

**Name:** Rohan Hemant Nemade.

**Class:** TY IT A

**Roll No:** 331013

**Batch:** A1

**Title:** *Comparative Study of DFS and Monte Carlo Tree Search (MCTS)*



Aim:

To compare the performance and computational characteristics of Depth-First Search (DFS) and Monte Carlo Tree Search (MCTS), and understand their respective algorithmic structures, practical applications, and suitability for solving different types of problems in both deterministic and uncertain environments.



Theory:

*Depth-First Search (DFS):*

DFS is one of the fundamental algorithms in computer science, used to explore nodes and edges of a graph. It explores a branch of the tree or graph as deeply as possible before backtracking. This method is especially effective for exhaustive search in puzzles, scheduling, and hierarchical structures.

* **Traversal Pattern:** Progresses depth-wise along a path until a terminal point is reached, then retraces steps to explore alternatives.
* **Approach Type:** Deterministic and uninformed; it does not incorporate knowledge about the goal.
* **Memory Requirements:** Efficient in terms of memory usage—typically linear relative to the depth of the structure.
* **Drawbacks:** May not yield the shortest solution path. In the presence of cycles, and without proper handling, it risks entering infinite loops.

*Monte Carlo Tree Search (MCTS):*

Monte Carlo Tree Search is an advanced, probabilistic search strategy used prominently in artificial intelligence, particularly for decision-making in uncertain or complex domains. Gaining attention through its application in systems like AlphaGo, MCTS combines random simulations with statistical analysis to iteratively improve decision quality.

* **Exploration Strategy:** Balances between exploring new possibilities and refining existing, promising options.
* **Nature of Operation:** Adaptive and heuristic-guided; relies on probabilistic feedback.
* **Core Process:** Consists of four iterative phases—selection, expansion, simulation (or rollout), and backpropagation—that collectively build and refine a decision tree.
* **Advantages:** Exceptionally effective in environments that are too vast for exhaustive exploration or when outcomes are uncertain.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Complexity Analysis: | | | |  |  |
| Algorithm Time  Complexity | | | | Space Complexity | Notes |
| **DFS** | | O(V + E) | | O(V) | Ideal for traversing structured or smaller graphs. |
| **MCTS** | | O(N × D) | O(N) | N is the number of simulations; D is average tree depth. Performance scales with the number of iterations. | |



Applications:

*DFS Applications:*

* **Maze and Puzzle Solving:** Useful in identifying any valid path to the goal by deeply exploring options.
* **Topological Sorting:** Essential for ordering tasks in dependency graphs.
* **Graph Traversal and Pathfinding:** Explores all potential paths, though not guaranteed to be optimal.
* **Cycle Detection**: Commonly applied in graphs to detect loops or repeated nodes.

*MCTS Applications:*

* Artificial Intelligence in Games: Widely implemented in strategic board games like Go and Chess for intelligent move prediction.
* **Robotic Decision-Making:** Enables evaluation of multiple action outcomes through simulation.
* **Uncertain Environments:** Well-suited for domains like poker or planning where not all information is available.
* **Real-Time Strategy (RTS) Games:** Facilitates rapid, adaptive decision-making within massive state trees.

Example and Algorithm:

**Problem:** Solving a simple maze or puzzle (e.g., 8-puzzle or tic-tac-toe)

**DFS Pseudocode:**

DFS(node):

if node is goal:

return solution

mark node as visited

for each neighbor of node:

if neighbor not visited:

DFS(neighbor)

**MCTS Pseudocode:**

MCTS(root):

for i = 1 to N iterations:

node = Selection(root)

child = Expansion(node)

result = Simulation(child)

Backpropagation(child, result)

return best move from root



Comparative Table:

|  |  |  |
| --- | --- | --- |
| Criteria | DFS | MCTS |
| Search Strategy | Depth-first traversal | Random simulation + statistical updates |
| Determinism | Deterministic | Probabilistic |
| Memory Usage | Low (linear in depth) | Moderate (grows with simulations) |
| Optimality | Not guaranteed | Improves with iterations; approximates optimal |
| Environment  Suitability | Deterministic, structured | Complex, dynamic, and uncertain |
| Implementation | Simple, recursive | Complex (requires tuning and simulation control) |
| Scalability | Limited in large graphs | Scales well with computational power |



Conclusion:

Both Depth-First Search (DFS) and Monte Carlo Tree Search (MCTS) offer distinct advantages and are suited for different categories of problems. DFS remains a strong choice for problems involving well-defined structures and where any valid solution is acceptable. Its simplicity and low memory overhead make it ideal for classic graph-related applications.

On the other hand, MCTS is highly effective in environments that are either too vast for exhaustive techniques or inherently unpredictable. Its balance of exploration and exploitation, coupled with adaptive feedback, makes it a powerful tool for decision-making, particularly in artificial intelligence and game theory.

Selecting between DFS and MCTS should be based on the specific problem characteristics—such as determinism, size of the search space, memory limitations, and the need for optimality or adaptability.

