avoids Robot 3, which has the highest drain rate ($D_3 = 30$). Robot 2, with the lowest drain, is most frequently selected—confirming that the updated quadratic cost function enforces long-term energy efficiency.

D. Contextual Variants: Task and Proximity Awareness

We now assess how task load and spatial awareness influence leader choice.

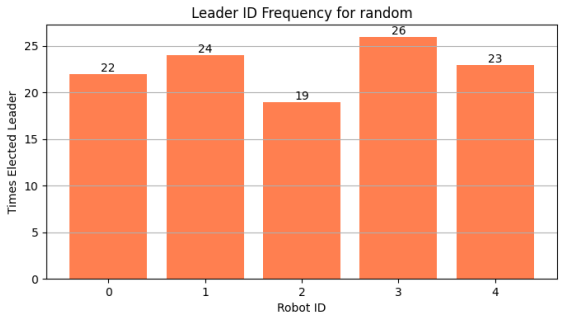

Fig. 2. Random selection: leader distribution is uniform across robots.

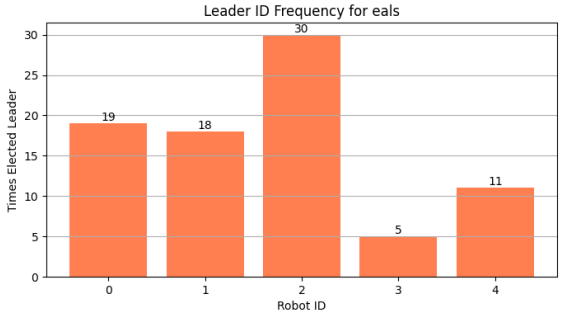

Fig. 3. EALS: high-drain Robot 3 is rarely selected.

a) Task-Aware Behavior.: Robots 2 and 4 had the highest task loads (7 and 8). Taskload-EALS successfully avoids selecting these robots, instead preferring Robot 1 and Robot 3 (task load 3 and 2), as seen in Figure 4. This behavior aligns with the system's goal to distribute leadership based on workload balance.

b) Proximity-Aware Behavior.: Proximity-EALS penalizes spatial outliers. Robot 1, isolated at (20, 20), is rarely elected. Robots 2 and 4, positioned centrally, dominate the leadership decisions—minimizing communication latency and improving coordination.

E. Greedy Selection and the Re-Election Feedback Loop

Greedy selection always picks the robot with the highest current battery percentage. While straightforward, this can cause problematic re-election patterns for high-drain robots.

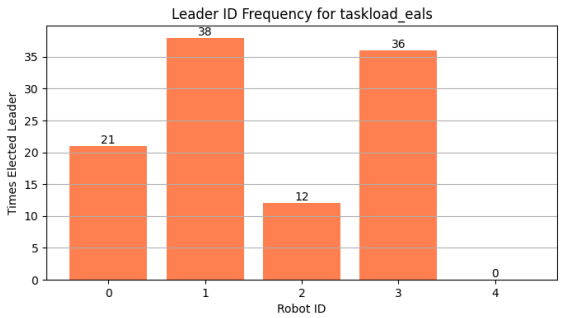

Fig. 4. Taskload-EALS: avoids highly loaded robots such as Robot 4.

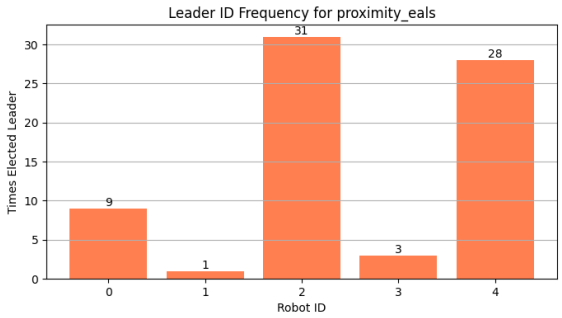

Fig. 5. Proximity-EALS: avoids distant Robot 1 located at (20, 20).

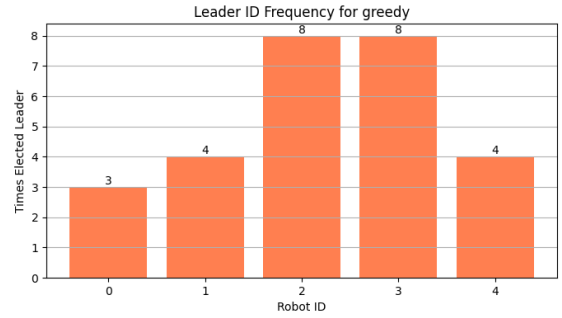

Fig. 6. Greedy: Robot 3 re-elected frequently due to recharge spikes.

a) Analysis.: Robot 3, with the highest energy drain, is still selected 8 times (Figure 6)—equal to Robot 2. This happens because Robot 3 recharges fully after every drain cycle, misleading the greedy strategy into repeatedly electing it.

This creates a feedback loop:

1) Robot 3 is elected due to full battery,
2) It drains quickly and dies,
3) Upon recharge, it becomes eligible again,
4) Greedy re-elects it, restarting the cycle.

Such cycles highlight the unsustainability of purely myopic selection schemes.

F. Robustness Under Uncertainty

To evaluate robustness, we introduce environmental uncertainty in the form of (i) Gaussian noise added to robot telemetry (battery levels and task load), (ii) intermittent communication dropouts, and (iii) transient robot failures. These perturbations simulate real-world inconsistencies such as sensor drift, lossy networks, and hardware resets.

- **Sensor Noise:** Battery readings are perturbed with $\mathcal{N}(0, 3\%)$, and task load with $\mathcal{N}(0, 1.0)$.
- **Dropout:** Each robot skips state updates with 10% probability per timestep.
- **Failure-Recovery:** Robots fail randomly with a 5% chance per timestep and recover with a 30% probability per tick.

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 = 120)

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

Findings: EALS variants remain effective under uncertainty, with PROXIMITY-EALS demonstrating the most resilience. While TASKLOAD-EALS suffers due to noisy or missing task data, it still maintains competitive performance. Algorithms without contextual filtering (e.g., RANDOM, GREEDY) are highly unstable under these perturbations.

G. Summary of Findings

- EALS consistently reduces leader churn and promotes energy-aware leadership by penalizing high-drain robots quadratically. Its performance remains stable even under environmental uncertainty.
- TASKLOAD-EALS improves contextual decision-making by accounting for workload imbalance, but exhibits increased sensitivity to noisy or missing task data, resulting in higher churn under uncertainty.
- PROXIMITY-EALS offers the best robustness across all settings by prioritizing spatially central robots and avoiding communication outliers. It achieves the lowest churn both in clean and noisy environments.
- GREEDY selection re-elects high battery robots (e.g., Robot 3) post recharge, creating a cyclic loop that leads to instability, particularly when sensor noise is injected.
- RANDOM and ROUND ROBIN strategies are highly volatile. They exhibit poor scalability and robustness, with Random flailing under sensor/communication dropout and Round Robin incurring maximum churn by design.
- Overall, the proposed EALS framework—augmented by contextual metrics—delivers scalable, interpretable, and fault-tolerant leader election strategies for distributed multi-robot systems.

VII. DISCUSSION

The EALS framework bridges the gap between energy-aware optimization and real-time adaptability in multi-robot systems. Its modular design, encompassing formal verification, simulation, and distributed deployment, enables comprehensive validation across theoretical and applied dimensions. This section discusses trade-offs, robustness, observed behaviors, and deployment implications.

A. Metric Design and Trade-offs

EALS leverages a simple, interpretable metric to prioritize energy-efficient leadership. Its quadratic penalization of high-drain robots stabilizes leadership and extends system lifetime, particularly in homogeneous teams. However, real-world deployments often exhibit heterogeneity in task demand and spatial distribution. Adaptive variants—Taskload-EALS and Proximity-EALS—address this by aggregating additional context metrics.

While these adaptations improve selection quality and churn reduction, they increase computational cost. Proximity-EALS, for example, incurs $O(n^2)$ runtime due to pairwise distance evaluations. Though tractable for moderate team sizes, scalability remains a concern for large swarms, motivating future integration of approximate centrality estimators.

B. Robustness Under Uncertainty

Leader election in the wild must tolerate noise in sensing, communication, and robot dynamics. The proposed framework maintains robustness through metric normalization, rank-based aggregation, and Pareto filtering. Proximity-EALS reduces re-election instability by selecting spatially central agents, which often have more stable communication and task coordination roles. Similarly, Taskload-EALS avoids overburdened robots, minimizing failure risks.

These variants demonstrate strong performance even when faced with fluctuating drain rates and intermittent failures. Our uncertainty experiments confirm that structured, interpretable metrics outperform randomized or greedy policies in maintaining leadership stability under degraded telemetry.

C. Behavioral Insights from Simulation

The Python-based simulator offered a controlled environment to examine algorithm dynamics. It revealed several emergent behaviors:

- **Greedy instability:** High-drain robots are cyclically re-elected post recharge, creating inefficient leadership loops.
- **EALS stability:** The base EALS metric curbs such loops through penalization, yielding significantly fewer leader transitions.
- **Context-aware avoidance:** Adaptive variants selectively exclude robots with high task load or spatial isolation.

These behaviors validate the premise that intelligent metric design translates to desirable operational characteristics: lower churn, energy balancing, and contextual fairness.

D. Real-Time Validation via ROS2

To confirm practical deployability, EALS was deployed in a decentralized ROS2-based architecture. Each robot publishes its telemetry data, and all nodes independently subscribe and execute the leader election logic.

This distributed design enables:

- **Scalability**: No central bottleneck; each robot computes the leader independently.
- **Fault Tolerance**: Leader unavailability triggers local re-election logic.
- **Modularity**: The system accommodates dynamic join/leave operations without reconfiguration.

The consistent leader selection arises from the deterministic nature of the EALS algorithm. Decentralized messaging ensures that all nodes converge to the same decision given synchronized state broadcasts. This confirms that EALS is suitable for real-world, distributed, and resilient multi-robot deployments.

E. Limitations and Assumptions

Despite promising results, the current framework has limitations:

- **Scalability Bottlenecks:** Proximity-EALS and hybrid variants may face performance issues in large swarms due to computational overhead.
- **Static Weighting:** All metric weights are pre-defined. Dynamic adjustment could improve responsiveness to mission shifts.
- **Trust Assumptions:** The system presumes truthful metric reporting. It does not yet defend against misbehaving or compromised agents.
- **Telemetry Dependence:** Advanced variants rely on high-fidelity logs for metrics such as failure history or link quality.

Addressing these limitations is critical for deploying EALS in adversarial or resource-constrained environments.

F. Toward Learning-Augmented EALS

While EALS is deterministic and explainable, it is well-positioned for hybridization with learning techniques:

- Reinforcement learning agents can optimize metric weights based on feedback or task success rates.
- Multi-armed bandits may fine-tune leader confidence levels based on prior outcomes.
- Supervised predictors can estimate failure probability or communication latency from sensor trends.

Such integration would retain EALS's safety guarantees while enabling mission-adaptive behavior in dynamic contexts.

G. System-Level Impact and Application Domains

The full-stack implementation of EALS—from Z3 validation to distributed ROS2 nodes—illustrates a reproducible and robust methodology for leader election. It is applicable to:

- **Energy-sensitive teams** such as UAV swarms and exploration rovers,
- **Unstable networks** like underwater or maritime robotic fleets,
- **Context-driven coordination** in heterogeneous multi-agent systems.

Its flexibility and interpretability position it as a strong candidate for next-generation coordination strategies across mission-critical applications.

H. Performance Under Sensor and Communication Noise

Our ablation study under noisy conditions demonstrates that context-aware variants—particularly Proximity-EALS—retain robust performance. Leader transitions remain low, and selection bias remains consistent, confirming EALS's ability to operate under real-world uncertainties such as latency, packet drops, or sensor faults.

VIII. Conclusion and Future Work

This work presented the **Energy-Aware Leadership Selection (EALS)** framework, a principled, interpretable, and modular approach to leader election in distributed multi-robot systems. By framing leader selection as a multi-objective decision problem, EALS integrates energy efficiency with additional dimensions such as task load, spatial proximity, communication reliability, and historical stability.

The framework was instantiated through four adaptive variants and evaluated across three tiers: formal verification using Z3, behavioral benchmarking via a multithreaded Python simulator, and real-time deployment in a decentralized ROS2-based system. Across all layers, EALS consistently outperformed traditional and heuristic baselines, reducing leader churn and maintaining robustness under dynamic operational constraints.

The modular structure and deterministic logic of EALS make it particularly well-suited for real-world, safety-critical deployments. Adaptive variants such as Proximity-EALS and Reliability-EALS further demonstrate the benefit of context-aware decision-making in improving system stability and longevity.

Future Work

Several promising directions can extend the capabilities of the EALS framework:

- **Scalable Proximity Computation:** Reducing the $O(n^2)$ cost in Proximity-EALS using approximate methods such as spatial hashing, landmark embedding, or clustering.
- **Context-Aware Weighting:** Replacing static rank aggregation with adaptive weight tuning driven by reinforcement learning or mission-level feedback.
- **Trust and Security:** Incorporating robustness to false metric reporting through trust modeling, anomaly detection, or Byzantine-resilient consensus.
- **Hardware Deployment:** Deploying EALS on physical robot platforms (e.g., TurtleBots, UAV swarms) to evaluate performance under real-world noise, latency, and control loop constraints.

In conclusion, EALS advances the state of multi-robot coordination by providing a deployable, explainable, and energy-aware leader selection framework—bridging theoretical soundness and practical resilience across diverse operational domains.

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.