Optimizing Drone-Based Crop Monitoring: An Analysis of the Shortest Path Problem

Yul Sung

Received April 17, 2025
Accepted July 17, 2025
Electronic access Ausugt 15, 2025

The Shortest Path Problem is relevant across a wide range of domains, including search and rescue, logistics, and agricultural crop monitoring. This study compares the performances of three specific algorithms in solving the Shortest Path Problem for drone-based crop monitoring applications. A key constraint in this analysis requires the drone to return to its starting point, simulating real-world monitoring tasks. The three algorithms this study examines are the Traveling Salesman Problem (TSP) algorithm, Dijkstra's algorithm, and A*. TSP employs a brute-force approach, Dijkstra's algorithm relies on a cost function to determine the shortest path, and A* adds on to Dijkstra's method with heuristics. The performance of these algorithms is evaluated based on total path length and computational efficiency. Each algorithm was tested multiple times on datasets containing three, five, ten, or eleven randomly generated points of interest to reduce the effects of errors or outliers. In this study, TSP achieved the highest accuracy, producing routes on average 6.3% shorter than those generated by Dijkstra's and A*, with an average total path length of 284.6 units for 11 points. However, it suffered from exponential growth in computation time, reaching an average of 10.9 seconds at the same dataset size. In contrast, Dijkstra's and A* algorithms offered significantly faster performance—5.1×10^{-3} and 1.1×10^{-2} seconds respectively for 11 nodes—but produced longer paths. Dijkstra's algorithm demonstrated greater computational efficiency than A*, outperforming it by an average of 3.45×10^{-3} seconds. These findings imply that, although A* is traditionally considered superior to Dijkstra's algorithm, the unique constraint in this study—requiring the drone to return to its starting position—causes Dijkstra's to significantly outperform A* by approximately an order of magnitude in computation time. This scenario of returning to the starting position is more representative of real-world conditions, where drones must return to a base station for charging and maintenance. These findings have practical implications for agriculture and drone-based surveillance, indicating that TSP is ideal for small-scale scenarios where accuracy is critical, whereas Dijkstra's algorithm is better suited for larger-scale operations where computational speed is a priority.

Keywords: Pathfinding, Cost functions, Shortest Path Problem, Crop Monitoring, Algorithms, Agriculture

Introduction

In today's rapidly evolving technological landscape, pathfinding algorithms are employed across a wide spectrum of disciplines including package delivery, wildfire suppression, and agricultural crop monitoring.

Our previous work at NASA[1] involved using drones with pathfinding capabilities and convolutional neural networks to detect disease in plants. This experience provided foundational insight into agricultural drone operations, particularly in generating efficient flight paths to investigate specific points of interest. In our XPRIZE Wildfire team, we implemented a pathfinding algorithm for multiple drones to navigate and suppress wildfires. Building upon this algorithm, we compared our new approach against the Rapidly Exploring Random Tree (RRT) algorithm for navigating complex wildfire environments. These efforts emphasize the growing need for efficient pathfinding algorithms to meet the demands of autonomous drone capabilities.

At the core of this need lies the Shortest Path Problem[2]. This problem focuses on finding the optimal path between a starting node, a set of points to visit, and the end destination. The problem is represented by a 2D graph, with nodes and edges that represent the distances between them. These distances are what is used to determine the path that minimizes the total travel cost the most. In this study's situation, the nodes represent points of interest that the farmer needs to investigate using a monitoring drone. This paper aims to evaluate which of three commonly used Shortest Path Problem algorithms is most effective for real-world drone-based crop monitoring. This study introduces a unique constraint not typically addressed: requiring the drone to return to its starting position, simulating actual operational needs like recharging or maintenance.

The Traveling Salesman Problem[3] (TSP) aims to find the shortest route for a "salesman"—in this case a drone—to visit each city on a list once before returning to the starting point. This goal is accomplished by evaluating every possible permu-

tation of the routes. However, initial testing revealed that TSP experienced significantly high computation times. In fact, when graphing results in bar charts, the execution times for Dijkstra's and A* algorithms were so low that they appeared negligible next to TSP. As a result, a pruning function was implemented in the TSP algorithm to eliminate any permutations once the path exceeded a predefined upper bound. This limit was determined by finding the average distance between nodes, multiplying that value by the number of connections, and finally scaling it with an adjustable constant for flexibility. This modification drastically improved TSP's computation time and resolved the aforementioned issues. Dijkstra's algorithm[4], known as a greedy algorithm, keeps a list of unvisited nodes. Beginning at the start point, the algorithm iteratively chooses the node with the smallest tentative cost (distance). The algorithm then visits all neighbors of that node and updates their tentative costs if a shorter path is found. This process is continued until it arrives at the destination. The A* algorithm[5] approaches the Shortest Path Problem by considering both the cost to reach a node and the heuristic estimate of the remaining distance to the destination. A* selects the node with the lowest combined cost, continuing this process until the destination is reached.

Drone-based crop monitoring requires solving the Shortest Path Problem to efficiently visit multiple points of interest—such as crop fields—and return to a base station. This paper evaluates the performance of three prominent pathfinding algorithms—Traveling Salesman Problem (TSP), Dijkstra's, and A*—by comparing their computation time and total path length under identical conditions. Each algorithm is tested on the same set of points with a fixed starting position. Unlike previous studies, this work introduces the constraint of requiring drones to return to the starting point, simulating real-world operational demands. The findings aim to help farmers optimize energy usage during remote drone monitoring by identifying which algorithm offers the best balance between accuracy and computational efficiency.

The comparison of the algorithms will be accomplished in four clear steps. The first step will be establishing the testing environment: this means defining the scale of the 2-dimensional map, generating the nodes, and selecting a start point. The second step will involve implementing The Traveling Salesman Problem, Dijkstra's algorithms, and A* algorithm. The third step consists of data collecting and grading the speed and accuracy of the algorithms. The final step will be analyzing the data collected in step three to develop a conclusion. However, some limitations of this experimentation lie in the simplification of the problem. This analysis did not account for obstacles in between paths. In an ideal world obstacle can be ignored, but real-world applications need to take into account obstacles that can hinder the movement of the monitoring drones.

Methods

The Traveling Salesman Problem[6] (TSP) identifies the shortest path by evaluating all possible routes. Past papers and studies have applied TSP to optimize navigation through crop fields[7], supplement truck-drone delivery systems using base stations[8], and coordinate multi-drone package delivery with the assistance of trucks[9]. However, most of this prior work focuses on logistics-based applications or overlooks the importance of returning to a base station in agricultural contexts. This paper seeks to determine the use case of TSP in scenarios where a drone must return to the home base in an agricultural setting.

To improve computational efficiency, the implementation of the TSP algorithm includes a route termination feature: any route whose distance exceeds a calculated threshold is discarded. This upper limit is determined by multiplying the average distance between points by the total number of connections and an adjustable control factor. The number of connections is defined as the total number of points plus one, accounting for the return trip to the starting location. The control factor allows dynamic adjustment of the extent the algorithm prunes suboptimal paths. The application of this pruning function significantly reduced TSP's computation time, making it more comparable to the other algorithms under evaluation. The equation used to calculate the limiting function, along with the corresponding pseudocode for the TSP implementation, is provided below.

$$\text{limit} = \text{avg\_distance} \times (\text{num\_points} + 1) \times 0.75 \qquad (1)$$

```
Set starting and end point the same;
Generate all (n-1)! permutations of routes;
Calculate the distance for each permutation;
   If cost > limit:
      Terminate route;
   If total cost < current shortest route:
      Set as new current shortest route;
Return the route with the minimum cost;
```

Dijkstra's algorithm[10] determines the shortest path by selecting the node with the smallest tentative cost (distance) at each step. It has been applied in past studies to optimize delivery routes for fresh agricultural products[11], improve the inefficiency of farm product collection and supply systems, and enable drone-based supply delivery to hard-to-reach areas[12]. However, similar to the Traveling Salesman Problem, many of these studies either failed to create scenarios where the agent must return to the start position or apply the algorithm in an agricultural context.

```
Mark the start node with a current
distance of 0;
Set all other nodes to infinity;
Set every node as \non-visited";
```

```
While still unvisited nodes:
   Select closest unvisited node;
   For each neighbor of current node:
      Calculate tentative distance;
      Update neighbor's distance
      if shorter;
   Mark current node as visited;

Return route;
```

The A* algorithm[13] finds the shortest path by combining two key factors: the actual cost from the start node to the current node, denoted as $g(n)$, and the estimated cost from the current node to the goal, denoted as $h(n)$. These are summed into the total cost function: $f(n) = g(n) + h(n)$.

In this study, the heuristic function $h(n)$ is defined as the Euclidean distance, which effectively models the straight-line flight path a drone would follow in open, unobstructed agricultural environments.

$$g(n) = \text{cost from start node to the current node (n)} \quad (2)$$

$$h(n) = \text{heuristic cost from current node (n) to end node} \quad (3)$$

$$f(n) = g(n) + h(n) \quad (4)$$

```
Open list that holds all nodes;
Closed node that tracks all nodes visited;

While open_list != empty:
   Choose node with lowest f(n) value;
   Remove n from open_list and add to
   closed_list;
   For each neighbor of n:
      If neighbor is in closed_list:
         Skip;
      Calculate tentative g score;
      If tentative g score < previous
      g score:
         Update current_node;
         g score of neighbor = tentative
         g score;
         Calculate f score for neighbor;
         Update heuristic estimate;

   If open_list empty && current_node !=
   final_node:
      Terminate, no path available;
```

The algorithms were performed on an Apple Mac Mini M1 with 8GB of memory, using PyCharm version 2024.1.4 64-bit evaluation copy. The methods for this experiment are as follows: initialization, algorithm execution, and data collection.

```
points = pd.DataFrame
({'x': np.random.rand(num_points)
* 100, 'y': np.random.rand(num_points) * 100})
computation_time = time.perf_counter()
- start_time
total_distance +=
calculate_distance(points.iloc[path[i]],
points.iloc[path[i + 1]])
```

Results

In terms of total distance traveled, Table I demonstrates the Traveling Salesman Problem's ability to return shorter routes compared to Dijkstra's and A* algorithm. For three nodes, all algorithms yielded the same distance due to the limited routes possible. However, divergences in the paths traveled by the algorithms began as the number of nodes increased to five. Although many experiments still resulted in equal distances, TSP periodically found shorter routes, as reflected in the average values shown in Table I. The flight paths in Figure 1 further visualize these variations, highlighting how algorithmic behavior begins to diverge with the addition of more nodes. By the time the node count reached eleven, TSP consistently found shorter paths, averaging 284.6 units traveled, while Dijkstra's and A* covered 318.0 units on average. The data highlights TSP's accuracy compared to Dijkstra's and A*, especially as the number of nodes increased. While Dijkstra's and A* algorithms were accurate with smaller datasets, they had higher chances of choosing a nearest point that differed from the true shortest path as the dataset increased. On average, TSP found a route 6.3% shorter than Dijkstra's and A*, making it the most accurate pathfinding algorithm of the three.

In terms of computation speed, the Traveling Salesman Problem demonstrated the fastest performance with fewer points. However, as Table II shows, TSP became inefficient as the number points increased as the computation time grew exponentially from 1.7×10^{-5} seconds with three nodes to 10.9 seconds with eleven nodes. This sharp increase in time exposes TSP's permutational limitations. In contrast, Dijkstra's algorithm had a more gradual increase in computational time, rising from 4.9×10^{-4} seconds for three nodes to 5.1×10^{-3} seconds for eleven nodes. The A* algorithm performed just below Dijkstra's, taking 1.1×10^{-3} seconds for three nodes and 1.1×10^{-2} seconds for eleven nodes. Similarly to Dijkstra's algorithm, the A* algorithm starts out slower than TSP, but eventually executes faster as the number of nodes increases. Overall, the computation time data reinforces the notion that TSP excels with fewer points, while also demonstrating the capabilities of Dijkstra's and A* algorithms as the number of points increases.

Table 1 Average total distance traveled

Algorithm	3 Nodes	5 Nodes	10 Nodes	11 Nodes
Traveling Salesman Problem	193.4 units	212.3 units	270.7 units	284.6 units
Dijkstra's	193.4 units	220.6 units	293.1 units	318.0 units
A*	193.4 units	220.6 units	293.1 units	318.0 units

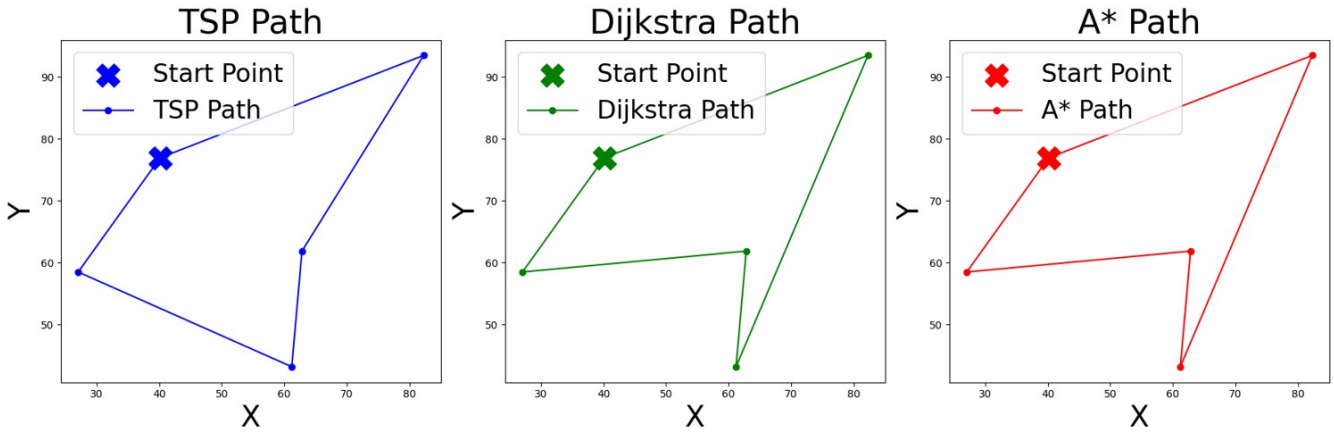

Fig. 1 Flight path of each algorithm with five points (Matlab)

Table 2 Average computation time

Algorithm	3 Nodes	5 Nodes	10 Nodes	11 Nodes
Traveling Salesman Problem	1.7e-5 sec	6e-5 sec	0.9 sec	10.9 sec
Dijkstra's	4.9e-4 sec	8.1e-4 sec	3.3e-3 sec	5.1e-3 sec
A*	1.1e-3 sec	2.4e-3 sec	9e-3 sec	1.1e-2 sec

Discussion

In this study, by evaluating all possible permutations, the Traveling Salesman Problem yielded the most accurate results. For smaller datasets, TSP computed solutions faster than both Dijkstra and A* algorithms due to the reduced complexity and fewer permutations. However, TSP has major scalability drawbacks, making it impractical for large datasets. As the number of points increases, computation time grows exponentially, with eleven points taking approximately 10.9 seconds on average. Despite the drawbacks, TSP remains accurate and can be applied in various scenarios.

Although the farmer's 5-field monitoring case was not tested in the real world, results show that TSP yields routes 6.3% shorter than Dijkstra's or A* across 11 nodes. This suggests that for small numbers of spatially dispersed points, where flight time is a major cost, TSP may be the most optimal method. While Dijkstra's and A* algorithms would yield routes more quickly, TSP finds the shortest route. Since the crop fields are far apart, even small differences could be costly, making accuracy a higher priority than computation time. What appears to be 1 unit of distance on a map could represent a meter, a kilometer, or even a mile. Therefore, in scenarios where accuracy or optimal distance is critical, TSP is the superior algorithm.

Statistical analysis using Repeated Measures ANOVA, which compares the averages of multiple groups measured under different conditions, revealed a statistically significant difference in average total distances among the three algorithms (p ¡ 0.05). Testing afterwards confirmed that TSP's shorter path lengths were significantly better than those of Dijkstra's and A*, especially at higher node counts. Similarly, an ANOVA on average computation time also found significant differences (p ¡ 0.001), showing that TSP's computation time increased sharply with more nodes, while Dijkstra's and A* remained efficient. These results reinforce the conclusion that TSP is best suited for small-scale precision operations, but not for real-time or large-scale tasks.

In contrast, Dijkstra's algorithm excels in computational efficiency, particularly when handling larger datasets. It outperformed both A* and TSP, especially when there are more than

five points of interest. In this study, Dijkstra's algorithm computed approximately 30% faster than A* algorithm on average. However, Dijkstra's algorithm failed to consistently yield the most accurate path as the dataset grows. An increase in the number of points raises the probability of choosing the incorrect closest point. Despite this drawback, Dijkstra's algorithm remains accurate in simple situations with smaller datasets. Additionally, its usefulness extends to situations involving large datasets or when minimizing computation time is more important than achieving the most accurate result. The measurements indicate that Dijkstra's is approximately 30% faster than A* (5.1×10^{-3}s vs. 1.1×10^{-2}s at 11 nodes). Thus, in scenarios demanding rapid response—such as diagnosing sporadic plant health issues—Dijkstra's may be beneficial. However, this inference assumes node distributions and urgency levels similar to those in this study; further real-world validation is needed.

The main advantage of the A* algorithm is its balance of computational speed and accuracy. By using heuristics to guide its search, the A* algorithm generally outperforms Dijkstra's in both computation time and accuracy—though in the trials conducted, this advantage was marginal due to the heuristic's limited benefit when the start and final positions were the same. While A* took longer than Dijkstra's in this experiment's setup, its strength lies in cases where the end node differs, allowing the heuristic to prune the search space more effectively. While A* did not outperform Dijkstra's in this study's same start and end position, literature[14,15] shows A* excels in environments with asymmetric start and goal positions and where obstacles are present. Accordingly, incorporating heuristics in real-world missions with non-symmetrical navigation points could enhance performance—though this remains to be empirically evaluated.

In this study, the results indicate that the Traveling Salesman Problem (TSP) produced the most accurate paths, generating routes on average 6.3% shorter than Dijkstra's and A*. However, it also experienced exponential growth in computation time, reaching 10.9 seconds for 11 nodes. Dijkstra's algorithm achieved the best performance in terms of computational speed, averaging only 5.1×10^{-3} seconds, and proved to be about 30% faster than A* on average. While A* was expected to outperform Dijkstra's due to its heuristic, its advantage was marginal in this study due to the start and end positions being the same. These findings achieve the study's initial objectives of determining which of the three algorithms would be most effective in the real world for agriculture applications. While TSP is ideal for low-node, high-accuracy tasks, Dijkstra's algorithm is better for large-scale, time-sensitive applications. A* offers a compromise, particularly when destination nodes differ from starting positions.

Some limitations of this study include the limited range of trials and node counts tested. By only examining node counts of 3, 5, 10, and 11, it risks obscuring trends observable at intermediate values. Future research will involve expanded testing across a broader range of node counts to capture more accurate results. Another significant limitation is the absence of obstacles in the simulation environment. While this simplification aligns with certain agricultural use cases, it does not represent the complexity of real-world environments where obstacles—such as trees, irrigation systems, or uneven terrain—can greatly influence pathfinding performance. Future work should incorporate 3D terrain data and obstacle modeling to evaluate how each algorithm adapts in constrained navigation scenarios. Finally, we plan to integrate Convolutional Neural Networks and hyperspectral imaging to detect crop disease, combining those outputs with our pathfinding algorithms for real-time autonomous navigation. This integrated approach will help validate our initial hypothesis[5] and potentially offer a system for smart agricultural monitoring.

Acknowledgments

Our previous work at NASA involved developing a system for drones to analyze crops for disease using convolutional neural networks. We have also worked on our AMSE Wildfire team, sponsored by the XPRIZE foundation, specifically the drone's ability to pathfind to fires. Additionally, we have competed in science fairs involving pathfinding algorithms and their applications to wildfire response.

References

1 S. Yotam Dubiner, J. Low, A. Nguyen, P. Singh and Y. Sung, *Plant Disease Detection: Exploring Applications in Hyperspectral Imaging and Machine Learning for Agriculture*, https://ntrs.nasa.gov/citations/20230013018, accessed Sep. 21, 2024).

2 A. Schrijver, *On the History of the Shortest Path Problem*.

3 D. Applegate, R. Bixby, V. Chvátal and W. Cook, *On the Solution of Traveling Salesman Problems*.

4 K. Magzhan and H. Jani, *A Review and Evaluations of Shortest Path Algorithms*.

5 *Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment* —, https://ieeexplore.ieee.org/abstract/document/9391698.

6 N. Sathya and A. Muthukumaravel, *A Review of the Optimization Algorithms on Traveling Salesman Problem*, https://doi.org/10.17485/ijst/2015/v8i1/84652.

7 I. Ait, E. Kofman and T. Pire, *A Travelling Salesman Problem Approach to Efficiently Navigate Crop Row Fields with a Car-Like Robot*, https://doi.org/10.1109/tla.2023.10130836.

8 S. Kim and I. Moon, *Traveling Salesman Problem With a Drone Station*, https://doi.org/10.1109/tsmc.2018.2867496.

9 Z. Luo, M. Poon, Z. Zhang, Z. Liu and A. Lim, *The Multi-visit Traveling Salesman Problem with Multi-Drones*, https://doi.org/10.1016/j.trc.2021.103172.

10 M. Xu, Y. Liu, Q. Huang, Y. Zhang and G. Luan, *An improved Dijkstra's shortest path algorithm for sparse network*, https://doi.org/10.1016/j.amc.2006.06.094.

11 Y. Tan and D. Wu, *Research on Optimization of Distribution Routes for Fresh Agricultural Products Based on Dijkstra Algorithm*, https://doi.org/10.4028/www.scientific.net/amm.336-338.2500.

12 A. Deaconu, R. Udroiu and C.- Nanau, *Algorithms for Delivery of Data by Drones in an Isolated Area Divided into Squares*, https://doi.org/10.3390/s21165472.

13 M. Wayahdi, S. Ginting and D. Syahputra, *Greedy, A-Star, and Dijkstra's Algorithms in Finding Shortest Path*, https://doi.org/10.25008/ijadis.v2i1.1206.

14 W. Zhang, J. Li, W. Yu, P. Ding, J. Wang and X. Zhang, *Algorithm for UAV path planning in high obstacle density environments: RFA-star*, https://doi.org/10.3389/fpls.2024.1391628.

15 S. Chakraborty, D. Elangovan, P. Govindarajan, M. ELnaggar, M. Alrashed and S. Kamel, *A Comprehensive Review of Path Planning for Agricultural Ground Robots*, https://doi.org/10.3390/su14159156.