Algorithm	Execution time	Memory usage	Success Rate	Solution Optimality	Scalability
Depth First Search (DFS (	DFS can be moderately fast because it goes deep quickly, but in large graphs it may waste time exploring long incorrect paths.	DFS uses very little memory because it only stores: • Stack • Visited nodes	DFS can reach the goal if a path exists. It may take a long or wrong route, but it eventually finds a path.	DFS does not produce an optimal route. It always returns the **first** path it finds, not the shortest or least-cost one.	DFS does not scale well. When the graph grows deeper or larger, DFS may explore unnecessary long paths before backtracking.
Breadth First Search (BFS)	Fast for small graphs, but grows quickly as the graph gets bigger (O(V + E)).	High memory usage because it stores many nodes in the queue	Always finds a solution if one exists (complete algorithm)	Guarantees the shortest path in unweighted graphs.	Does not scale well for large graphs due to memory growth.
Uniform Cost Search (UCS (	**Time Complexity: O(E log V)**	Uniform Cost Search stores all generated nodes and paths in memory until the goal is reached. • **Space Complexity: O(V)**	Uniform Cost Search has a **high success rate**. It always finds a solution if one exists	Uniform Cost Search is optimal. For the Drone Driven Route problem, this ensures minimal battery consumption, travel time, or distance.	Uniform Cost Search has limited scalability. For largescale drone delivery systems, heuristicbased algorithms such as A* are more efficient and scalable.
A* Search	A* is generally faster than DFS and BFS because it explores only the most promising routes. However, with a weak heuristic, its performance may degrade **O(b^d)**	**A* uses high memory because it stores:** • **Priority Queue** • **Multiple partial paths** • **Cost information for each node** **O(b^d)**	A* always finds a solution **if one exists**, provided the graph is finite and connected.	A* produces an **optimal solution** when using an **admissible heuristic**. It guarantees the shortest or least-cost route.	A* scales **better than DFS and BFS**, but for very large problems (many delivery points), its memory usage can become a limitation.

Greedy Best-First Search	The Greedy algorithm has low execution time, making it suitable for real-time drone routing decisions. $O(n^2)$ Fast Suitable for Real-Time	TheGreedy approach requires minimal memory, which is suitable for resource constrained drone systems O(n) Memory	The Greedy approach requires minimal memory, which is suitable for resource constrained drone systems O(n) Memory	While the Greedy algorithm does not guarantee an optimal solution, it often produces near optimal routes. Not Optimal Near Optimal located in Local Optimum	The Greedy algorithm scales reasonably well for small to medium sized delivery scenarios.
Genetic Algorithm (GA)	GA may take longer than simple search algorithms because it evaluates many solutions over multiple generations.	Stores the population and fitness values; memory grows with population size and number of delivery points.	GA reliable finds a feasible route that visits all delivery points. Mutation and crossover help explore diverse solutions and avoid getting stuck.	GA produces near optimal or optimal routes, minimizing total distance or cost.	By adjusting population size and number of generations, it can handle more delivery points without exploring unnecessary paths.
Hill climbing	Hill Climbing requires more execution time than Greedy due to iterative solution improvement. Time depends on: Number of attempts, Number of neighbors Time Complexity $O(k \times n^2)$	Hill Climbing uses moderate memory, storing only the current and neighboring solutions. Memory Complexity: O(n)	Hill Climbing generally succeeds in producing a valid route but may get stuck in local optima. The solution isn't always the best; it can be improved by: Random Restart Sideways Moves	Hill Climbing improves solution quality compared to Greedy but does not guarantee global optimality. The solution improves gradually	Hill Climbing scales moderately and performs well on medium-sized problems. To explain: When the number of points increases: The number of neighbors increases The time increases

The best algorithm is **A* Search** because it provides an optimal solution with better performance and scalability compared to other search algorithms.