CPC Algorithm: Exact Area Coverage by a Mobile Robot Using Approximate Cellular Decomposition

This paper, "CPC Algorithm: Exact Area Coverage by a Mobile Robot Using Approximate Cellular Decomposition" by K. R. Guruprasad and T. D. Ranjitha, proposes a novel Coverage Path Planning (CPP) algorithm for mobile robots. The authors aim to achieve exact area coverage with lower overlap compared to existing algorithms, particularly those based on approximate cellular decomposition. The paper provides a formal analysis of the algorithm and presents results from experiments conducted on TurtleBot and Fire Bird V mobile robots.

Here's a detailed review of the paper:

1. Introduction and Problem Statement

Area coverage is a fundamental task for mobile robots across various real-world applications, including landmine detection, spray painting, floor cleaning, lawn mowing, manufacturing, and structural inspection. This problem is relevant for ground, aerial, and underwater robots. The core task of a robot with a given tool (e.g., sensor, actuator, camera field of view) is to visit every point within a bounded work-area, avoiding obstacles. A CPP algorithm should ensure complete coverage with minimal or no retraces (overlap).

Existing CPP algorithms are broadly classified based on:

- **Availability of a priori information**: Off-line (known environment) or on-line (sensor-based). This paper addresses an online CPP problem where complete a priori information may not be available.
- **Completeness guarantee**: Heuristic or provably complete.
- **Cellular decomposition**: Approximate or exact.

Several algorithms have been proposed:

- **Spanning Tree-based Coverage (STC) algorithms**: These are based on approximate cellular decomposition and have been extended for multi-robot scenarios.
- **Backtracking Spiral Algorithms (BSA) and extended-BSA**: These use spiral filling paths for covering decomposed regions.
- **Boustrophedon Decomposition-based Coverage (BDC) algorithm**: This uses an exact cellular decomposition scheme.
- Other approaches include neural networks, optimal coverage for known environments, and algorithms considering battery/fuel constraints.

2. Limitations of Existing Algorithms

The paper highlights several limitations of current approximate cellular decomposition-based CPP algorithms:

- **Incomplete Coverage**: Most approximate cellular decomposition-based algorithms, such as STC and BSA, only cover cells completely free of obstacles. This means partially occupied cells, even if accessible, are often left uncovered. For instance, the original STC algorithm might leave cells four times the robot's footprint size uncovered if partially occupied, even if they are largely free. The full-STC algorithm improves resolution to the robot's footprint size but still treats partially occupied cells as fully occupied and hence not covered.
- **Retracing and Overlap**: Graph search-based algorithms like BSA or BDC cannot completely avoid path retracing (due to dead ends). While STC aims to avoid retracing and overlaps by providing return paths, it sacrifices completeness by ignoring partially occupied cells. The full-STC algorithm results in retracing of coverage path segments. Algorithms like BSA and BDC provide exact coverage but don't guarantee non-overlapping coverage, even for free regions, and may require retracting to reach uncovered pockets. BDC can lead to higher coverage overlap when cell width is not an exact multiple of the robot's size, especially near obstacles.

The paper's goal is to address these issues by proposing an algorithm that achieves exact coverage of arbitrary, initially unknown regions (which may not be "coverage conducive") while reducing coverage overlap.

3. Proposed CPC Algorithm: Cell Permeability-based Coverage (CPC)

3.1. Problem Setting and Area Decomposition

The robot is equipped with a square coverage tool of side D. The region of interest Q is a topologically connected bounded subset of R^2 with obstacles O. Q is enclosed within a minimal rectangular area QR of size $2mD \times 2nD$. This enclosing rectangle is divided into $m \times n$ "major cells" of sides $2D$. Each major cell is further divided into four $D \times D$ sized "subcells". The space outside Q ($QR \setminus Q$) is considered a virtual obstacle.

- **Subcell States**: A subcell can be completely free (white), completely occupied by obstacles (black), or partially occupied by obstacles/virtual obstacles (partly white/black).
- **Definition 1 (Resolution Complete Coverage)**: A robot path is resolution complete if it passes through all *completely free* cells.
- **Definition 2 (Exact Coverage)**: A robot path achieves exact coverage if it passes through all cells (completely or partially free) such that the coverage tool covers the entire $Q \setminus O$. Resolution complete coverage approaches exact coverage as cell size approaches zero.

- **Definition 3 (Coverage Conducive Region)**: A region Q is coverage conducive if $Q \setminus O$ is a contiguous region formed by the union of $2D \times 2D$ square cells/grids. The paper notes that non-overlapping coverage is impossible for non-conducive regions when exact coverage is required.

3.2. Adjacency Graph Construction

The CPC algorithm incrementally constructs a spanning tree on an adjacency graph formed by "minor nodes".

- **Major/Subcells and Nodes**: Major cells (2D x 2D) have a major node at their center. Subcells (D x D) have subnodes at their center. These nodes correspond to physical spaces.

- **Minor Nodes**: This is a new concept introduced in the paper. Minor nodes (represented as triangles) do *not* have a physical existence but are associated with the boundaries of major cells. Each major cell has four minor nodes, one for each of its north, east, west, and south boundaries. The spanning tree in CPC is constructed over these minor nodes, unlike STC which uses major nodes.

- **Minor Node Connectivity**:

    - **Inter-cellular Minor Node Connectivity**: Two minor nodes in adjacent major cells (e.g., 'C' and 'N') are connected if the robot can move between 'C' and 'N' through their common boundary (ACN). This boundary is considered "permeable" if the robot can pass through it.
        - **Definition 4 (Permeability)**: A major cell boundary ACN is permeable if, when extended by length D at both ends (forming $\tilde{A}CN$), there is a free segment of length D in $\tilde{A}CN$. If ACN is permeable, the corresponding inter-cellular minor nodes are connected.
    - **Minor Node Multiplication**: This occurs when a major cell boundary has more than one connected component of free space, leading to multiple paths connecting two adjacent major cells. This can "double" or "triple" minor nodes (e.g., wN splits into $w1N$ and $w2N$). The paper primarily assumes these situations do not arise for its main analysis but discusses them separately.
    - **Intra-cellular Minor Node Connectivity**: Minor nodes *within* a major cell are considered adjacent based on connectivity, depending on obstacles. All four minor nodes within a major cell are considered adjacent if they are connected and a robot can move between corresponding neighbors through the major cell.
- **Clustered Minor Nodes**: The intra-cellular minor node adjacency graph clusters minor nodes into completely connected components. A "clustered minor node" is defined as each maximal subset of connected minor nodes within a major cell.

- If a major cell is free or the free region within it is a single topologically connected set with all boundaries permeable, all four minor nodes form a single clustered minor node, making it an "effectively free major cell".
- Multiple clustered minor nodes can form if the free region is disconnected (e.g., four, three, or two clustered nodes as illustrated in Fig. 7).
- A "blocked" minor node is one not connected to an adjacent minor node in a neighboring major cell (due to an impermeable boundary). If all minor nodes of a major cell are blocked, the cell is considered completely occupied as the robot cannot enter it. Unlike STC, CPC only considers a major cell blocked if *all four* minor nodes are blocked.

3.3. Details of the CPC Algorithm (Algorithm 1)

The CPC algorithm incrementally constructs a spanning tree on the adjacency graph of minor nodes using a depth-first search (DFS) approach.

Algorithm Steps (Recursive function `CPC(P, C)` where `P` is previous cell, `C` is current cell, initialized with `P`=null, `C`=starting cell `S`):

1. **Mark Current Minor Node and Cluster as "Old"**: The minor node (`lC`) corresponding to the entry point and all other minor nodes within its cluster are marked "old". This implies that once a minor node is set to "old," its entire cluster is also considered "old".
2. **Scan for Prospective Minor Nodes**: The robot scans neighboring major cells in an anticlockwise direction to find "prospective" minor nodes. A "prospective" minor node is a "new" minor node connected to the current minor node in the minor node adjacency graph.
3. **Construct Spanning Tree Edge**: An edge is constructed from the current minor node to the first available prospective minor node (`iNj`).
4. **Mark `iNj` and its Cluster as "Old"**: Similar to step 1, `iNj` and all minor nodes in its cluster are marked "old".
5. **Create and Follow Path**: A path is created through the corresponding subnodes on the *right side* of the spanning tree edge. The robot moves along this path, avoiding obstacles using a wall-following algorithm, covering subcells including partially occupied ones to ensure exact coverage.
6. **Update Cells and Recurse**: The current cell becomes the previous (`C → P`), and the newly entered cell becomes the current (`Nj → C`). The `CPC` function is then recursively called with the new `(P, C)`.
7. **Branch End Handling**: If no more prospective minor nodes are found from the current cell (end of a spanning tree branch), the robot circumnavigates along the constructed spanning tree via subnodes to reach the current parent cell (or a previously visited cell). This avoids retracing the path.
8. **Termination**: The algorithm terminates when the robot returns to the subcell it started from.

The graph-level path through subnodes is created on the right side of spanning tree edges and circumnavigates the edges on return, similar to STC.

4. Analysis of the Proposed Algorithm (Mathematical/Formal Properties)

The analysis compares CPC with STC and highlights several properties (P1-P8) that form the basis for the theorems.

- **P1**: The adjacency graph over free minor nodes is equivalent to the adjacency graph over clustered minor nodes.
- **P2**: The algorithm constructs a spanning tree over the adjacency graph of *clustered minor nodes*, unlike STC which uses major nodes. This is because creating an edge to one minor node effectively marks its entire cluster as "old".
- **P3, P4**: Graph-level robot paths are created on the right side of spanning tree edges and circumnavigate on reaching branch ends, similar to STC.
- **P5 (Free Major Cells)**: If a major cell is free (all minor nodes connected), the graph-level path covers subcells in a counterclockwise order (e.g., 1→2→3→4 cyclically). All subcells are visited exactly once by the robot.
- **P6 (Effectively Free Major Cells)**: P5 also applies to effectively free but physically occupied major cells. Unlike STC, CPC creates spanning tree edges and graph-level paths through such cells, ensuring all subcells are covered and each subnode is visited exactly once at the graph-level.
- **P7 (Partially Occupied, Not Effectively Free Major Cells)**: If such a cell has at least one permeable boundary, a spanning tree edge is created to its corresponding clustered minor node. The graph-level path will include all four subnodes.
    - If a clustered minor node has only one minor node, one spanning tree edge is created.
    - If two, two edges (one incoming, one outgoing).
    - If three, two or three edges.
    - If four, it's effectively free (see P5, P6).
    - At the graph-level, the number of repetitive coverage of a subnode equals the number of clustered nodes (at most four). However, the actual robot path using wall following results in substantially lower overlap.
- **P8**: No spanning tree is created to a blocked major cell.

Based on these observations, the paper states the following theorems (proofs follow from P1-P8):

- **Theorem 1**: The CPC algorithm makes a robot cover a coverage conducive region completely without any overlap. For such regions, CPC's path is identical to STC's.
- **Lemma 1**: With the CPC algorithm, the graph-level path passes through every subnode corresponding to every free or partially free major cell, at least once. This guarantees graph-level completeness.

- **Lemma 2**: With the CPC algorithm, every subnode corresponding to a free or effectively free major node appears exactly once in the graph-level coverage path. This guarantees a non-repetitive path at the graph-level for free or effectively free cells.
- **Theorem 2**: With the CPC algorithm, a robot using a suitable obstacle avoidance strategy (like wall following) covers all subcells corresponding to free or partially occupied major cells completely.

4.1. Comparison of CPC with STC/full-STC and BDC

- **Completeness**: CPC covers even partially free subcells, achieving exact coverage, unlike full-STC which only covers completely free subcells within a partially occupied major cell. STC is 2D x 2D resolution complete, and full-STC is D x D resolution complete, while CPC is complete in the exact sense.
- **Overlap and Retrace**: Full-STC's path deformation in partially occupied cells leads to complete retracing of path segments. In contrast, CPC ensures complete coverage of these partially occupied subcells *without path retraces* (if avoidable). While coverage overlap near obstacles is unavoidable for exact coverage, CPC results in *substantially lower coverage overlap* compared to full-STC. The overlap with CPC is equal to the area of the major cell occupied by obstacles.
- **Competitiveness**: The competitiveness of full-STC is established in prior work, implying CPC is also competitive.
- **Comparison with BSA/BDC**: While BSA and BDC also provide exact coverage, they don't guarantee non-overlapping coverage, even for free cells. They may involve "retracting" through covered regions to reach uncovered pockets. BDC can also lead to higher coverage overlap where cell width is not an exact multiple of D, especially near obstacles. CPC combines the advantages of STC (non-overlapping coverage for conducive regions) and BDC (exact coverage). CPC yields an optimal path that minimizes overlap while maximizing coverage, especially in partially occupied major cells.

4.2. Implication of Node Multiplication on Coverage

The theorems and analyses in the paper hold when minor node multiplication (doubling or tripling) does not occur. In coverage conducive regions, such situations do not arise, so Theorem 1 is always applicable. If node multiplication occurs, the algorithm may face difficulties in path generation due to multiple possible paths. The paper suggests a modification where the algorithm skips a connected minor node if it finds that the corresponding major cell boundary has more than one connected path. With this modification, CPC still retains completeness and competitiveness when no minor node multiplication is present, but it *may not provide complete coverage* for such specific major cells (illustrated in Fig. 17). For instance, parts of subcells might be left uncovered. STC would skip these entire major cells, while full-STC would cover any completely free subcells within them.

5. Implementation Issues

The paper discusses practical challenges for implementing CPC on a physical robot:

- **Obstacle Detection**: The algorithm relies on proximity sensors to detect obstacles for spanning tree construction and wall following. CPC is not highly sensitive to reasonable inaccuracies in obstacle detection; performance is expected to degrade only marginally. In contrast, STC can leave a $2D \times 2D$ region completely uncovered due to a small obstacle detection error ($d \approx D$). CPC would cover most of the region, leaving only a small area wrongly assumed to be occupied by the obstacle.
- **Localization**: Accurate localization is crucial for path planning. In approximate cellular decomposition, the robot's movement can cause the constructed grid to deviate from the initial grid. While STC might fail if an assumed-free cell is later found to have an obstacle on the return path, CPC mitigates this using obstacle avoidance by wall following. Performance degrades gracefully with nominal localization error but may become unsatisfactory if the error is comparable to the robot's size (D). The paper acknowledges that a detailed formal analysis of sensory noise is future work.

6. Results and Discussion

The paper provides both simulated and experimental results to demonstrate the CPC algorithm's performance.

- **Coverage Path Comparison (Qualitative and Quantitative)**:
    - For an environment with 9.375% obstacles, STC covered only 82.75% of the free area, skipping entire partially occupied major cells (25% of total area) [57, Fig. 19(a)].
    - Full-STC covered about 96% of the free space but skipped partially occupied subcells (14% of total area) and introduced 5% path retracing, leading to 5% coverage overlap [57, Fig. 19(b)].
    - **CPC** achieved exact coverage of the whole free space with no retraced paths, though overlap occurred around obstacle boundaries (which is unavoidable for exact coverage in such scenarios) [58, Fig. 20(d)].
    - In an "office-like" environment, CPC achieved complete coverage in the exact sense, while STC and full-STC left most of the interior regions uncovered [59, Fig. 21, 22].
    - BDC for the same office environment showed higher coverage overlap due to "non-coverage conducive" cells (cells with width not an exact multiple of D), in addition to overlaps from retracting [60, Fig. 23].
- **Simulation (TurtleBot in ROS/Gazebo)**:
    - The CPC algorithm was implemented and simulated on a TurtleBot within the ROS/Gazebo environment [61, Fig. 24(a),(b)].
    - Results showed that CPC covered all partially occupied cells that STC failed to cover, ensuring exact coverage [61, Fig. 24(c),(d)].

    - Despite a laser sensor accuracy of `±30 mm` and a robot footprint `D = 360 mm`, the robot successfully covered the entire obstacle-free region. Performance degradation (coverage gaps or overlap) due to sensory errors was observed but remained nominal.
- **Experiments (Fire Bird V Robot)**:
    - Experiments were conducted using a Fire Bird V differential wheeled robot.
    - To manage localization challenges in a lab environment, the authors printed `D × D` grids with solid black lines [65, Fig. 25]. The robot used line following sensors to follow these grids, which served as directional and relative positional guides. Wheel encoders were used when the robot moved off the grid for obstacle avoidance.
    - The robot successfully moved along the planned path (as in Fig. 20(d)), demonstrating exact coverage [66, Fig. 26]. The experiment served as a demonstration of the algorithm's feasibility.

7. Conclusions and Future Work

The paper successfully proposes the CPC algorithm, an online CPP algorithm based on approximate cellular decomposition. It achieves exact coverage with reduced overlap by introducing the concept of minor nodes and constructing a spanning tree over their adjacency graph. The algorithm's competitiveness and completeness are formally established under the assumption that minor node doubling/tripling does not occur.

Future work includes:

- Refining the algorithm to handle minor node doubling/tripling situations for guaranteed exact coverage.
- Modifying the algorithm to generate back-and-forth motion instead of spiraling motion.
- Full implementation of CPC on a mobile robot, addressing all practical issues (e.g., detection of cell permeability using onboard sensors, suitable wall-following for actual robot path translation).
- A formal and detailed quantitative analysis of how coverage performance varies with sensory errors.

This detailed review covers the core aspects of the paper, including its motivation, methodology, formal analysis, and empirical validation, without incorporating information from outside the provided sources.