Survey on Swarm Robotics and Multi-Rover Control Methods

Aniketh Bhargav

1 Introduction

Swarm robotics employs decentralized coordination inspired by natural collective behaviors, such as ant colonies or bird flocks, to address tasks like area coverage, essential for applications including exploration, surveillance, mapping, and foraging. Multi-rover systems, characterized by complex dynamics and non-holonomic constraints, require advanced control strategies like Nonlinear Model Predictive Control (NMPC) to manage torque saturation and cooperative manipulation. Single-robot coverage algorithms, such as the Cell Permeability-based Coverage (CPC) algorithm, provide exact coverage with minimal overlap in unknown environments. Recent advances include the Additively Weighted Guaranteed Voronoi (AWGV) diagram for multi-agent coverage under localization uncertainty and Optimal Transport-based Coverage Control (OTCC) for generalizing Voronoi-based methods. This survey integrates theoretical concepts from a 2015–2022 review of swarm robotics for area coverage, a study on NMPC for multi-rover systems, the CPC algorithm, AWGV for imprecise localization, and OTCC for swarm systems, providing detailed explanations of each method's theory, mathematical formulation, and relevance.

2 Theoretical Foundations of Swarm Robotics Area Coverage Methods

Swarm robotics algorithms for area coverage are categorized into metaheuristic (bio-inspired) and classical (deterministic) approaches, each rooted in distinct theoretical principles. References to specific papers and authors are provided for each algorithm.

2.1 Metaheuristic Algorithms

Metaheuristic algorithms leverage swarm intelligence, drawing from natural systems to achieve decentralized, scalable, and robust coordination.

2.1.1 Ant Colony Optimization (ACO)

Theory: ACO[1] is inspired by the foraging behavior of ants, where individuals deposit pheromones to mark paths to food sources, creating a positive feedback loop that guides the colony toward optimal routes. In swarm robotics, artificial pheromones, implemented via robot-to-robot communication or environmental markers, guide exploration and coverage. The theory relies on stigmergy, where indirect coordination emerges from environmental modifications, enabling

decentralized decision-making. ACO balances exploration (searching new areas) and exploitation (following known paths) through pheromone evaporation and deposition, making it suitable for dynamic environments. The algorithm's robustness stems from its ability to adapt to environmental changes, such as obstacles or new targets, by updating pheromone trails dynamically. This mimics the biological principle of self-organization, where simple local rules lead to complex global behavior, ensuring scalability and fault tolerance in robotic swarms.

Mathematical Framework:

- **Pheromone Update**: For a path between nodes i and j, the pheromone level τ_{ij} is updated as:
$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \sum_{\text{robot } k} \Delta\tau_{ij}^{k}, \tag{1}$$
where $\rho \in [0,1]$ is the evaporation rate to prevent convergence to suboptimal paths, and $\Delta\tau_{ij}^{k}$ is the pheromone deposited by robot k, typically proportional to path quality (e.g., inverse distance or coverage efficiency).

- **Transition Probability**: Robot k moves from node i to j with probability:
$$p_{ij}^{k} = \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{l\in\text{allowed}} \tau_{il}^{\alpha}\eta_{il}^{\beta}}, \tag{2}$$
where η_{ij} is a heuristic (e.g., $\eta_{ij} = 1/d_{ij}$, with d_{ij} as distance), and α, β control the influence of pheromones and heuristics, respectively.

Variants:

- **Cellular Automata Ant Memory Model (CAAM)**: Discretizes the environment into a grid, using cellular automata rules and repulsive (inverted) pheromones to discourage revisiting covered areas. Each robot maintains a tabu list:
$$\text{tabu}_i = \{c_1, c_2, \ldots, c_n\}, \tag{3}$$
where c_i are recently visited cells, inspired by Tabu Search to avoid local optima. This approach enhances coverage by promoting exploration of unvisited regions.

- **Ant Foraging with Adaptive Brownian Lévy Flight**: Dynamically switches between Brownian motion (short, random steps for local exploration) and Lévy flight (long jumps for wide-area search). The step length s in Lévy flight follows a power-law distribution:
$$p(s) \sim s^{-\mu}, \quad 1 < \mu \leq 3, \tag{4}$$
enabling efficient coverage in sparse or large environments.

- **Genetic Shared Tabu Inverted Ant Cellular Automata (GSTIACA)**: Integrates genetic algorithms for path optimization, inverted pheromones for robot repulsion, and shared memory for coordination. Robots repel each other to maximize coverage, using:
$$\text{fitness}_i = f(\text{coverage area}, \text{obstacle avoidance}), \tag{5}$$
where genetic algorithms optimize parameters like repulsion strength.

Strengths: Robust to environmental noise, effective for both local and global coverage, enhances other algorithms through hybridization. **Limitations**: Uncertain convergence time due to probabilistic nature, high computational cost for CAAM, and limited real-robot validation for GSTIACA.

2.1.2 Particle Swarm Optimization (PSO)

Theory: PSO[1] emulates the flocking behavior of birds, where individuals adjust their positions based on personal and collective experiences. Each robot (particle) navigates semi-autonomously, balancing individual exploration with swarm-wide exploitation. The theory is grounded in social optimization, where local interactions and shared knowledge drive the swarm toward optimal solutions. PSO is particularly effective for continuous optimization problems, requiring no gradient information, making it versatile for area coverage in unknown environments. The algorithm mimics the cognitive and social behaviors of birds, where each robot evaluates its own best position (cognitive component) and the swarm's best position (social component) to update its trajectory. This dual influence ensures a balance between exploring new areas and converging toward known optimal regions, making PSO adaptable to dynamic or uncertain environments.

Mathematical Framework:

- **Velocity and Position Update**: For robot i, velocity $\mathbf{v}_i$ and position $\mathbf{x}_i$ are updated as:

$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + c_1 r_1(\mathbf{p}_i - \mathbf{x}_i(t)) + c_2 r_2(\mathbf{p}_g - \mathbf{x}_i(t)), \tag{6}$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1), \tag{7}$$

  where w is the inertia weight controlling momentum, c_1, c_2 are cognitive and social coefficients, $r_1, r_2 \in [0,1]$ are random factors for stochasticity, $\mathbf{p}_i$ is the robot's best-known position, and $\mathbf{p}_g$ is the swarm's global best position.

Variants:

- **PSO with Inertia Weight**: Adjusts w dynamically (e.g., decreasing linearly over time) to balance exploration (large w) and exploitation (small w).
- **Robotic Darwinian PSO (RDPSO)**: Incorporates Probabilistic Finite State Machines and Depth First Search to reduce exploration time, using:

$$\text{state}_i = f(\text{current position}, \text{target proximity}), \tag{8}$$

  to guide robots toward uncovered areas.
- **Exploration-Enhanced RPSO (E2RPSO)**: Enhances multi-target search by adjusting inertia weight and diversity metrics to avoid local optima.
- **Adaptive Exploration RPSO (AERPSO)**: Uses adaptive inertia weight and evolutionary speed:

$$w(t) = w_{\text{max}} - \frac{(w_{\text{max}} - w_{\text{min}})t}{T}, \tag{9}$$

  where T is the total iteration count, improving obstacle avoidance and search efficiency.

Strengths: Simple implementation, fast computation, robust to local optima, scalable with swarm size. **Limitations**: Requires powerful robots for obstacle navigation, often needs external positioning systems (e.g., GPS).

2.1.3 Bacterial Foraging Optimization (BFO)

Theory: BFO[1] mimics the foraging behavior of E. coli bacteria, which move via chemotaxis (directed movement toward nutrients), reproduction, and elimination-dispersal. The theory emphasizes decentralized optimization, where each robot (bacterium) independently seeks optimal coverage areas based on local environmental cues. BFO's strength lies in its ability to escape local optima through random dispersal, making it suitable for dynamic or complex environments. The algorithm replicates bacterial strategies of tumbling (random direction changes) and swimming (directed movement), allowing robots to explore the environment efficiently. The reproduction and elimination-dispersal mechanisms ensure that high-performing robots (those covering more area) persist, while underperforming ones are relocated, promoting diversity and preventing stagnation.

Mathematical Framework:

- **Position Update**: For robot i, position θ_i updates via chemotaxis:

$$\theta_i(t+1) = \theta_i(t) + C_i \frac{\Delta_i}{\sqrt{\Delta_i^T \Delta_i}}, \tag{10}$$

  where C_i is the step size, and Δ_i is a random direction vector. The fitness function evaluates coverage:

$$J_i = f(\theta_i, \text{covered area}, \text{obstacles}). \tag{11}$$

- **Reproduction**: Robots with high fitness replicate, while low-fitness robots are eliminated.

- **Elimination-Dispersal**: Robots are randomly relocated with probability p_{ed} to explore new areas.

Variants:

- **Bacterial Chemotaxis (B.C.)**: Uses Voronoi partitioning to divide the environment:

$$V_i = \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{p}_i\| \leq \|\mathbf{x} - \mathbf{p}_j\|, \forall j \neq i\}, \tag{12}$$

  where $\mathbf{p}_i$ is robot i's position, enabling decentralized target search.

- **Bacterial Chemotaxis-Inspired Coordination Strategy (BCCS)**: Applies chaotic preprocessing for initial positions to enhance exploration:

$$\theta_i(0) = \text{chaos}(\text{seed}), \tag{13}$$

  reducing convergence time.

Strengths: Robust to local optima, effective for target search and coverage. **Limitations**: Time-consuming initialization, limited real-robot validation.

2.1.4 Bee Algorithms

Theory: Bee algorithms[1] are inspired by honey bee foraging and mating behaviors, where a "queen" robot guides others toward optimal solutions. The theory leverages labor division and probabilistic selection, with employed bees exploiting known areas, onlookers exploring

promising regions, and scouts searching randomly. This approach is effective for dynamic environments requiring adaptive coverage strategies. The algorithm models the waggle dance of bees, where robots share information about high-quality coverage areas, enabling efficient allocation of exploration efforts. The division of roles ensures a balance between exploiting known high-value regions and discovering new areas, making the algorithm adaptable to changing environmental conditions.

Mathematical Framework:

- **Honey Bee Optimization (MBO)**: Robots evaluate food sources (coverage areas) based on fitness:

$$P(\text{select}_i) = \frac{f_i}{\sum_j f_j}, \tag{14}$$

  where f_i is the fitness of source i (e.g., uncovered area size). Employed bees search locally:

$$\mathbf{x}_i^{\text{new}} = \mathbf{x}_i + \phi(\mathbf{x}_i - \mathbf{x}_k), \tag{15}$$

  where $\phi \in [-1, 1]$ is a random factor, and $\mathbf{x}_k$ is a neighbor's position.

- **Fast Marriage in Honey Bee Optimization (FMHBO)**: The queen selects drones based on fitness, with probabilistic acceptance:

$$P(\text{accept}_d) = e^{-\Delta f/T}, \tag{16}$$

  where Δf is the fitness difference, and T is a temperature parameter.

Strengths: Scalable, low computational cost, adaptable to dynamic scenarios. **Limitations**: Slow sequential processing, requires new fitness function designs.

2.1.5 Bio-Inspired Neural Network

Theory: Bio-Inspired Neural Network[1] is an approach that uses neural dynamics to guide robots, treating other robots and obstacles as dynamic elements in a neural landscape. The theory is based on computational neuroscience, where each robot's path is generated by a neural network that processes local sensory inputs without global planning. This method is fault-tolerant and suitable for real-time coverage in dynamic environments. The neural network models the environment as a topographic map, where neurons represent spatial locations, and their activation levels guide robot movement. This approach enables robots to respond rapidly to environmental changes, such as moving obstacles, by updating the neural landscape in real time, ensuring robust navigation with minimal computational overhead.

Mathematical Framework:

- **Path Generation**: Robot i's position evolves as:

$$\dot{\mathbf{x}}_i = f(\text{neural landscape}, \text{previous positions}), \tag{17}$$

  where f is a neural network function integrating inputs from neighboring robots and obstacles:

$$f = \sum_j w_{ij}\sigma(\mathbf{x}_j - \mathbf{x}_i), \tag{18}$$

  with w_{ij} as weights and σ as an activation function.

Strengths: Reduces completion time, robust to faults. **Limitations**: Limited accuracy due to simplified modeling.

2.1.6 Cohort Intelligence (CI)

Theory: CI[1] models robot interactions as a cohort striving for a common goal, inspired by social learning. Each robot improves its behavior by observing others, using probabilistic selection to explore diverse solutions. The theory emphasizes emergent intelligence through local interactions, making it suitable for search and rescue tasks. CI simulates a learning process where robots adopt behaviors from high-performing peers, fostering competition and cooperation within the cohort. This iterative learning mechanism allows the swarm to adapt to complex environments, such as those with non-convex obstacles, by continuously refining individual and collective strategies.

Mathematical Framework:

- **Selection Probability**: Robot i is selected for behavior adoption:

$$P(\text{select}_i) = \frac{f_i}{\sum_j f_j}, \tag{19}$$

  where f_i is fitness based on coverage or task completion. Perturbation avoids local optima:

$$\mathbf{x}_i^{\text{new}} = \mathbf{x}_i + \epsilon \mathbf{r}, \tag{20}$$

  with ϵ as perturbation magnitude and $\mathbf{r}$ as a random vector.

Strengths: Avoids non-convex obstacles, scalable. **Limitations**: May require additional techniques for complex scenarios.

2.2 Classical Algorithms

Classical algorithms use deterministic models to achieve structured, predictable coverage.

2.2.1 Dynamic Voronoi-Based Algorithm

Theory: Dynamic Voronoi-Based Algorithm[1] method divides the environment into Voronoi cells, where each robot covers the region closest to itself. The theory is rooted in computational geometry, ensuring optimal partitioning for distributed coverage. Dynamic updates adjust cells as robots move, making it suitable for dynamic environments. The algorithm leverages the property of Voronoi diagrams that each cell contains points closer to a given robot than to any other, ensuring efficient division of labor. Robots continuously recompute their Voronoi cells based on current positions, moving toward the centroids of their cells to maximize coverage, which is particularly effective in environments requiring uniform distribution of sensors or resources.

Mathematical Framework:

- **Voronoi Partitioning**:

$$V_i = \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{p}_i\| \leq \|\mathbf{x} - \mathbf{p}_j\|, \forall j \neq i\}, \tag{21}$$

  where $\mathbf{p}_i$ is robot i's position. Robots move toward the centroid of their cell:

$$\mathbf{c}_i = \frac{1}{|V_i|} \int_{V_i} \mathbf{x} \, d\mathbf{x}. \tag{22}$$

Strengths: Energy-efficient, escapes local optima. **Limitations**: Computationally intensive, untested on physical robots.

2.2.2 Decentralised Space-Based Potential Field (D-SBPF)

Theory: D-SBPF[1] uses virtual potential fields to guide robots, with attractive forces toward unexplored areas and repulsive forces from covered regions or obstacles. The theory is based on physics-inspired navigation, enabling decentralized exploration without global maps. The potential field approach treats the environment as a force field, where robots are driven by gradients that balance exploration and avoidance behaviors. This method is particularly suited for real-time applications, as it requires only local information, allowing robots to react swiftly to environmental changes, such as newly discovered obstacles or teammates' positions.

Mathematical Framework:

- **Force Calculation**:

$$\mathbf{F}_i = \sum_{\text{cells}} k_a \mathbf{u}_a - k_r \mathbf{u}_r, \tag{23}$$

  where k_a, k_r are attractive/repulsive coefficients, and $\mathbf{u}_a, \mathbf{u}_r$ are unit vectors toward unexplored/covered cells.

Strengths: Simple, decentralized, supports dynamic team changes. **Limitations**: Reduced efficiency with sparse robot teams.

2.2.3 Force Vector Algorithm (FVA)

Theory: FVA[1] treats robots as particles subject to virtual forces, balancing repulsion from neighbors, attraction to uncovered areas, and obstacle avoidance. The theory draws from particle dynamics, enabling simple, decentralized coverage with minimal computational requirements. By modeling robots as particles in a force field, FVA ensures that robots spread out to cover the environment while avoiding collisions and obstacles. This approach is computationally lightweight, making it suitable for resource-constrained robots, and its simplicity allows for easy implementation in real-world scenarios with minimal sensor requirements.

Mathematical Framework:

- **Resultant Force**:

$$\mathbf{F}_i = \mathbf{F}_{\text{repel}} + \mathbf{F}_{\text{attract}} - \mathbf{F}_{\text{obstacle}}, \tag{24}$$

  where $\mathbf{F}_{\text{repel}} = \sum_{j \neq i} k_r / \|\mathbf{x}_i - \mathbf{x}_j\|^2$, and $\mathbf{F}_{\text{attract}}$ targets uncovered regions.

Strengths: Simple, effective for minimal hardware. **Limitations**: Less reliable than metaheuristic methods.

2.2.4 Cooperative Distributed Asynchronous (CORDA)

Theory: CORDA[1] models robot behavior as sequential cycles of observation, computation, and movement, mimicking real-world asynchronous interactions. The theory emphasizes fault tolerance and scalability, suitable for environments with obstacles. CORDA models robots as independent agents that execute their actions through three sequential stages: Observe (sense the environment and gather information), Compute (decide what to do based on local knowledge), and Move (execute the action, e.g., move, explore, avoid). Each robot performs these stages asynchronously, without a global clock or waiting for others, which enhances robustness in heterogeneous or failure-prone swarms. This asynchronous nature allows CORDA to handle communication delays and partial sensor failures, making it practical for real-world deployments.

Mathematical Framework: Robots operate in cycles:

$$\text{state}_i(t+1) = f(\text{observe}, \text{compute}, \text{move}), \tag{25}$$

where f updates based on local sensor data and neighbor positions.

Strengths: Fault-tolerant, reduces system cost. **Limitations**: Affected by limited visibility.

2.2.5 Deployment Entropy with Potential Fields

Theory: This method[1] maximizes entropy to achieve uniform robot distribution, using potential fields for navigation. The theory is based on information theory, ensuring optimal sensor coverage in persistent tasks. By maximizing entropy, the algorithm promotes a uniform distribution of robots across the environment, which is critical for applications like surveillance or environmental monitoring. The potential fields guide robots to areas with lower robot density, ensuring balanced coverage and preventing clustering, while the entropy metric provides a quantitative measure of distribution quality.

Mathematical Framework:

- **Entropy**:

$$H = -\sum p_i \log p_i, \tag{26}$$

  where p_i is the probability of a robot in region i. Robots move to maximize H using potential fields.

Strengths: Scalable, uniform coverage. **Limitations**: Lacks persistence data.

2.2.6 Self-Organising Area Coverage with Gradient and Grouping (GGC)

Theory: GGC[1] enables self-organization through gradient-based navigation and group coordination, allowing parallel coverage. The theory emphasizes emergent behavior with minimal robot capabilities, ideal for micro-scale applications. Robots form groups based on local gradients, which represent uncovered areas, and coordinate within groups to cover the environment efficiently. This self-organizing behavior reduces the need for centralized control, making GGC suitable for large-scale swarms or resource-constrained robots, such as those in micro-robotics or nanotechnology applications.

Mathematical Framework: Robots follow gradients:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) - \nabla U(\mathbf{x}_i), \tag{27}$$

where U is a potential function representing uncovered areas.

Strengths: Fast, robust, supports parallel coverage. **Limitations**: Limited by team size and coverage range.

2.2.7 Frontier-Led Swarming

Theory: Frontier-Led Swarming[1] method guides robots toward unexplored frontiers while maintaining formation, inspired by exploration strategies in unknown environments. The theory ensures connectivity for short-range communication, effective in cluttered spaces. Robots prioritize moving toward boundaries between explored and unexplored regions, maintaining a cohesive formation to ensure communication and coordination. This approach is particularly

effective in complex environments with obstacles, as it systematically expands the explored area while preserving swarm connectivity, critical for applications like search and rescue.

Mathematical Framework: Robots prioritize frontier cells:

$$\text{priority}_c = f(\text{distance}, \text{unexplored status}), \tag{28}$$

moving to maximize coverage while adhering to formation constraints.

Strengths: High performance in cluttered environments. **Limitations**: Cannot track dynamic obstacles.

2.2.8 Cell Permeability-based Coverage (CPC) Algorithm

Theory: The CPC algorithm[2] addresses single-robot area coverage in unknown environments, achieving exact coverage with reduced overlap compared to existing approximate cellular decomposition-based algorithms like Spanning Tree-based Coverage (STC) and Boustrophedon Decomposition-based Coverage (BDC). The theory is grounded in graph-based path planning, introducing the concept of "minor nodes" associated with major cell boundaries to construct a spanning tree that ensures coverage of both free and partially occupied cells. Unlike STC, which skips partially occupied cells, CPC uses a permeability-based adjacency graph to enable robots to navigate through accessible boundaries, ensuring completeness. The algorithm employs depth-first search (DFS) for spanning tree construction and wall-following for obstacle avoidance, balancing computational simplicity with exact coverage requirements. It is particularly effective for non-coverage conducive regions, where exact coverage inherently introduces some overlap near obstacles. The permeability concept allows the robot to assess boundary accessibility dynamically, adapting to complex environments with partial obstructions, making it robust for real-world applications like cleaning or mapping.

CPC Algorithm

Mathematical Framework:

- **Area Decomposition**: The region $Q \subset \mathbb{R}^2$ is enclosed in a rectangle QR of size $2mD \times 2nD$, divided into $m \times n$ major cells of size $2D \times 2D$. Each major cell is subdivided into four subcells of size $D \times D$, where D is the robot's coverage tool size. Subcells are classified as:

$$\text{State}(s) \in \{\text{free (white), occupied (black), partially occupied}\}. \tag{29}$$

- **Node Definitions**:
  - Major nodes at the center of major cells.
  - Subnodes at the center of subcells.
  - Minor nodes (virtual) at the boundaries of major cells (north, east, west, south).
- **Permeability**: A boundary ACN between major cells C and N is permeable if its extension $A\tilde{C}N$ (lengthened by D at both ends) contains a free segment of length D:

$$\text{Permeable}(ACN) \iff \exists\, \text{segment} \in A\tilde{C}N,\ \text{length} \geq D,\ \text{free}. \tag{30}$$

Algorithm 1 Cell Permeability-based Coverage (CPC) Algorithm

Input: Starting cell S
Recursive Function: CPC(P, C), where P is the previous cell and C is the current cell
Initialization: $P \leftarrow$ null, $C \leftarrow S$
function CPC(P, C)
    Mark the current minor node lC ($l \in \{n, e, w, s\}$) and all other minor nodes in its cluster as "old"
    while lC has a prospective minor node **do**
        Scan neighboring major cells in anticlockwise direction for a prospective minor node
        Construct a spanning tree edge from the current minor node to the first available prospective minor node (say, iNj)
        Mark iNj and all minor nodes in its cluster as "old"
        Create a path through corresponding subnodes on the right side of the spanning tree edge
        Move the robot along this path to cell Nj, avoiding obstacles using a wall-following algorithm
        $P \leftarrow C$, $C \leftarrow Nj$
        CPC(P, C)
    if end of the spanning tree branch is reached **then**
        Circumnavigate along the constructed spanning tree to reach a cell (say, N) via subnodes
        Move the robot along this path to cell N, avoiding obstacles if any
        $P \leftarrow C$, $C \leftarrow N$
        CPC(P, C)
    if $P = S$ **then**
        CPC(P, C)
    return End of CPC(P, C)

- **Adjacency Graph**: Minor nodes are connected if their corresponding boundary is permeable (inter-cellular) or if they are within the same major cell and connected (intra-cellular). Clustered minor nodes form maximal connected subsets:

$$\text{Cluster}_k = \{\text{minor nodes } m_i \mid m_i \text{ connected in major cell } C\}. \tag{31}$$

- **Spanning Tree Construction**: Using DFS, the algorithm constructs a spanning tree over clustered minor nodes, marking nodes as "old" once visited:

$$\text{Edge}(lC, iNj) \text{ added if } iNj \text{ is prospective (new, connected).} \tag{32}$$

- **Path Generation**: The robot follows a graph-level path through subnodes on the right side of spanning tree edges, using wall-following for obstacle avoidance:

$$\text{Path} = \{\text{subnode}_1, \text{subnode}_2, \dots\}, \text{ right of Edge}(lC, iNj). \tag{33}$$

Key Properties:

- **P1**: The adjacency graph over free minor nodes equals that over clustered minor nodes.
- **P5, P6**: Free or effectively free major cells are covered non-repetitively, with subnodes visited exactly once.
- **P7**: Partially occupied cells with permeable boundaries are covered, with subnode repetition proportional to clustered nodes (up to four), but actual overlap is minimized via wall-following.
- **P8**: Blocked major cells (all minor nodes impermeable) are not covered.

Theorems:

- **Theorem 1**: CPC achieves complete, non-overlapping coverage for coverage conducive regions (identical to STC).
- **Theorem 2**: With suitable obstacle avoidance, CPC covers all free and partially occupied subcells.
- **Theorem 3**: The graph-level path passes through every subnode of free/partially free major cells at least once.
- **Theorem 4**: Subnodes of free/effectively free major cells appear exactly once in the graph-level path.

Strengths: Achieves exact coverage, minimizes overlap compared to STC and BDC, robust to minor sensory errors. **Limitations**: Challenges with minor node multiplication, requires accurate localization, computational complexity for complex environments.

2.2.9 Additively Weighted Guaranteed Voronoi (AWGV) Diagram

Theory: The AWGV[3] algorithm addresses static area coverage for a network of mobile, sensor-equipped agents operating with imprecise localization and heterogeneous sensing capabilities. It introduces a novel partitioning scheme, the Additively Weighted Guaranteed Voronoi (AWGV) diagram, which generalizes the Guaranteed Voronoi (GV) diagram by incorporating additively weighted distances based on agents' guaranteed sensing radii to account for positioning uncertainty. Each agent is assigned an AWGV cell, defining its area of responsibility, ensuring points are closer to the agent's position (within its uncertainty disk) than to others, adjusted by sensing weights. A distributed gradient-ascent control law maximizes coverage, enhanced with collision avoidance and region confinement mechanisms. Unlike traditional Voronoi tessellations, AWGV does not fully partition the region, leaving a neutral area, but guarantees coverage within R-limited AWGV cells. The approach is validated through simulations and real-world experiments using differential drive robots with ultra-wideband (UWB) positioning systems. The AWGV framework accounts for practical challenges, such as sensor noise and communication limitations, by ensuring that coverage is guaranteed within the bounds of localization uncertainty, making it highly applicable to real-world scenarios like surveillance or environmental monitoring.

Mathematical Framework:

- **Agent Dynamics**: The position q_i of agent i evolves as:

$$\dot{q}_i = u_i, \tag{34}$$

  where u_i is the control input.

- **Positioning Uncertainty**: Agent i's true position lies within an uncertainty disk:

$$C_{u_i}(q_i, r_i) = \{q \in O \mid \|q - q_i\| \leq r_i\}, \tag{35}$$

  where r_i is the known upper bound of positioning uncertainty, and O is a compact convex planar region.

- **Sensing Region**: Agent i is equipped with an omnidirectional sensor with a circular sensing footprint:

$$C_{s_i}(q_i, R_i) = \{q \in O \mid \|q - q_i\| \leq R_i\}, \tag{36}$$

  where R_i is the sensing radius.

- **Guaranteed Sensed Region (GSR)**: The region guaranteed to be sensed for all possible positions within C_{u_i} is:

$$C_{gs_i}(q_i, R_{g_i}) = \{q \in O \mid \|q - q_i\| \leq R_{g_i}\}, \quad R_{g_i} = R_i - r_i, \tag{37}$$

  where $R_{g_i} \geq 0$ if $R_i \geq r_i$; otherwise, C_{gs_i} is empty. If $r_i = 0$, $C_{gs_i} = C_{s_i}$.

- **AWGV Cell**: For agent i, the AWGV cell is defined as:

$$W_{g_i} = \{q \in O \mid \max(\|q - q_i\| - R_{g_i}) \leq \min(\|q - q_j\| - R_{g_j}), \forall j \neq i, q_i \in D_i, q_j \in D_j\}, \tag{38}$$

  where $D_i = C_{u_i}$ are uncertainty regions. The boundary ∂H_{ij} between agents i and j is a hyperbola:

$$\|q - q_j\| - \|q - q_i\| = r_i + r_j - R_{g_i} + R_{g_j}. \tag{39}$$

- **R-limited AWGV Cell**: Ensures guaranteed coverage:

$$W_{R_i} = W_{g_i} \cap C_{gs_i}, \tag{40}$$

where W_{R_i} are disjoint due to the disjointness of AWGV cells.

- **Coverage Objective**: Maximize the area coverage criterion:

$$H = \int_{\cup W_{R_i}} \phi(q)\, dq = \sum_i \int_{W_{R_i}} \phi(q)\, dq, \tag{41}$$

where $\phi(q)$ is an importance function for point $q \in O$.

- **Complete Control Law**: A gradient-ascent distributed control law ensures monotonic increase of H:

$$u_i = K_i \int_{\partial W_{R_i} \cap \partial C_{gs_i}} n_i \phi\, dq + K_i \sum_{j \in N_i} \left[\int_{\partial W_{R_i} \cap \partial H_{ij}} u_{i_i} n_i \phi\, dq + \int_{\partial W_{R_j} \cap \partial H_{ji}} u_{i_j} n_j \phi\, dq \right], \tag{42}$$

where N_i are AWGV neighbors, n_i is the outward unit normal, u_{i_i}, u_{i_j} are Jacobian matrices, and $K_i > 0$ is a gain. The time derivative satisfies:

$$\frac{dH}{dt} = \sum_i \|u_i\|^2 \geq 0. \tag{43}$$

- **Simplified Control Law**: A computationally efficient version:

$$u_i = K_i \int_{\partial W_{R_i} \cap \partial C_{gs_i}} n_i \phi\, dq, \tag{44}$$

using only the first term, with comparable performance but potentially slower convergence.

- **Control Enhancements**:
  - **Region Confinement**: Agents are constrained to a subset $O_{s_i} \subseteq O$, the Minkowski difference of O with a disk of radius r_i. If q_i is on ∂O_{s_i} and u_i points outward, u_i is projected onto ∂O_{s_i}.
  - **Collision Avoidance**: Agent i stops ($u_i = 0$) if the distance to agent j, $\|q_i - q_j\| \leq r_i + r_j + \epsilon$, and u_i moves toward j, where ϵ is small.

Strengths: Accounts for localization uncertainty and heterogeneous sensing, distributed control, experimentally validated with real robots, includes collision avoidance. **Limitations**: Does not fully tessellate the region, computationally intensive for the complete control law, simplified law may converge to suboptimal local optima.

2.2.10 Optimal Transport-based Coverage Control (OTCC)

Theory: The OTCC algorithm[4] generalizes Voronoi tessellation-based methods for multi-agent area coverage by leveraging optimal transport theory. It addresses dynamic coverage for swarm robots in a bounded convex domain, optimizing agent positions to minimize a cost functional that balances coverage quality (based on a density function representing area importance) and transport effort. Unlike traditional Voronoi methods, which assign regions based on Euclidean distances, OTCC uses optimal transport to partition the environment into cells that account for non-uniform importance distributions and heterogeneous agent capabilities. A distributed gradient-based control law drives agents to optimal configurations, ensuring adaptability to environmental changes and constraints. The approach is particularly effective for applications like environmental monitoring, surveillance, or resource allocation, where non-uniform coverage is critical. OTCC's use of optimal transport allows for a more flexible partitioning that aligns with the physical or operational constraints of the environment, such as varying sensor importance or resource availability, enhancing efficiency in complex scenarios.

Mathematical Framework:

- **Agent Dynamics**: The position p_i of agent i evolves as:

$$\dot{p}_i = u_i, \tag{45}$$

  where u_i is the control input.

- **Coverage Objective**: Minimize the cost functional:

$$J = \int_O \phi(q) d(q, P)\, dq, \tag{46}$$

  where $O \subset \mathbb{R}^2$ is the bounded convex domain, $\phi(q)$ is a positive density function representing the importance of point q, $P = \{p_1, \ldots, p_n\}$ is the set of agent positions, and $d(q, P)$ is the distance to the nearest agent, defined via optimal transport.

- **Optimal Transport Partitioning**: The region O is partitioned into cells W_i based on optimal transport:

$$W_i = \{q \in O \mid c(q, p_i) + \lambda_i \leq c(q, p_j) + \lambda_j, \forall j \neq i\}, \tag{47}$$

  where $c(q, p_i)$ is the transport cost (e.g., squared Euclidean distance, $c(q, p_i) = \|q - p_i\|^2$), and λ_i are Lagrange multipliers ensuring mass balance:

$$\int_{W_i} \phi(q)\, dq = \mu_i, \tag{48}$$

  where $\mu_i > 0$ is the capacity (or sensing weight) of agent i, satisfying $\sum_i \mu_i = \int_O \phi(q)\, dq$.

- **Control Law**: A distributed gradient-based control law minimizes J:

$$u_i = -k \nabla_{p_i} J = k \int_{W_i} \phi(q) \frac{p_i - q}{\|p_i - q\|}\, dq, \tag{49}$$

  where $k > 0$ is a control gain, and the integral is computed over the agent's assigned cell W_i. This drives each agent toward the centroid of its cell weighted by $\phi(q)$.

- **Centroidal Alignment**: The optimal configuration aligns each agent p_i with the centroid of its cell W_i:

$$p_i = \frac{\int_{W_i} q\phi(q)\, dq}{\int_{W_i} \phi(q)\, dq}. \tag{50}$$

- **Convergence**: Under suitable conditions (e.g., convex O, positive ϕ), the control law ensures convergence to a critical point of J, where cells W_i form a generalized centroidal tessellation.

Strengths: Flexible partitioning adapts to non-uniform importance distributions, supports heterogeneous agent capabilities, distributed control, theoretically robust for dynamic environments. **Limitations**: Computationally intensive for large swarms due to optimal transport calculations, requires accurate estimation of $\phi(q)$, limited experimental validation compared to simulation-based results.

3 Theoretical Foundations of Multi-Rover Control Methods

The study[5] presents Optimal Control Allocation (OCA) and Nonlinear Model Predictive Control (NMPC) for managing a system of two rovers with mecanum wheels and 7-DOF arms, addressing torque saturation and non-holonomic constraints.

3.1 Optimal Control Allocation (OCA)

Theory: OCA[5] is a two-stage control strategy designed for underdetermined systems, where the number of control inputs exceeds the degrees of freedom. The theory is rooted in optimization, aiming to allocate control efforts (e.g., wheel moments, joint torques) to minimize a cost function while satisfying dynamic constraints. Stage 1 generates reference trajectories using impedance control, which mimics a mass-spring-damper system to ensure smooth tracking. Stage 2 optimally distributes control inputs to minimize torque saturation and contact forces, leveraging quadratic programming for computational efficiency. OCA is particularly suited for multi-rover systems with redundant actuators, such as mecanum wheels, which provide omnidirectional mobility but introduce complexity in torque allocation. The impedance control in Stage 1 ensures compliance in cooperative tasks, such as load carrying, while the quadratic programming in Stage 2 ensures that control inputs remain within physical limits, enhancing system stability and performance.

Mathematical Framework:

- **Stage 1: Reference Trajectory Generation**: Uses impedance control:

$$\ddot{\mathbf{X}}_i = \mathbf{M}_i^{-1}\mathbf{B}(\dot{\mathbf{X}}_{d_i} - \dot{\mathbf{X}}_i) + \mathbf{M}_i^{-1}\mathbf{K}(\mathbf{X}_{d_i} - \mathbf{X}_i), \tag{51}$$

  where $\mathbf{M}_i, \mathbf{K}, \mathbf{B}$ are positive definite matrices representing mass, stiffness, and damping, $\mathbf{X}_i$ is the end-effector trajectory, and $\mathbf{X}_{d_i}$ is the desired trajectory. Inverse kinematics solve for joint rates via least-squares minimization.

- **Stage 2: Control Allocation**: Minimizes a quadratic cost:

$$C = \frac{1}{2}\tilde{\mathbf{S}}^T\mathbf{H}\tilde{\mathbf{S}} + \tilde{\lambda}^T\mathbf{E}, \tag{52}$$

where $\tilde{\mathbf{S}}$ includes contact forces/moments, wheel moments, and joint torques, $\mathbf{H}$ is a positive definite weighting matrix, $\tilde{\lambda}$ is the Lagrangian multiplier, and $\mathbf{E}$ represents constraints (e.g., system dynamics, load dynamics). The solution is:

$$\tilde{\mathbf{S}} = \mathbf{\Delta}^{-1}\mathbf{P}, \tag{53}$$

where $\mathbf{\Delta}$ and $\mathbf{P}$ are matrices derived from system parameters and dynamic terms.

Strengths: Efficient for underdetermined systems, computationally lightweight. **Limitations**: Less effective than NMPC in minimizing torques and forces.

3.2 Nonlinear Model Predictive Control (NMPC)

Theory: NMPC[5] is an advanced control strategy that optimizes system behavior over a finite prediction horizon, accounting for nonlinear dynamics and constraints. The theory is based on optimal control, where a cost function is minimized at each control interval to predict and adjust future states, inputs, and outputs. NMPC's strength lies in its ability to handle non-holonomic constraints and torque limits explicitly, making it ideal for complex multi-rover systems carrying a common load. The discretized formulation enhances computational feasibility while preserving nonlinear dynamics. NMPC's predictive nature allows it to anticipate future system states, such as potential torque saturation or load instability, and adjust control inputs proactively. This is particularly critical for cooperative manipulation tasks, where rovers must synchronize their movements to maintain load stability while navigating non-holonomic constraints inherent in wheeled systems. The algorithm's ability to incorporate constraints directly into the optimization problem ensures robust performance in dynamic environments.

Mathematical Framework:

- **Continuous Formulation**: Minimizes a cost function over horizon T_p:

$$C = \int_0^{T_p} \left[(\mathbf{y}(t) - \mathbf{y}_r(t))^T \mathbf{K} (\mathbf{y}(t) - \mathbf{y}_r(t)) + \mathbf{S}(t)^T \mathbf{H} \mathbf{S}(t)\right] dt, \tag{54}$$

subject to:

$$\dot{\mathbf{y}} = \mathbf{g}(\mathbf{y}) + \mathbf{L}\mathbf{S}, \quad \mathbf{z} = \mathbf{g}_z(\mathbf{y}) + \mathbf{H}\mathbf{S}, \quad \mathbf{y}(0) = \mathbf{y}(t_0), \tag{55}$$

where $\mathbf{y}$ is the state vector, $\mathbf{y}_r$ is the reference state, $\mathbf{S}$ includes control inputs, $\mathbf{K}, \mathbf{H}$ are weighting matrices, and $\mathbf{z}$ is the output vector.

- **Discretized Formulation**: The system is discretized for computational efficiency:

$$\mathbf{y}(k_t + 1) = \mathbf{A}_g(k_t)\mathbf{y}(k_t) + \mathbf{B}_g(k_t)\mathbf{S}(k_t), \tag{56}$$

$$\mathbf{z}(k_t) = \mathbf{C}_g(k_t)\mathbf{y}(k_t) + \mathbf{D}_g(k_t)\mathbf{S}(k_t), \tag{57}$$

minimizing:

$$C = \frac{1}{2} \sum_{i=1}^{N_p} \left[(\mathbf{y}(k_t + i) - \mathbf{y}_r(k_t + i))^T \mathbf{K} (\mathbf{y}(k_t + i) - \mathbf{y}_r(k_t + i)) + \mathbf{S}(k_t + i - 1)^T \mathbf{H} \mathbf{S}(k_t + i - 1)\right], \tag{58}$$

where N_p is the prediction horizon, and k_t is the sampling instant.

Strengths: Superior in reducing joint/wheel torques and contact forces, handles nonlinear constraints effectively. **Limitations**: Computationally intensive, requiring advanced hardware.

4 Discussion

Metaheuristic algorithms like ACO, PSO, and BFO offer scalability and robustness for area coverage, leveraging decentralized coordination to adapt to dynamic environments. Their bio-inspired nature enables efficient exploration and exploitation, though computational costs and real-world validation remain challenges. Classical algorithms, such as D-SBPF, FVA, CPC, AWGV, and OTCC, provide simplicity and determinism, ideal for structured environments. CPC excels in single-robot exact coverage with minimal overlap, particularly in non-coverage conducive regions. AWGV addresses multi-agent static coverage under localization uncertainty, enhancing robustness for heterogeneous agents through its novel partitioning scheme. OTCC generalizes Voronoi-based methods, offering flexible partitioning for non-uniform environments using optimal transport theory. For multi-rover systems, NMPC's predictive capabilities outperform OCA's static optimization, particularly in managing torque saturation and non-holonomic constraints, though at a higher computational cost. The choice of method depends on environmental complexity, robot capabilities, localization precision, and coverage objectives.

5 Conclusion

This survey elucidates the theoretical foundations of swarm robotics and multi-rover control methods, from bio-inspired algorithms like ACO and PSO to deterministic approaches like CPC, AWGV, and OTCC, and advanced control strategies like NMPC. Each method's mathematical framework supports specific applications: metaheuristics and OTCC excel in scalability and adaptability to non-uniform environments, AWGV ensures robustness to localization uncertainty, CPC provides exact coverage with minimal overlap, and NMPC offers precise control for complex dynamics. Future research should prioritize real-world validation, addressing challenges like minor node multiplication in CPC, non-tessellated regions in AWGV, computational complexity in OTCC, and integrating dynamic constraints to bridge theoretical advancements with practical deployment.

References

[1] D. K. Muhsen, A. T. Sadiq, and F. A. Raheem, "A Survey on Swarm Robotics for Area Coverage Problem," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 3, pp. 2345–2367, 2023.

[2] K. R. Guruprasad and T. D. Ranjitha, "CPC Algorithm: Exact Area Coverage by a Mobile Robot Using Approximate Cellular Decomposition," *Robotica*, vol. 38, no. 7, pp. 1141–1162, 2020.

[3] S. Papatheodorou, A. Tzes, K. Giannousakis, and Y. Stergiopoulos, "Distributed Area Coverage Control with Imprecise Robot Localization: Simulation and Experimental Studies," *International Journal of Advanced Robotic Systems*, vol. 16, no. 5, pp. 1–15, 2019.

[4] D. Inoue, Y. Ito, and H. Yoshida, "Optimal Transport-based Coverage Control for Swarm Robot Systems: Generalization of the Voronoi Tessellation-based Method," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1298–1313, 2021.

[5] S. Kalaycioglu and A. de Ruiter, "Nonlinear Model Predictive Control of Rover Robotics System," *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 8, pp. 1456–1470, 2021.