Ant System: Finding the Longest Path

Samantha Ball
1603701
School of Computer Science and Applied Mathematics
University of the Witwatersrand

***Abstract*—While many algorithms exist to solve the problem of finding the shortest path in polynomial time, the longest path problem remains a challenging task as it is NP-complete, therefore requiring more advanced approaches. In this study we investigate the use of the ant colony optimization metaheuristic in comparison to the more classical beam search. While ant colony techniques rely on a population-based approach, beam search provides a memory-efficient version of a traditional breadth first search, therefore providing a strong contrast between classical and more advanced methods.**

In particular, the popular Ant System (AS) algorithm and traditional beam search is utilised to find the longest path through a given maze. Ten different mazes of increasing size are evaluated in order to compare the two approaches. Several key challenges are introduced at each increase in size such as memory restrictions, algorithm efficiency and choice of heuristic. These factors are explored in-depth for each algorithm, demonstrating the respective benefits and drawbacks of ant system and beam search respectively.

From the analysis, several key findings provide insight into the challenges associated with the longest path problem. It can be observed that ant system is more suited to finding the longest path due to its versatility. Another crucial finding is that the underlying representation of the maze must be constructed in an efficient manner in order to solve larger mazes. Although optimisation was successfully employed for the underlying tree structure, further optimisation is required for the ant system and beam search algorithms to run within a feasible time on the largest maze. Thus, several key insights into the use of the ant system and beam search algorithms for the longest path problem were obtained, with ant system showing superior results on the majority of the maze problems.

***Index Terms*—ant system, beam search, longest path problem**

I. Introduction

Ant Colony Optimization (ACO) is a population-based optimization technique often used for combinatorial optimization problems such as the classical travelling salesman problem (TSP), quadratic assignment problem (QAP) and other graph-based problems. Ant Colony Optimization (ACO) draws from the natural behaviour of biological ants such as the laying down of pheromones to direct other members of the colony towards a desirable resource [1]. Due to this, ant-based techniques display advantageous performance for graph-based problems. Several variants of Ant Colony Optimization (ACO) have been developed, stemming from the popular Ant System (AS) [1].

In this investigation, we explore the use of Ant System (AS) to find the longest path in a given maze. Due to the natural advantages of ant-based optimization for solving graph problems, we model the maze as an interconnected graph structure with nodes representing decision points in the maze. Finding the longest path is a variation on the traditional shortest path problem. However, unlike the task of finding the shortest path, the longest path problem is NP-complete.

We compare the performance of the population-based Ant System metaheuristic to the heuristic beam search algorithm. Beam search provides a more efficient method of implementing breadth-first search by only exploring certain nodes chosen according to a predefined heuristic. Beam search is thus a greedy algorithm as it expands only the most favourable nodes at each iteration [2]. This diminishes the memory requirements of the search which is a key factor for larger problems.

As both algorithms, are heuristic-based approaches, suitable heuristic choice is vital for optimal performance of the search algorithms. The approaches are evaluated on ten mazes of differing size, therefore providing a thorough measure of both performance and efficiency. Therefore, through this investigation, a comparison of ant system and beam search is explored, particularly with respect to the trade-offs that are encountered in larger mazes in terms of both efficiency and the length of the path found.

The paper is structured as follows. In Section II, we outline the relevant techniques and core differences between the two search algorithms. In Section III, we provide an in-depth discussion of the algorithms, their specific design features, and chosen heuristics. Section IV presents our results and a comprehensive comparison of the two algorithms in terms of performance and efficiency. Lastly, Section V summarizes our key findings and discusses the core benefits and drawbacks of the search algorithms in the longest path context.

II. Background

The central functioning of Ant System and Beam Search is discussed below, together with a discussion of the longest path problem and its associated challenges.

A. Ant System

Ant System (AS) was the first ant-based algorithm proposed in the literature and is often considered as the most popular of the ant-inspired metaheuristics. Ant-based algorithms employ the concept of 'stimergy' or indirect communication, through the use of the application of pheromone along the paths in a graph structure.

The Ant System (AS) algorithm, assuming the shortest path problem, is summarized in the process below. A population of ants is initialised at the starting point of the graph. Each ant in the population then moves to an adjacent node in the graph according to the transition probability $P_{i,j}$. Once all the nodes have reached the end point, the ants retrace their discovered paths, laying pheromone on each path segment as a signal to other ants in the population. This process is repeated for several iterations.

Algorithm 1 Ant System

```
Let iteration counter $t = 0$
while $t < t_{max}$ do
  Initialize a population of $K$ ants at the starting node 0
  Initialize a small amount of pheromone $\tau_{i,j} = c$ on every edge in the graph
  Initialize the best solution $\hat{x}$
  while not all ants at end node do
    for $k$ in $population$ do
      Calculate heuristic information $\eta_{i,j}$
      Move to next node according to transition probability $P_{i,j}$
    end for
    Update best solution $\hat{x}$
    for edge $i, j$ in graph edges do
      Perform pheromone update for edge $i, j$
    end for
    $t = t + 1$
  end while
end while
return $\hat{x}$
```

A core feature of the Ant System (AS) algorithm is the use of heuristic information to guide the movement of the ant population. Heuristic information refers to additional knowledge regarding how beneficial a particular movement will be, often in terms of euclidean distance to the end of the path, in the case of a path-based problems.

The transition probability therefore includes both pheromone signals as well as heuristic information and can be written as follows,

$$P_{i,j} = \frac{(\tau_{i,j}(t))^{\alpha}(\eta_{i,j}(t))^{\beta}}{\sum_{e \in N_i^k}(\tau_{i,e}(t))^{\alpha}(\eta_{i,e}(t))^{\beta}} \tag{1}$$

where $\tau_{i,j}$ represents the amount of pheromone on the graph edge from node i to node j, $\eta_{i,j}$ represents the desirability of the move from node i to node j and α and β control the relative balance between the influence of pheromone and heuristic information.

The neighbourhood N_i of each graph node i and ant k depends on the neighbouring nodes j connected to node i as well as which neighbouring nodes are allowable for a specific ant k. An additional feature specific to the Ant System (AS) algorithm is the introduction of a tabu list for each ant k in the population. The tabu list serves as a memory device to record which nodes the ant k has already visited. The neighbourhood N_i^k then excludes visited nodes in order to prevent the ant k from revisiting the same nodes and introducing loops.

The last central feature of the Ant System (AS) algorithm is an increased elitism stemming from an emphasis on the best path found. This is implemented by the inclusion of an additional term in the pheromone update equation which ensures that further pheromone is laid on the best path found by the ant population as a whole. This pheromone update equation can be denoted as follows,

$$\tau_{i,j}(t+1) = \rho\tau_{i,j}(t) + \triangle\tau_{i,j}(t) + n_e\tau'_{i,j}(t) \tag{2}$$

and the update $\tau'_{i,j}(t+1)$ is

$$\tau'_{i,j}(t+1) = \begin{cases} \frac{Q}{f(\hat{x}(t))} & if\ (i,j) \in \hat{x}(t) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where ρ is the evaporation rate, $\hat{x}(t)$ is the best path found by all the ants in the population and n_e regulates the strength of the elitest force. The evaporation rate ensures that paths do not become oversaturated, therefore limiting exploration. The values of ρ and n_e are thus hyperparameters and require careful tuning for optimal performance.

B. Beam Search

Beam search is a classical local search method which provides an optimization of the popular breadth first search (BFS) algorithm by reducing the memory requirements. An overview of the beam search algorithm for the traditional shortest path problem is given below.

Algorithm 2 Beam Search

```
Initialize queue containing path starting at node 0
Initialize beam width $bw$
while len(queue) $\neq 0$ do
  Let path = queue.pop
  Let $x_i$ = path[-1]
  if $x_i$ = end node then
    Add path to $found\,paths$
  end if
  for node $j$ in $N_i$ do
    new path = path + node $j$
    Add new path to queue
  end for
  if len(queue) $>$ $bw$ then
    Sort queue according to heuristic cost
    queue = sorted queue$[0 : bw]$
  end if
end while
return found paths
```

where N_i is the neighbourhood of the current node x_i.

Beam search is a greedy approach as it selects only a predefined number of optimal nodes to be explored, allowing the remaining nodes to be pruned. The number of nodes explored at each stage is specified by the beam width bw. This allows for lower memory requirements as less nodes are stored and less paths are explored.

Thus, while ant system represents a population-based approach to combinatorial problems, beam search is more representative of classical local search methods. Therefore the algorithms provide a suitable contrast for investigation into the advantages and limitations of local search and population-based methods in a combinatorial context.

C. *Longest Path Problem*

While the above algorithms are traditionally formulated in terms of the shortest path problem, they can be adapted to find the longest path in a given maze. The longest path problem aims to find the simple path of maximum length in a given graph structure, where simple refers to the absence of repeated vertices. As outlined above, the longest path problem is NP-complete and thus requires efficient search strategies. The longest path problem can be formulated as follows,

$$\begin{gathered} P = (n_1, n_2, n_3.....n_n) \\ such\ that \sum_{i=1}^{n-1} f(e_{i,i+1})\ is\ maximised \end{gathered} \tag{4}$$

where the path P passes through nodes n_1 to n_n such that the sum of the distances of all the edges between the nodes is maximised. Here the length of edge e is represented as a function f mapping a given edge e to a distance. In the context of a maze problem, node n_1 and node n_n refer to the entrance and exit of the maze respectively.

III. Implementation

In order to compare the efficacy of the ant system and beam search algorithms, the methods were assessed on ten mazes of differing sizes. The specific design choices and hyperparameter values chosen to meet the problem requirements are discussed further in this section. Crucially, the underlying representation of each maze is discussed as further design choices hinge on the chosen representation.

A. *Maze Representation*

Each of the ten maze images was converted to a tree structure in order to represent the maze as a graphical problem. A tree structure was chosen to represent the maze in order to for increased prevention of loops.

To convert the maze images to a tree format, each image was read in as a binary array. For every pixel with value 0 representing a valid path, a node was added to the tree containing the pixel location, while pixels with a value of 1 represented the walls of the maze and were not added as nodes. For each node, the child nodes were then generated by finding the neighbouring pixels with value 0.

A.2. Node Reduction

While the above naïve method worked well for the smallest maze size, a very large number of nodes were generated using this approach as every white pixel in the maze was represented as a node in the tree. In order to provide a more suitable representation for larger mazes, only points in the maze which required a decision between several different paths were represented as nodes, with adjoining pixels being represented as edges or paths between nodes in the tree.

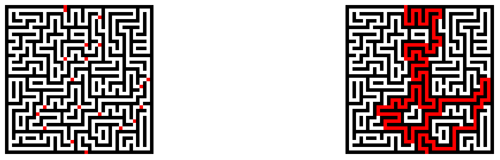
(a) Decision points (b) Full path

Fig. 1: Optimization to reduce the number of nodes stored in the tree structure.

This optimization is depicted in Fig. 1 above, showing the sparse nodes representing decision points in the path in comparison to the full path. This allowed for a significant reduction in the number of nodes constructed allowing for a reduction in the associated memory and speed requirements.

A.3. Dynamic Tree Structure

However, while the reduction in tree nodes allowed for the algorithms to effectively solve the small-medium mazes, larger mazes still posed a problem in terms of speed and memory requirements. Therefore, in order to further optimize the tree structure, the paths between the parent and child nodes were found and built dynamically during the search. This prevented the entire tree structure from being constructed up-front and therefore prevented redundant paths from being discovered. This allowed for a faster and more memory-efficient search.

Hence optimizations of the underlying tree structure formed a crucial part of the investigation in order to allows for larger mazes to be effectively traversed.

B. *Ant System (AS)*

Several design choices and hyperparameters required careful tuning to ensure ant system was adapted to the longest path problem. Notably, different maze sizes required significantly different strategies regarding hyperparameter and heuristic choice.

B.1. Pheromone Update

The first modification to the Ant System algorithm was the adaptation of the pheromone update step to the the longest path problem. For the shortest path problem, the pheromone added by each individual ant is inversely proportional to the length of the path segment. Therefore, for the longest path problem, this is modified to the additional pheromone being proportional to the length of the path segment, in order to encourage longer path segments to be chosen by the ant population.

Three variants of Ant System exist in terms of the pheromone update strategy, known as Ant-Cycle, Ant-Density and Ant-Quantity ant system. The definitions of each are included below, where Q represents a positive constant.

a) Ant-Cycle AS

Ant-Cycle AS is dependent on a chosen fitness function, such as the entire length of the path found by an individual ant. In this case, the pheromone update is proportional to the fitness as higher fitness indicates a longer overall path found. Therefore for longer paths, more pheromone is added to the path segment, encouraging other ants in the population to select the longer segment.

$$\triangle\tau_{i,j}^{k}(t) = \begin{cases} Q * f(x^k(t)) & if\ (i,j) \in x^k(t) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

b) Ant-Density AS

Ant-density AS relies only on the number of ants that traverse a given path segment, as every traversal will result in the addition of Q pheromone. Therefore the pheromone updated for Ant-Density AS is independent of the path segment length.

$$\triangle\tau_{i,j}^{k}(t) = \begin{cases} Q & if\ (i,j) \in x^k(t) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

c) Ant-Quantity AS

Ant-Quantity AS relates the amount of pheromone added to the length of the specific path segment. For the longest path problem, the update is proportional to the length of the path to encourage longer paths to be taken.

$$\triangle\tau_{i,j}^{k}(t) = \begin{cases} Q * d_{i,j} & if\ (i,j) \in x^k(t) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

For the purposes of this study, Ant-Quantity AS was implemented due to ensure the length of the path was taken into account.

B.2. Heuristic Choice

Since the transition probability calculated to determine the next movement of each ant depends on the heuristic information of a given move, heuristic choice plays a key role in the correct functioning of the ant system algorithm. An important consideration regarding the heuristic choice for ant system is that the heuristic information must be > 1. If the heuristic information if less than 1, when it is raised to a power in the transition probability equation, it will diminish instead of increasing its emphasis with an increase in β.

In order to encourage longer paths to be chosen, the euclidean distance was calculated from the candidate location to the end of the maze. This was then normalized by the distance from the entrance to the exit of the maze. Several different variants were considered, including $1 + d/d_{max}$ to represent the longest path heuristic or $2 - d/d_{max}$ to represent a shortest path heuristic. However, the calculated euclidean distance d with no modifications was used as the primary heuristic.

B.3. Cycle Removal

In order to ensure that the longest simple path is found, all cycles are removed from the paths found by the ants in the population. Cycles arise when an ant reaches a dead end and needs to backtrack to a previous node which contains another possible direction or when an ant traverses an existing loop in the maze.

In order to remove the loops, matching nodes for each node in the final path found were identified and any nodes inbetween the first occurrence and subsequent occurrences were removed. This ensures that only simple paths are output by the algorithm.

B.4. Hyperparameter Tuning

Several key hyperparameters required tuning for the ant system algorithm. These hyperparameters also had a crucial influence when solving larger mazes which presented significantly different challenges to the smaller mazes.

a) Pheromone-Heuristic Intensity (α, β)

The influence of the pheromone levels and heuristic information respectively are controlled by the α and β parameters. The optimal balance of parameters is highly dependent on maze size. For smaller mazes, paths are more easily found within a practicable time and therefore the heuristic information can be exploited to encourage the algorithm to find paths further away from the exit and thus longer paths through the maze. Therefore the heuristic information can be emphasized with larger values of β.

In contrast, for larger mazes, there exists the dual goal of making sure a path is found as well as trying to find the longest path. Therefore the heuristic information is emphasized less and pheromone levels are relied upon to help guide ants to the exit after previous ants have located a successful path. This can be accomplished through the use of larger values of α. This balance indicates the ability of ant system to be fine-tuned and customised to a given problem.

TABLE I: Selected Hyperparameters for Different Maze Sizes

Maze Category	**Size**	α	β
Small	43×43	1	5
Small-Medium	203×203	1	5
Medium	2003×2003	3	1

b) Population size K

The ant population size determines the amount of exploration performed, with more ants allowing for more exploration of the paths in the maze. However this comes at a cost of a longer run-time, presenting a trade-off at each maze size. Thus for smaller maze sizes, a population of **10** was selected to allow for more exploration. It was found that populations larger than this did not bring any additional benefit for these maze sizes. For the larger maze sizes, a smaller population of **5** was selected to ensure that the algorithm could still run within a feasible amount of time.

c) Evaporation rate ρ

The evaporation rate serves to prevent over-saturation of the edges in the graph, and therefore prevents the domination of a single path. However, an evaporation rate that is too high causes the pheromone signals left by the ants to be too weak. Therefore an evaporation rate of $\rho = 0.3$ was chosen.

In contrast to the balance between pheromone and heuristic information, and the chosen ant population size, the evaporation rate only impacts the ant system algorithm after the first iteration has occurred. Therefore, for larger maze sizes where it can be difficult to ensure a single iteration in a feasible amount of time, the evaporation rate has little influence.

d) Elitism parameter n_e

The parameter n_e controls the strength of the elitest force by weighting how much the additional pheromone is laid down on the best path found. The selection of the elitism parameter involves a trade-off between exploration and exploitation. An n_e value that is too low results in a lack of exploitation as known information is not utilised. Conversely, an n_e value that is too high causes over-exploitation and therefore limits exploration.

A value of $n_e = \mathbf{3}$ was chosen to provide a balance between exploration and exploitation. This was increased to $n_e = \mathbf{5}$ for larger maze sizes in order to help guide ants towards the maze exit within a feasible amount of time. Similarly to the evaporation rate, the elitism parameter only has an effect after the first iteration has been successfully performed.

3) Number of iterations t_{max}

The number of iterations is a core part of the ant system algorithm as the pheromone levels are updated after each iteration, therefore allowing the pheromone to build up over the course of multiple iterations.

Too few iterations results in the ant population being unable to learn from the pheromone signals as pheromone has not yet been built up. However, too many iterations can cause a run-time that is too long, especially in the case of larger mazes where a single iteration may take a substantial amount of time.

Accordingly, for smaller maze sizes, the algorithm was performed for **10** iterations, while for larger maze sizes only **1-2** iterations were performed in order to ensure a feasible run-time. Table II below summarises the chosen hyperparameters for the ant system algorithm.

TABLE II: Selected Hyperparameters for Different Maze Sizes

Maze Category	**Population Size K**	ρ	n_e	t_{max}
Small	10	0.3	3	10
Small-Medium	10	0.3	3	10
Medium	5	0.3	5	1-2

C. Beam Search

The performance of beam search is extremely dependent on the choice of beam width and heuristic information.

C.1. Beam Width

Beam width refers to the quantity of nodes that are stored and whose paths will be further expanded. A beam width that is too large increases the run-time of the algorithm since many more paths are explored. This is particularly disadvantageous for large mazes. In addition, if the beam width is too large, too much memory capacity may be required.

Conversely, a beam width that is too small prunes too many paths and may not find any promising solutions. However, smaller beam widths allow for a faster search. Thus, the most suitable beamwidth was dependent on the size of the maze being tested and was determined experimentally. The final beam widths chosen for each maze size are shown in Table III below.

TABLE III: Selected Beam Widths for Different Maze Sizes

Maze Category	**Size**	**Beam Width**
Small	43×43	30
Small-Medium	203×203	1500
Medium	2003×2003	6000

C.2. Heuristic Choice

Heuristic choice plays a pivotal role in the beam search algorithm as potential paths are pruned based on the heuristic chosen. A variety of heuristics were considered and tested during the investigation. A core difficulty was uncovered regarding the balance between encouraging the algorithm to find a long path and ensuring that the search could still locate the maze exit.

In order to balance this, the heuristic was adapted throughout the search. For maze depths less than a quarter of the maximum depth of the maze, a short path heuristic was utilised to ensure that all valid paths leading to the exit were not pruned. Subsequently, for maze depths less than half of the maximum depth of the maze, a long path heuristic was utilised to encourage exploration of longer paths in the maze. For the remaining depths, the short heuristic strategy was employed to ensure the exit of the maze could be reached. This process is summarised in the following equation,

$$\eta_j = \begin{cases} 1/d_j & \text{if } \mathit{current\ depth} < \mathit{max\ depth}/4 \\ d_j & \text{if } \mathit{current\ depth} < \mathit{max\ depth}/2 \\ 1/d_j & \text{otherwise} \end{cases} \tag{8}$$

where η represents the heuristic information for node j and d_j is the distance between node j and the exit of the maze. Unlike the heuristic for ant system, since the heuristic used in beam search is used to order the candidate nodes, there is no requirement for the heuristic information to be greater than 1 and thus the inverse of the distance is used for the shortest path heuristic.

The depths at which the heuristics are switched between the short and long path heuristics can be seen as a hyperparameter and were determined experimentally. While different mazes performed better when switching at different depths, the parameters quoted above worked best for most mazes.

D. *Evaluation*

Each algorithm was evaluated on each of the ten mazes, ranging from small, small-medium, medium and large maze sizes. Each maze was read in from a BMP file. In order to display the final path found, the pixels along the longest path found were coloured red in the generated maze image. The output image was also stored as a BMP file, showing the marked path in red.

The performance of the algorithms was compared in terms of both the longest path found for each maze, as well as the efficiency of the search. All algorithms were implemented in Python 3 and all experiments were run in a Google Colab environment.

IV. Results

The longest path through each maze obtained for the ant system and beam search algorithms respectively are shown in Table IV. It should be noted that the longest path found is reported in number of pixels rather than the number of nodes to provide a fair comparison.

TABLE IV: Longest path found by the ant system and beam search algorithms respectively, for each test maze

Maze	Size	Algorithm	Longest Path
Small 1	43×43	Ant System	249
		Beam Search	249
Small 2	43×43	Ant System	77
		Beam Search	77
Small-Medium 1	203×203	Ant System	6021
		Beam Search	1361
Small-Medium 2	203×203	Ant System	5241
		Beam Search	1485
Small-Medium 3	203×203	Ant System	2433
		Beam Search	2433
Small-Medium 4	203×203	Ant System	269
		Beam Search	269
Medium 1	2003×2003	Ant System	73735
		Beam Search	9995
Medium 2	2003×2003	Ant System	73277
		Beam Search	7397
Medium 3	2003×2003	Ant System	N/A
		Beam Search	14379
Large 1	10001×10001	Ant System	N/A
		Beam Search	N/A

It can be observed that ant system far outperforms beam search in terms of the length of the path found for almost all mazes except for the smallest maze size.

A. *Discussion*

The results obtained by each algorithm at the differing maze sizes will be discussed in terms of multiple factors including quality of solution found, efficiency, completeness and reliability.

Small Mazes

It was found that ant system and beam search performed equivalently for the smallest maze size for both *Small1* and *Small2*. Given the finite size of the small mazes, almost all paths were explored by the two algorithms. Therefore the mazes were small enough for beam search to find the longest path given a reasonable beam width.

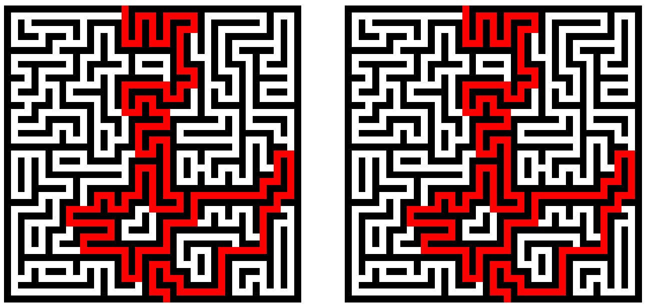
(a) Beam search (b) Ant system

Fig. 2: Longest path found by beam search and ant system respectively for the smallest maze.

The equivalence between the two approaches is depicted in Fig. 2, showing the longest path found through the *Small1* maze for each algorithm. In addition, both ant system and beam search found the same path through *Small2* as there was only a single valid path through the maze.

Small-Medium Mazes

In contrast to the small mazes, the benefits of the ant system meta-heuristic can be clearly observed for the small-medium mazes. The paths found by ant system are significantly longer than those found by beam search. This significant advantage can be observed in Figure 3 showing the much longer path found by ant system.

Since the maze size is still appropriately sized for ant system to run in a feasible amount of time, the hyperparameters could be tuned optimally to push the algorithm to find longer paths. Therefore ant system provides very few drawbacks at this maze size.

On the other hand, beam search can no longer explores the same proportion of paths as for the smallest maze. Therefore due to the restrictions imposed by the switched short and long heuristics, beam search can only perform a limited amount of exploration of longer paths while also successfully reaching the endpoint.

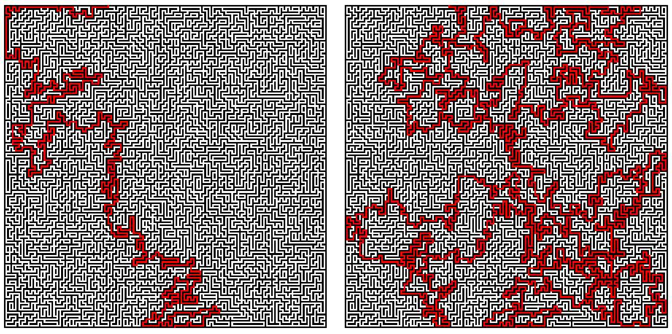
(a) Beam search (b) Ant system

Fig. 3: Longest path found by beam search and ant system respectively for the small-medium maze.

However, beam search and ant system perform equivalently for *Small-Medium3* and *Small-Medium4* as only a single path is found through the maze. It can be noted that beam search is deterministic for fixed hyperparameters and multiple simulations produce the same path through the maze.

Medium Mazes

A critical challenge for the medium mazes constituted the efficiency of the search, particularly with regard to the ant system algorithm. Several strategies were employed to attempt to ensure that ant system could be performed within a reasonable amount of time, such as utilising a larger population of ants and allowing the next iteration to occur without waiting for all ants in the population to reach the end of the maze. While this helped to improve run-time, a more long-term solution is warranted.

With respect to the final solutions found, the ant system approach yielded significantly longer paths than beam search. However, this was at the cost of a significantly slower run-time as well as unpredictable results due to the stochasticity present in the ant system algorithm. While beam search provided a faster search, the heuristic had to be modified to spend a larger fraction searching for shorter paths in order to successfully find the exit.

In addition, beam search provided a distinct advantage for the *Medium3* maze as beam search was able to find the solution rapidly while ant system took too long to find a valid solution. It was observed that beam search continued to find the same path for different beam widths and therefore only one valid path was present. This suggests that beam search provides an advantage when there is only a single path through the maze, as the systematic nature of beam search is able to locate this path.

Large Mazes

Due to the inefficiency of the ant system algorithm and the representation of nodes, the largest maze could not be solved within a reasonable amount of time. Since too many nodes are created for the largest size, memory capacity is exceeded. This could be overcome by finding the nodes dynamically instead of only the paths between the nodes. However, given the long run-time of the current ant system implementation on the medium mazes, the ant system algorithm would require significant optimization to run on a maze of this size. Therefore both the underlying tree structure and ant system algorithm itself requires further optimization.

Further Improvements

Several key improvements remain for further exploration in order to increase the efficacy of each algorithm. Firstly, the ant system algorithm could be further optimized in terms of both the implementation of the algorithm and associated hyperparameters in order to ensure a feasible run-time on larger mazes. Although several different strategies in terms of the implementation and chosen hyperparameters were tested, further investigation is crucial for a more efficient search.

In terms of the heuristic choice, more intelligent heuristics could be designed for both ant system and beam search by introducing additional information or different distance measures. This is pivotal in order to effectively guide the search at different maze sizes. Moreover, backtracking could be introduced into the beam search algorithm to prevent longer paths from being completely pruned. However, this may introduce much longer run-times for larger mazes.

Lastly, the tree structure used to represent the maze could be further optimized to allow for the largest maze to be traversed in a feasible length of time.

V. Conclusion

Ant colony optimization approaches model the natural behaviour and communication structure of ant populations to solve combinatorial, graph-based problems. The more classic beam search employs a breadth first search approach in an incomplete manner by retaining only the most promising nodes. In this study, these contrasting methods were applied to the NP-complete problem of finding the longest simple path through a given maze.

It has been observed that ant system is more readily applied to the longest path problem due to its versatility and large number of hyperparameters. However, a significant drawback of the ant system approach is the long run-time required to solve larger mazes. In contrast, for larger maze sizes, beam search provides a more efficient approach at the cost of the length of the path found. Additionally, beam search provides more reliable results as it is deterministic for fixed hyperparameter values, while the stochasticity of ant system introduces unpredictability into the length of the run-time and the solution found.

Most critically, the underlying representation of the maze is found to be of critical importance, as inefficient storage of the maze prohibits effective application of either algorithm. This problem was countered through two strategies, comprising a reduction in the number of nodes stored, and the dynamic construction of the paths between the nodes at run-time. However, further optimizations of the underlying structure are required in order to feasibly solve the largest maze size.

Therefore both ant system and beam search provide a promising approach to the challenging problem of finding the longest simple path through a maze. While this investigation provides a baseline exploration into the effects of the respective hyperparameters and possible heuristics, further investigation into the optimal heuristic and hyperparameter choice is warranted in order to maximise the efficacy of these techniques.

References

[1] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996.

[2] Christopher Makoto Wilt, Jordan Tyler Thayer, and Wheeler Ruml. A comparison of greedy search algorithms. In *Third Annual Symposium on Combinatorial Search*, 2010.