Ant Colony System Shortest Path Solution Optimization Experiment

Charles Walker

06/23/2024

CS 350 / CS 528 Summer A, 2024

Embry-Riddle Aeronautical University

Daytona Beach campus

1 Aerospace Boulevard

Daytona Beach, FL 32114

Contents

[1. Introduction: 3](#_Toc170083031)

[2. Research Questions: 3](#_Toc170083032)

[3. Background: 3](#_Toc170083033)

[4. Model and Simulation Design: 5](#_Toc170083034)

[5. Simulation Verification and Model Validation: 9](#_Toc170083035)

[6. Experiment Design: 11](#_Toc170083036)

[6.1. Screening Process Experiment Design: 11](#_Toc170083037)

[6.2. Main Experiment Design: 13](#_Toc170083038)

[7. Experimental Results: 14](#_Toc170083039)

[8. Related Work: 18](#_Toc170083040)

[8.1. Beiranvand et al. (2017) Evaluation and reporting methods for algorithm comparison [2]: 18](#_Toc170083041)

[8.2. Chiarandini et al. (2007) Statistical evaluations as performance metrics [3]: 18](#_Toc170083042)

[8.3. Halim et al. (2020) How to assess metaheuristic optimization algorithms [4]: 19](#_Toc170083043)

[9. Conclusions and Future Work: 19](#_Toc170083044)

[9.1. Research Question 1 & 4: 19](#_Toc170083045)

[9.2. Research Questions 2 & 3: 20](#_Toc170083046)

[9.3. Future Work: 20](#_Toc170083047)

[10. References: 21](#_Toc170083048)

[A. Appendices: 21](#_Toc170083049)

# Introduction:

Many industries, such as transportation, computer networking, and circuit board manufacturers, have real-world problems that translate to the Shortest Path Problem (SPP). These industries benefit from algorithms that effectively find solutions to various forms of the SPP. This experiment uses a version of the Ant Colony Optimization (ACO) algorithm, called the Ant Colony System (ACS), to solve varying sizes of a directed graph single-pair SPP with three known best solutions. This experiment aims to determine if there are optimal independent variable values for the ACS algorithm when applied to varying sizes of the single-pair SSP.

# Research Questions:

1. Can the ACS algorithm dependent variables be adjusted to find a solution for a SPP within a known range of best solutions in an estimated average number of computations?
2. Do optimal independent values vary based on the problem's size?
3. Do optimal independent values vary based on the differences in edge costs?
4. What independent variables hold more weight in finding a solution, and at what problem sizes?

# Background:

The ACO algorithm is a referent of an ant colony that uses ants as homogeneous agents. It produces solutions by diffusion through the social interactions of ants using pheromones. An actual ant colony is extremely complex, so the ACO model's fidelity is limited to the aggregate behavior of the exploring ants. There is no communication or interaction between the ants other than the social rules regarding their pheromones. Each ant's degree of information consists of only the pheromone strength of the edges connected to its current node.

The ACO algorithm has five dependent variables that dictate the ants’ route optimization behavior. The five constants are the number of ants (k), the routes’ initial pheromone strength (τ0), the pheromone decay rate (ρ), the pheromone influence (α), and the heuristic influence (β), which is either the edge length (ηij) or total solution length (LAB).

The ACS version adds three alterations to the original algorithm: the local pheromone updating rule (Equation (4)), the global updating pheromone rule (Equation (3)), and the state transition rule (Equation (5)), which gives the sixth dependent variable (q0) [1]. Both pheromone rules are the primary update functions, which each update the pheromone strength of edges, making this model ambiguous.

Equation (1) is the primary decision-making method of the ants, where the probability of traveling to each possible node is calculated by dividing the product of the pheromone and heuristic strength of a single node by the sum of all other possible nodes’ pheromone and heuristic strength products. A random number between 0 and 1 decides each ant’s next location.

The secondary decision-making factor is the transition state rule, where j is the next node, Pij, max is the edge with the highest calculated probability and Pij,k is the primary decision-making method. Equation (6), a modified method adapted from [1], determines the initial pheromone strength. (c+1) represents the number of edges used in a solution and LAB, NN represents the route determined by the nearest neighbor heuristic method. This method simply finds a path by choosing the shortest edge at each node.

(1)

(2)

(3)

(4)

(5)

(6)

The global update function applies a pheromone update to all trails by decaying the pheromones based on the decay rate, ρ, a number from 0 to 1. Only the edges apart of the best solution found are increased by the product of ρ(1 / best solution). The local update function is designed to slightly reduce the pheromone on a single edge if an ant chooses it to encourage the other ants to explore alternate routes. This function updates as the ants choose their edges.

There are multiple ways to adjust the ACS algorithm, which will yield different results depending on the problem it's being applied to. This experiment aims to determine the answers to the three research questions by methodically adjusting the dependent variables and recording the results for statistical analysis.

The single-pair SPP is designed to have only one starting and one ending point. There are a series of columns with rows of nodes. The connections between the nodes are called edges; each has a cost, or distance, to traverse. Every node in a column has an edge connecting to a node in the next column. The graph is one-directional, meaning the ants can only traverse the graph from left to right and cannot backtrack to nodes on previous columns. There are no edges between nodes in the same column. The number of solutions to this type of SPP is dictated by the number of rows raised to a power equal to the number of columns. Another problem that must be addressed in the design is how to design the graph to allow for simpler validation of results.

# Model and Simulation Design:

In some cases, SPP models in simulations have randomness that may include solutions with various lengths or numbers of edges. However, this model will use a more controlled approach with a limited randomness factor. The graph is a one-directional single-pair SPP where the ants can only travel to a node in the next column. They cannot travel to another node in the same column or skip a node. All solutions use the same number of edges equal to one more than the number of columns in the graph, or C+1. Certain edges will have a range of possible costs depending on the number of columns in the graph. This is done in an attempt to prevent a bias towards creating an optimal ACS setting that only works for this particular problem in hopes of creating a good baseline that can be used for other problems.

Emphasis is applied to the graph design to automate its creation depending on the desired graph complexity while allowing for simple results validation. There are randomized routes and three known solutions regardless of the graph’s size. Equations (7) and (8) determine the best and least acceptable solutions. In both equations, c represents the number of columns used in the graph, which is always a positive odd integer greater than five. This constraint is implemented to simplify the experiment's validation by only using the three equations below. Equation (7) is used to terminate the experiments, which is explained further in Section 6. Experiment Design. Equation (9) determines the worst possible solution. All path examples are shown in Figure (2)

(7)

(8)

(9)

The number of possible solutions will be determined by rc, where r is the number of rows and c is the number of columns. Figure (1) below shows the graph layout and edge cost rules that were used to derive the three known solution equations mentioned above. These rules allow for a way to test the ACS algorithm independent variables against graphs that can be created larger with and without increasing the differences between edge costs and possible solution values.

Increasing the number of rows increases the graph complexity while the ranges of edge costs and solution values remain unchanged. Increasing the number of columns increases graph complexity, range of solution values, and edge costs. This can increase the possibility of convergence due to larger differences in edges and reduce the chance of exploration through portions of the graph. Adjusting these values allows further experimentation with the ACS algorithm’s ability to produce solutions depending on its applied problem and independent variable values. Figure (2) shows the routes for the known solutions, with an example of a solution whose value falls between the best and least acceptable. In both figures, C is the number of columns, and n is the Row number. The randomness in the graph rules allows for a range of possible solution values anywhere between the best and worst solutions. The range of all possible solution values, acceptable values, and rejected values can be determined using Equations (7), (8), and (9) regardless of the number of rows or columns used to create a graph.

With the simulation incorporating the ability to adjust the graph size both with and without affecting the difference in the edge costs and solution values, more detailed testing can be done to determine if there is a way to adjust the independent variables of ACS based on the differences in possible costs of edges and complexity of the graph.

A diagram of a network

Description automatically generated with medium confidence

*Figure (1): Color-Coated Graph Example with Rules for Edge Costs*

A diagram of a network

Description automatically generated

*Figure (2): Known Route Examples*

The simulation begins by requesting the desired control variable values, the maximum number of iterations, and the graph size from the user. The ant colony is modeled as starting from position A and seeking the shortest path to destination B. Each ant uses Equations (1) and (5) to navigate the graph. Each time an ant chooses an edge to traverse, that edge's pheromones are reduced using Equation (4) to encourage the next ant to explore other routes. After every ant reaches the destination, the simulation checks to see if any termination methods have been met. The two termination methods are finding a solution that equals Equation (6) or reaching the maximum number of iterations. If neither termination method is met, the simulation uses Equations (2) and (3) to update the pheromone trails globally by only increasing the pheromone level of the edges used in the best solution so far.

A diagram of a flowchart

Description automatically generated

*Figure (3): Simulation Software Process Flow Chart*

*A diagram of a computer program

Description automatically generated with medium confidence*

*Figure (4): Simulation UML Class Diagram*

# Simulation Verification and Model Validation:

This section documents the verification and validation of the model. Include functional requirements, test cases, and test results for verification. Include test scenarios and results for validation.

Verification: Ensuring that the simulation does what was intended.

Validation: Does the simulation perform as expected?

Requirements:

|  |  |
| --- | --- |
| Req 1. | The simulation shall create graphs with edge costs determined by the rules stated in Figure (1). |
| Req 2. | The simulation shall use a greedy path algorithm to determine the value of LAB, NN in Equation (6). |
| Req 3. | The simulation shall use Equation (6) to set the initial pheromone value for all graph edges. |
| Req 4. | The simulation shall use Equation (5) to determine which method an ant will use to choose the next edge. |
| Req 5. | The simulation shall use Equation (1) to choose the next path randomly. |
| Req 6. | The simulation shall use Equations (2) and (3) to globally update the pheromone levels of all edges after all ants have reached the destination. |
| Req 7. | The simulation shall use Equation (4) to update each edge’s pheromone levels after an ant has traversed it. |
| Req 8. | The simulation shall calculate the known solution values after creating a graph using Equations (7), (8), and (9). |

Requirement-Based Test Cases

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case # | Requirements Tested | Test Scenario / Pass Requirements | PASS/FAIL |
| 1. | 1 | Three graphs of sizes 5x5, 6x7, and 5x9 are created using the createGraph() function and tested to ensure each edge is the value dictated by the rules in Figure (1). / Every edge cost equals the value or is in the range of the values determined by the rules in Figure (1). | PASS |
| 2 | 2 | A custom test graph with a known cheapest path is created. The greedyPath() function is called, and its resulting path is tested against the expected path. / The resulting path equals the expected path. | PASS |
| 3 | 3 | A custom test graph and greedy path are passed to the setInitialPheromone() function. Pheromone levels for all edges are tested against an expected value, calculated as the inverse of the product of the greedy path's cost and the number of edges in that path. / All pheromone levels equal the expected value. | PASS |
| 4.1 | 4, 5 | The exploitOrExplore() function is passed two separate values that should force a true or false return. The results are tested to ensure the correct Boolean value is returned. / If the function returns a true when passed the exploit variable and a false when passed the explore variable. | PASS |
| 4.2 | 4, 5 | The calcPathStrengths() function is passed a custom double array of edge cost, pheromones, alpha value, and beta value. The resulting array of edge strengths is compared with an array of expected values. / The expected array equals the returned array. | PASS |
| 4.3 | 4, 5 | The selectStrongestPath() function is passed an array with the strongest function in index 4. The return of the function is tested to match this index. / The return integer matches the expected index number. | PASS |
| 4.4 | 5 | The createRandomPathProbability() function is tested by taking the sum of all elements in its returned array and ensuring it equals 1. / The sum of all elements in pathStrengths array equals 1. | PASS |
| 4.5 | 5 | The selectRandomPath() function is tested by passing an array of custom path strengths with the first four elements set to -1 and the last element set to 5. This ensures the selection process only selects the first value that’s less than the random number. / The function returns a 4. | PASS |
| 5 | 6 | The global update function is tested by passing it a test graph twice and checking the results of the graph after each update. / All non-best edges equal the nonBestPheromone value after the first update and the secNonBest variable after the second update. All edges of the best bath equal the bestPheromone variable after the first update and the secBest variable after the second update. | PASS |
| 6 | 7 | A test graph is created and passed to the setInitialPheromones() function in the equations class to set the needed initialPheromone variable. The expected pheromone value is set to the expectedPheromone variable. The localPheromonUpdate() function is called and its return is compared against the expectedPheromone variable. / The return of the function equals the expectedPheromone variable. | PASS |
| 7 | 8 | Three graphs of sizes 5x5, 6x7, and 5x9 are created using the createGraph() function, and the calcKnownSolution() function is called each time to calculate the results from Equations (7), (8), and (9). The path costs of the three known solution paths, as seen in Figure (2), are calculated and tested against the results from Equations (7), (8), and (9). / The calculated path costs of the three paths equal the calculated costs from the equations in all three cases of different graph sizes. | PASS |

# Experiment Design:

This simulation is used for a partial factorial experiment design or a fractional factorial design, as noted in reference [2]. Due to time constraints, testing all variables with different values to analyze emergent behavior between the models isn’t feasible. First, a screening process determines which variables should be blocked to create constants. Changing row and column sizes is crucial to answering research question three, so they will not be blocked, leaving them as independent variables. The five independent variables tested in the screening process are ρ, α, β, and q0. τ0 is set to the result of Equation (6) for every experiment, making it a dependent variable that relies on the results of the greedy algorithm and the size of the graph. Due to the unknown possibilities of these experiments' run times, k remained a controlled variable that changed depending on the experiment's needs and was not included in the tested independent variables, but the value of k was recorded for each experiment. After considering how graph size and differences in edge costs could affect the results, the row size and column size are also variables of interest but are not blocked. The goal of the screening process is to block two of the variables that are determined to have less effect on the results.

# Screening Process Experiment Design:

The screening process consists of running the simulation using baseline values for all independent variables. Baseline values are the initial values the variables begin with during the screening process. The rest of the values will have baselines of k = 5, ρ = 0.005, α = 1, β = 1, and q0 = 0.25. Each screening experiment consists of 50 tests that terminate when 1000 iterations are reached or when the best possible solution is found. The graph size will be 5 rows and 5 columns, giving 55 or 3125 possible solutions. Initial tests were run to determine the baseline values of the graph size and number of iterations. The above values resulted in consistent results, so they were chosen as the starting values. The results of each screening experiment are compared to the results of the initial baseline experiment. The results of interest for the screening experiments are recorded in Table 1 below. In Table 2, “Base” stands for the baseline value of the variable. “High” means plus 25% of its baseline value, and “Low” means minus 25% of its baseline value. These high and low values were arbitrarily chosen to test the weights of the variable's effects on the outcomes.

|  |  |
| --- | --- |
| Symbol | What the symbol represents |
| T | The number of times the best solution was found in an experiment. |
| μi | The average number of iterations needed to find the best solution in each experiment. |
| σi | The standard deviation of the number of iterations needed to find the best solutions in each experiment. |

*Table 1: Screening Data Recorded*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Experiment # / variable tested | ρ | α | β | q0 |
| 1/Baseline | Base | Base | Base | Base |
| 2/ ρ | High | Base | Base | Base |
| 3/ ρ | Low | Base | Base | Base |
| 4/ α | Base | High | Base | Base |
| 5/ α | Base | Low | Base | Base |
| 6/ β | Base | Base | High | Base |
| 7/ β | Base | Base | Low | Base |
| 8/ q0 | Base | Base | Base | High |
| 9/ q0 | Base | Base | Base | Low |

*Table 2: Screening Experiments’ Values*

|  |  |  |  |
| --- | --- | --- | --- |
| Experiment # / variable tested | SBEST / Percent Difference | μi / Percent Difference | σi / Percent Difference |
| 1/Baseline | 26 | 349.192 | 209.366 |
| 2/ ρ High | 26 / 0% | 263.346 / -24.58% | 181.017 / -13.54% |
| 3/ ρ Low | 32 / +23.08% | 353.844 / +1.33% | 206.972 / -1.14% |
| 4/ α High | 27 / +3.7% | 263.222 / -24.62% | 172.49 / -17.61% |
| 5/ α Low | 32 / +23.08% | 427.188 / +22.34% | 247.076 / +18.01% |
| 6/ β High | 23 / -11.54% | 301.087 / -13.78% | 146.869 / -29.85% |
| 7/ β Low | 34 / +30.77% | 301.971 / -13.52% | 215.158 / +2.77% |
| 8/ q0 High | 21 / -19.23% | 473.571 / +35.62% | 264.113 / +26.15% |
| 9/ q0 Low | 31 / +16.13% | 304.645 / -12.76% | 204.972 / -2.1% |

*Table 3: Results of Screening Experiments*

The screening process results in Table 3 show the variables' possible effects on the algorithm’s ability to find a solution. A larger difference in the average iterations needed to find a solution indicates a larger impact on performance. A larger difference in the standard deviation indicates a change in consistency. The two variables with the greater sum of absolute percent differences are q0 and α, so they will not be blocked for the rest of the experiments.

# Main Experiment Design:

The main experiment design involves comparing the results of different values on the same graph sizes. There are four total experiments, each running on a separate graph size of 6x7, 12x5, 5x9, and 18x5. Each experiment begins by running baseline values on graphs. All baseline values remain the same except k, which is increased to 10. These graph sizes are chosen because they each have a pair with similar amounts of possible solutions with different ranges of edges. Table 4 shows how each test in an experiment will be conducted. In the first column, L, ML, MH, and H correlate to the change in the value tested. Every test in this section will run until a max of 2000 iterations has been reached or the best value has been found. Each experiment will consist of 75 tests. The recorded data from each test is displayed in Table 5. Following the algorithm assessment methods covered in [3], the algorithm’s ability to find an acceptable solution determines its effectiveness, and the number of iterations required to do so determines its efficiency. The total number of possible solutions and the best possible, worst possible, and least acceptable solutions for each graph size tested are listed in Table 6 below.

|  |  |  |
| --- | --- | --- |
| Experiment # / variable tested | q0 | α |
| 1/Baseline | Base | Base |
| 2/q0 L | -30% | Base |
| 3/ q0 ML | -15% | Base |
| 4/ q0 MH | +15% | Base |
| 5/ q0 H | +30% | Base |
| 6/ α L | Base | -30% |
| 7/ α ML | Base | -15% |
| 8/ α MH | Base | +15% |
| 9/ α H | Base | +30% |
| 10/ α & q0 | Best performing value | Best performing value |

Table 4: Experiment Example for One Graph Size

|  |  |
| --- | --- |
| Symbol | What the symbol represents |
| TBSC | The number of times the best solution was found in an experiment. |
| μBSI | The average number of iterations needed to find the best solution in each experiment. |
| σBSI | The standard deviation of the number of iterations needed to find the best solutions in each experiment. |
| TLASI | The number of times an acceptable solution was found |
| μLASI | The number of iterations needed to find an acceptable solution in each experiment |
| σLASI | The standard deviation of the number of iterations needed to find an acceptable solution in each experiment |

*Table 5: Main Experiment Data Recorded*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Graph Size  Rows x Columns | Best Possible Solution | Least Acceptable Solution | Worst Possible Solution | Number of Possible Paths |
| 6x7 | 32 | 37 | 76 | 279,936 |
| 12x5 | 18 | 22 | 40 | 248,832 |
| 5x9 | 50 | 56 | 124 | 1,953,125 |
| 18x5 | 18 | 22 | 40 | 1,889,568 |

Table 6: SolutionGraph Size

# Experimental Results:

The raw data from each experiment are included in four separate Excel files.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Experiment # / variable tested | TBSC | μBSI | σBSI | TLASC | μLASI | σLASI |
| 1/Baseline | 11 | 676.909 | 95.177 | 75 | 74.32 | 59.6 |
| 2/q0 L | 24 | 525.083 | 235.776 | 75 | 59.467 | 45.418 |
| 3/ q0 ML | 20 | 601.75 | 342.972 | 75 | 72.213 | 43.935 |
| 4/ q0 MH | 2 | 346.5 | 98.874 | 75 | 91.373 | 61.757 |
| 5/ q0 H | 5 | 572.6 | 216.941 | 75 | 111.907 | 84.295 |
| 6/ α L | 18 | 946.278 | 499.67 | 75 | 85.853 | 65.946 |
| 7/ α ML | 14 | 737.429 | 448.675 | 75 | 96.52 | 65.65 |
| 8/ α MH | 5 | 318.4 | 99.533 | 75 | 72.547 | 61.392 |
| 9/ α H | 14 | 415.714 | 175.963 | 75 | 69.72 | 60.603 |
| 10/ q0L & αL | 30 | 1080.933 | 539.904 | 75 | 74.44 | 67.783 |

*Table 7: 6x7 Graph Experiment Results*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Experiment # / variable tested | TBSC Percent Difference | μBSI Percent Difference | σBSI Percent Difference | TLASC Percent Difference | μLASI Percent Difference | σLASI  Percent Difference |
| 1/Baseline | 11 | 676.909 | 95.177 | 75 | 74.32 | 59.6 |
| 2/q0 L | 118.182% | -22.429% | 147.724% | 0.000% | -19.985% | -23.795% |
| 3/ q0 ML | 81.818% | -11.103% | 260.352% | 0.000% | -2.835% | -26.284% |
| 4/ q0 MH | -81.818% | -48.811% | 3.884% | 0.000% | 22.945% | 3.619% |
| 5/ q0 H | -54.545% | -15.410% | 127.934% | 0.000% | 50.575% | 41.435% |
| 6/ α L | 63.636% | 39.794% | 424.990% | 0.000% | 15.518% | 10.648% |
| 7/ α ML | 27.273% | 8.941% | 371.411% | 0.000% | 29.871% | 10.151% |
| 8/ α MH | -54.545% | -52.963% | 4.577% | 0.000% | -2.386% | 3.007% |
| 9/ α H | 27.273% | -38.586% | 84.880% | 0.000% | -6.189% | 1.683% |
| 10/ q0L & αL | 172.727% | 59.687% | 467.263% | 0.000% | 0.161% | 13.730% |

*Table 8: 6x7 Graph Experiment Percent Difference Results*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Experiment # / variable tested | TBSC | μBSI | σBSI | TLASC | μLASI | σLASI |
| 1/Baseline | 10 | 743.6 | 380.374 | 75 | 35.64 | 37.381 |
| 2/q0 L | 20 | 701.35 | 375.024 | 75 | 29.107 | 27.812 |
| 3/ q0 ML | 11 | 624 | 272.368 | 75 | 36.853 | 31.046 |
| 4/ q0 MH | 4 | 710.5 | 444.67 | 75 | 34.107 | 29.726 |
| 5/ q0 H | 3 | 498.333 | 265.813 | 75 | 44.893 | 41.489 |
| 6/ α L | 14 | 990.429 | 556.504 | 75 | 37.133 | 35.238 |
| 7/ α ML | 6 | 902.5 | 325.868 | 75 | 35.52 | 32.695 |
| 8/ α MH | 9 | 726.222 | 398 | 75 | 36.347 | 39.426 |
| 9/ α H | 8 | 545.125 | 154.135 | 75 | 34.187 | 29.765 |
| 10/ q0L & αL | 19 | 944.684 | 451.168 | 75 | 31.72 | 30.344 |

*Table 9: 12x5 Graph Experiment Results*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Experiment # / variable tested | TBSC Percent Difference | μBSI Percent Difference | σBSI Percent Difference | TLASC Percent Difference | μLASI Percent Difference | σLASI  Percent Difference |
| 1/Baseline | 10 | 743.6 | 380.374 | 75 | 35.64 | 37.381 |
| 2/q0 L | 100.000% | -5.682% | -1.407% | 0.000% | -18.331% | -25.599% |
| 3/ q0 ML | 10.000% | -16.084% | -28.395% | 0.000% | 3.403% | -16.947% |
| 4/ q0 MH | -60.000% | -4.451% | 16.903% | 0.000% | -4.301% | -20.478% |
| 5/ q0 H | -70.000% | -32.984% | -30.118% | 0.000% | 25.962% | 10.990% |
| 6/ α L | 40.000% | 33.194% | 46.304% | 0.000% | 4.189% | -5.733% |
| 7/ α ML | -40.000% | 21.369% | -14.330% | 0.000% | -0.337% | -12.536% |
| 8/ α MH | -10.000% | -2.337% | 4.634% | 0.000% | 1.984% | 5.471% |
| 9/ α H | -20.000% | -26.691% | -59.478% | 0.000% | -4.077% | -20.374% |
| 10/ q0L & αL | 90.000% | 27.042% | 18.612% | 0.000% | -10.999% | -18.825% |

*Table 10: 12x5 Graph Experiment Percent Difference Results*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Experiment # / variable tested | TBSC | μBSI | σBSI | TLASC | μLASI | σLASI |
| 1/Baseline | 2 | 787 | 152.735 | 75 | 131.147 | 128.989 |
| 2/q0 L | 17 | 715.765 | 446.864 | 75 | 89.813 | 54.726 |
| 3/ q0 ML | 10 | 770.5 | 379.484 | 75 | 101.387 | 63.109 |
| 4/ q0 MH | 1 | 1080 | 0 | 75 | 158.093 | 154.624 |
| 5/ q0 H | 1 | 779 | 0 | 74 | 179.527 | 122.869 |
| 6/ α L | 7 | 1196.571 | 384.541 | 75 | 17.55 | 151.991 |
| 7/ α ML | 8 | 763.25 | 337.892 | 75 | 132.347 | 83.921 |
| 8/ α MH | 4 | 418.5 | 46.054 | 75 | 110.44 | 70.51 |
| 9/ α H | 2 | 459 | 513.36 | 75 | 94.133 | 80.414 |
| 10/ q0L & αL | 22 | 977.864 | 452.782 | 75 | 101.133 | 95.234 |

*Table 11: 5x9 Graph Experiment Results*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Experiment # / variable tested | TBSC Percent Difference | μBSI Percent Difference | σBSI Percent Difference | TLASC Percent Difference | μLASI Percent Difference | σLASI  Percent Difference |
| 1/Baseline | 2 | 787 | 152.735 | 75 | 131.147 | 128.989 |
| 2/q0 L | 750.000% | -9.051% | 192.575% | 0.000% | -31.517% | -57.573% |
| 3/ q0 ML | 400.000% | -2.097% | 148.459% | 0.000% | -22.692% | -51.074% |
| 4/ q0 MH | -50.000% | 37.230% | -100.000% | 0.000% | 20.546% | 19.874% |
| 5/ q0 H | -50.000% | -1.017% | -100.000% | -1.333% | 36.890% | -4.745% |
| 6/ α L | 250.000% | 52.042% | 151.770% | 0.000% | -86.618% | 17.833% |
| 7/ α ML | 300.000% | -3.018% | 121.228% | 0.000% | 0.915% | -34.939% |
| 8/ α MH | 100.000% | -46.823% | -69.847% | 0.000% | -15.789% | -45.336% |
| 9/ α H | 0.000% | -41.677% | 236.112% | 0.000% | -28.223% | -37.658% |
| 10/ q0L & αL | 1000.000% | 24.252% | 196.449% | 0.000% | -22.886% | -26.169% |

*Table 12: 5x9 Graph Experiment Percent Difference Results*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Experiment # / variable tested | TBSC | μBSI | σBSI | TLASC | μLASI | σLASI |
| 1/Baseline | 1 | 530 | 0 | 75 | 60.467 | 56.173 |
| 2/q0 L | 2 | 798.5 | 282.136 | 75 | 57.907 | 45.662 |
| 3/ q0 ML | 1 | 275 | 0 | 75 | 59.973 | 49.923 |
| 4/ q0 MH | 0 | 0 | 0 | 75 | 64.827 | 67.112 |
| 5/ q0 H | 1 | 507 | 0 | 75 | 66.8 | 59.967 |
| 6/ α L | 4 | 1156.75 | 346.55 | 75 | 52.093 | 50.557 |
| 7/ α ML | 4 | 1072.75 | 703.081 | 75 | 62.693 | 61.764 |
| 8/ α MH | 1 | 917 | 0 | 75 | 76.533 | 66.339 |
| 9/ α H | 2 | 477.5 | 133.643 | 75 | 58.267 | 51.501 |
| 10/ q0L & αL | 9 | 964 | 459.1184 | 75 | 67.653 | 69 |

*Table 13: 18x5 Graph Experiment Results*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Experiment # / variable tested | TBSC Percent Difference | μBSI Percent Difference | σBSI Percent Difference | TLASC Percent Difference | μLASI Percent Difference | σLASI  Percent Difference |
| 1/Baseline | 1 | 530 | 0 | 75 | 60.467 | 56.173 |
| 2/q0 L | 100.000% | 50.660% | N/A | 0.000% | -4.234% | -18.712% |
| 3/ q0 ML | 0.000% | -48.113% | N/A | 0.000% | -0.817% | -11.126% |
| 4/ q0 MH | -100.000% | -100.000% | N/A | 0.000% | 7.211% | 19.474% |
| 5/ q0 H | 0.000% | -4.340% | N/A | 0.000% | 10.473% | 6.754% |
| 6/ α L | 300.000% | 118.255% | N/A | 0.000% | -13.849% | -9.998% |
| 7/ α ML | 300.000% | 102.406% | N/A | 0.000% | 3.681% | 9.953% |
| 8/ α MH | 0.000% | 73.019% | N/A | 0.000% | 26.570% | 18.098% |
| 9/ α H | 100.000% | -9.906% | N/A | 0.000% | -3.638% | -8.317% |
| 10/ q0L & αL | 800.000% | 81.887% | N/A | 0.000% | 11.884% | 22.835% |

*Table 14: 18x5 Graph Experiment Percent Difference Results*

# Related Work:

This section presents information on work that others have done, highlighting both similarities and differences.

# Beiranvand et al. (2017) Evaluation and reporting methods for algorithm comparison [2]:

This paper explains the importance of benchmarking, which involves using premade test sets or problems for algorithms to provide solutions and create data that allows for fair assessments and comparisons. The only difference between the benchmarking process explained in this paper and the experiment process used here is that the graph used above tests a single algorithm’s efficiency and effectiveness against itself with various values. This paper explains how to measure efficiency by running time, fundamental evaluations, and memory usage, where the experiments above experiments used iterations needed to find solutions as fundamental evaluations. The paper covers how to assess a wide range of algorithms.

# Chiarandini et al. (2007) Statistical evaluations as performance metrics [3]:

This paper addresses the lack of methodology in analyzing metaheuristic algorithms to increase the scientific value of the data collected. There are two scenarios explained, which are similar to how the experiments in this paper were conducted, differing in how their data were examined. Scenario 1 explains studying solution costs and run times when a termination method is met. Scenario 2 covers the same while the algorithm runs until a certain termination method is reached. It further explains in scenario 2 the need to examine the solution cost over the run time and use chosen bounds on solution costs from known optimums or lower bounds. This correlates to the least acceptable solution found at how many iterations in this paper’s experiments. Where this paper’s analysis of the data differs is its explanation of how to represent the data in equations and graphs, whereas this paper's experiment results were analyzed through percent differences in averages over multiple tests.

# Halim et al. (2020) An exhaustive review of how to assess metaheuristic optimization algorithms [4]:

This paper covers extensive information from a large pool of research papers to explain effective ways to assess the quality of solutions produced by simulation-driven metaheuristic algorithms. The subjects covered range from how to graphically represent certain types of data, how to assess the quality of various algorithms, how to analyze and report results, and benchmarking. Only a fraction of the information covered in this paper applies to the experiments covered here.

# Conclusions and Future Work:

This section presents a brief recap of the work, concise answers to the research questions, and suggestions as to how the work can be extended.

# Research Question 1 & 4:

1. Can the ACS algorithm independent variables be adjusted to find a solution for a SPP within a known range of best solutions in an estimated average number of computations?
2. What independent variables hold more weight in finding a solution, and at what problem sizes?

From the start, it was known that not enough time was allotted to test all variables to answer these questions. The intent was to highlight which tested variables hold more weight in finding a solution and, depending on the results, state that the rest could be tested in the same manner in future work. Unfortunately, no conclusion can be drawn to answer these two research questions. There was an oversight in the rules that dictated the graph’s edge costs and the way the experiments were conducted, making it impossible to validate results regarding acceptable solutions. To determine the effectiveness and efficiency of the algorithm’s setup, the percentage of acceptable solutions in each graph needs to be known and consistent throughout each experiment.

In each test, a new graph was made. While this didn’t affect the screening process, it invalidated the main experiments. Two major components in the comparisons of each test were the number of times an acceptable solution was found and the average iterations needed to find an acceptable solution. It's impossible to know how many acceptable solutions existed in any graph tested. One test could have had 15% of its possible solutions in the acceptable range, while another test in the same experiment could have had 30% of its possible solutions acceptable. This makes it possible for a value that did actually perform better to appear worse or vice versa.

# Research Questions 2 & 3:

1. Do optimal independent values vary based on the problem's size?
2. Do optimal independent values vary based on the differences in edge costs?

While no conclusions can be drawn from the experiments regarding the tested independent values' capabilities to find a range of solutions, the number of best solutions found remains a valid test result because each graph only had one best solution.

These results are not all-inclusive, and more testing is needed to verify any conclusions. However, the results show that the same values of the tested variables, regardless of the problems’ size or range of edge costs, resulted in the best solution being found more often. This might point to these two variables having optimal values for finding the best solution. When these better-performing values from each tested variable were tested together, the best solution was found more often than with the other tests of each experiment.

# Future Work:

The graph rules need to be redesigned to solve the issue of not knowing how many acceptable solutions exist for the problem. This would involve analyzing the possible outcomes based on the new rules and deriving an equation that determines the number of acceptable solutions based on the graph size, similar to how the equations that determine the known solutions were derived.

Findings in further literature reviews should be considered when redesigning the graph model, code for the simulation, and experiment design. For example, examining the guidelines for test sets (the graph), what data to collect for experiments, and other proven methods of displaying algorithm-based experiment data for analysis, as mentioned in [4]. The experiment design was generally sufficient, although lacking in scale, but the simulation design invalidated the results. In future work, more time should be allotted to testing more variables and different combinations of values on different graphs.

# References:

[1] M. Dorigo and L. M. Gambardella, “Ant Colony System: A cooperative learning approach to the traveling salesman problem,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, Apr. 1997. doi:10.1109/4235.585892

[2] V. Beiranvand, W. Hare, and Y. Lucet, “Best practices for comparing optimization algorithms,” Optimization and Engineering, vol. 18, no. 4, pp. 815–848, Sep. 2017. doi:10.1007/s11081-017-9366-1

[3] M. Chiarandini, L. Paquete, M. Preuss, and E. Ridge, “Experiments on Metaheuristics: Methodological Overview and Open Issues,” Institut for Matematik og Datalogi Syddansk Universitet, Preprints vol. 4, Mar. 2007. ISSN No. 0903-3920

[4] A. H. Halim, I. Ismail, and S. Das, “Performance assessment of the metaheuristic optimization algorithms: An exhaustive review,” Artificial Intelligence Review, vol. 54, no. 3, pp. 2323–2409, Oct. 2020. doi:10.1007/s10462-020-09906-6

# Appendices:

All data is included in external files.

**Backlog**

**Table 15: Sprint 1 Backlog**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **User Story Description / Notion of Done** | **Est**  **Cost** | **Done** |
| 1 | Choose a type of simulation. | 8 | Done |
| 2 | Choose a problem to apply the simulation to. | 8 | Done |

**Table 16: Sprint 2 Backlog**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **User Story Description / Notion of Done** | **Est**  **Cost** | **Done** |
| 3 | Research types of Ant Colony Optimization algorithms to determine which could best be applied to the shortest path problem. /Found an algorithm to use. | 10 | Done |
| 4 | Researched the type of shortest path problem to solve for. | 5 | Done |
| 5 | Determined what research questions to solve. | 4 | Done |
| 6 | Finish the background, introduction, and research questions of the document. | 10 | Done |

**Table 17: Sprint 3 Backlog**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **User Story Description / Notion of Done** | **Est**  **Cost** | **Done** |
| 8 | Create a flow-chart of the simulation process to base the UML class diagram on. | 2 | Done |
| 9 | Create a thorough write-up of the model with a graph design and ensure the best and least acceptable solutions can be predicted regardless of graph size. | 12 | Done |
| 10 | Create a plan for data recording to automate as much of the process as feasibly possible. / A file type to store data has been chosen, along with a file organization method and the file contents of each experiment, and a plan to automate experiment results has been incorporated into the UML class diagram. | 1 | Done |
| 11 | Determine the limitations of experiments for the control variables, including maximum computations, convergence termination factors, range of control variable values, acceptable solution ranges, and what dictates the weight of a control variable. / A completed list of experiments with the required number of computations and values for control variables. A method of defining a weighted value and a way to test it has also been found. | 12 | Done |

**Table 18: Sprint 4&5 Backlog**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **User Story Description / Notion of Done** | **Est**  **Cost** | **Done** |
| 7 | Research C# program design methods. / All models of the simulation, minimal user interface, and file in/out processes have been planned out for C#. | 3 | Done |
| 12 | Create the initial UML class diagram. /The UML class diagram represents a fully functioning program. | 1 | Done |
| 13 | Create the function that creates the graph following the rules described in Figure (1) of the document. | 4 | Done |
| 14 | Create the main class / A main class has been created that requests and accepts all valid user input for the dependent variable values and termination methods. This also implies that it has been tested to ensure it works as intended. | 1 | Done |
| 15 | Create a class for the ant colony model / A class with a constructor method has been created for the ant colony. The constructor method uses user input to create the ant colony. This also implies that it has been tested to ensure it works as intended. | 1 | Done |
| 16 | Create a class for the graph model. / A class containing a constructor method and function that builds a graph using user input and the rules stated in Figure (1) has been created. This also implies that it has been tested to ensure it works as intended. | 1 | Done |
| 17 | Create the file class for the simulation. / A class that creates and writes the recorded data to excel files has been created. This also implies that it has been tested to ensure it works as intended. | 4 | Done |
| 18 | Create the control class for the simulation. / A class that controls the process of the simulation has been created. This also implies that it has been tested to ensure it works as intended. | 5 | Done |
| 19 | Run the first pilot tests to ensure the simulation is working as intended. The software and all test cases are complete. All test cases have been run and passed. | 19 | Done |

**Table 19: Sprint 6 Backlog**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **User Story Description / Notion of Done** | **Est**  **Cost** | **Done** |
| 20 | Start the experiment screen process to determine which independent variables must be blocked. / Using initial data collection from the screening process, at least two variables have been chosen to remain constant throughout the experiments. | 4 | Done |
| 21 | Review the literature on similar experiments to update the Experiment Design section of the document. /Examples have been found. | 3 | Done |
| 22 | Rewrite the Experiment Design section of the document to include references to justify design choices and clarify the experiments. / The Experiment Design section has been rewritten to reference prior experiments and justify design choices. | 1 | Done |
| 23 | Add tables to reduce the written contents of the Experiment Design section of the document. / Tables showing data to be collected from experiments has been included in the Experiment Design section. | 1 | Done |
| 24 | Rewrite the paragraphs of the Experiment Design document to condense written content and ease the digestibility of information. / The Experiment Design section has been condensed and made easier to understand. | 1 | Done |
| 25 | Finalize what content needs to be recorded and adjust the FileReadWrtie class to record that data. / The FileReadWrite class records the necessary data. | 1 | Done |
| 26 | Run all experiments and record all data. / All experiments have been run and data has been collected. | 11 | Done |
| 27 | Analyze the data and record the findings. / Results of the analyses have been recorded. | 6 | Done |
| 28 | Summarize the results and analysis of the experiments and determine future inquiries to look into. / The Conclusion and Future Work section of the document has been completed. | 6 | Done |
| 29 | Fill out the Related Works section. / The related works section of the document contains related works from other sources. | 1 | Done |