Creating Varied Terrain-Considerate Road Networks on Heightmaps with Directed Alternating Physarum Agents

Jeroen Hijzelendoorn
Game and Media Technology Master's Thesis
Utrecht University
6262279
j.m.hijzelendoorn@students.uu.nl

***Abstract*—This thesis presents a novel pathfinding technique for creating terrain-considerate walking-path networks over heightmap terrain called Directed Alternating Physarum Agents (DAPA). DAPA has similar dynamics to slime mould networks, but has more control over the network generated, can connect any two nodes of choice and can realistically navigate heightmap-generated terrain. Unlike most pathfinding techniques, DAPA paths consists of line segments unrestrained by the heightmap grid, making for smooth looking paths. Furthermore, same-destination paths may join together due to shared exploration resources.**

***Index Terms*—road network, pathfinding, slime mould, multi-agent, emergent behaviour, heightmap**

I. Introduction

Open world videogames have seen a steady rise in popularity over the years. These games offer a lot of freedom to players by allowing them to go to virtually anywhere in the game world from the very start. The appeal of these games is the opportunity to explore the world's regions in any order the player desires. To facilitate this feeling of freedom and exploration, these game worlds are often very large and typically feature natural landscapes with i.e. forests, mountains and settlements placed at large intervals. The settlements are typically connected by manually-designed road networks.

This thesis presents a procedural pathfinding method for these kinds of roads. Procedural road network generation is an uncommon sight in videogames, while research tends to focus on urban street-planning applications(1). For the purposes of connecting points in virtual open worlds, manual designs or regular pathfinding techniques are typically used. However, these techniques either ignore or abstract verticality to some extent, are incapable of path interactions such as intersections, are focused on creating networks from scratch rather than connecting existing points, or restrain paths to a grid representation of the world, lowering path resolution.

Directed Alternating Physarum Agents (DAPA) connects multiple points of interest over large distances while navigating natural landscapes. The paths attempt to optimize length and terrain traversability and may merge with other paths if they share a common destination.

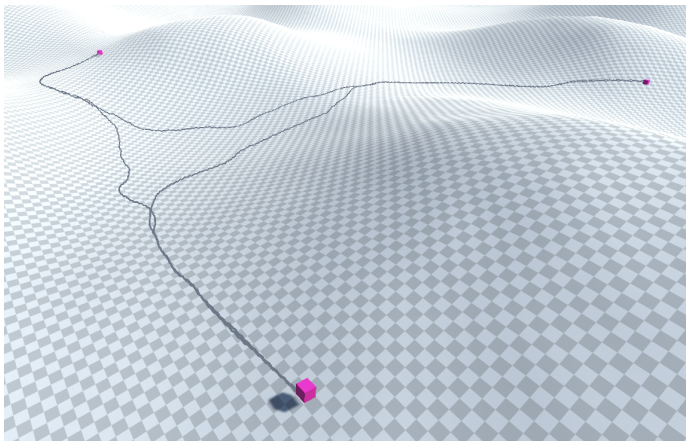

Fig. 1: Example path network
Three point of interest are connected by different paths, joining up as they meet each other while moving around the hills.

Pathfinding is based on heightmap[1] data, but resulting paths are not restricted to discrete grid points, rather being comprised of short line segments with floating point start and end coordinates.

A. Requirements

To resemble real walking paths, the technique aims to fulfill three primary and three secondary requirements.

Primary

- Heightmap Considerate: paths need to avoid hard-to-navigate terrain like steep mountain sides if a serviceable detour is possible. That is, it must prefer paths that minimize both horizontal and vertical distance travelled. Furthermore, gradual slopes must be preferred over sud-

[1]Heightmaps are 2D textures which can be translated to 3D terrain. In the texture, the brightness of each pixel indicates the height of the corresponding grid coordinate of the terrain. Heightmaps are a popular means of creating 3D terrain, as they can be created with relative ease compared to other ways like 3D sculpting.

den jumps in height. The technique will need to consider the 2.5D[2] heightmap data for this purpose.
- Smooth: Real paths consist of a long trail which bends around obstacles. Discrete 4- or 8-neighbour path representations cannot capture this; a path with 360° of freedom is required.
- Large Scale: The technique must be able to create paths over large distances without significant performance drawbacks. As a point of reference, this must be possible when creating a network between 3 points spaced 1600m apart on a 2048x2048-resolution heightmap, using no more than 2GB of memory and taking at most 5 minutes.

Secondary
- Joining: paths should join into one if they run close to one another, rather than having multiple separate paths side by side.
- Winding: paths must ascend steep slopes in a winding pattern, as seen in real-life mountain scaling paths.
- Imperfect: path need to be short, but not necessarily perfect, as this is also not the case in real life. The final path should be no more than 110% of the optimal path's length when simulating paths over heightmap terrain.

Navigating nonuniform cost terrain like heightmaps and creating smooth paths is already individually possible with modern techniques. However, a combination of the two is not common.
Here, we differentiate between smooth and smoothed paths. Smoothed paths are based on discrete ones and use these discrete points to create smooth curves. The discrete basis means the final path doesn't consider terrain features as much as purely smooth paths. However, this could be considered as a trade-off, trading quality for an increase in imperfectness. The creation of smooth paths on heightmap terrain together with its application on a large scale is the goal this thesis.

We will create these paths by answering 3 research questions (RQ).
- RQ1: Which existing techniques serve as a suitable baseline for this project?
- RQ2: What modifications and extensions are required for a chosen baseline to solve our pathfinding problem?
- RQ3: How effective is the new technique at attaining the primary and secondary requirements as opposed to the baseline, and how do we measure this?

II. Related Work

As mentioned, many procedural road network generation algorithms mainly focus on applications for creating urban layouts, a very different use case from the one mentioned above. As such, these specific works will not be featured here. Below is a summation of pathfinding and network creation algorithms with applications related to the thesis requirements. These techniques double as potential candidates for a baseline for the thesis.

A. A*

First off, the 'golden standard' of pathfinding in games, A* (2), a graph search algorithm which can use a given heuristic to adjust its exploration behaviour. This generally makes it a more time-efficient algorithm than other graph search techniques. Being graph-based, it lends itself very well to the structure of videogames, whose environments are often times represented as grids that can easily be converted to graphs. Furthermore, the option to add a custom heuristic makes A* adaptable to many different tasks.

A* has plenty of variants that add features and optimizations to the method. Iterative-Deepening A* (IDA*) or Simplified Memory Bounded A* (SMA*) produce identical results to A* but have reduced memory consumption, while any-angle path planning variants permit diagonal paths outside of the restrictive 8-neighbour 45° grid paths that regular A* must abide by.
Other variants of interest include incentivised joint graph traversal between paths (3) (as opposed to cooperate pathfinding), reducing node expansion using bidirectional search or even more so using preprocessing techniques (4).

B. Navmesh

The second pathfinding approach frequently used in video game environments are navigation meshes (5), more commonly referred to as navmeshes. A navmesh is a collection of convex polygons which covers all walkable space in an environment. Movement within a convex polygon is trivial, being able to walk in a straight line between any two points within it. A graph is constructed containing all polygons which mirrors the layout in the scene. Using a graph search algorithm, a sequence of any-angle polygon traversals can be determined to get to the destination. With this approach it is possible to walk efficient paths without having to adhere to grid structures.

C. Potential Field

Potential fields can create continuous paths by having obstacles exude negative force while the goal exudes positive ones. A path is generated by following the strongest positive force. This is applicable in both discrete and continuous worlds. However, as heightmaps contain no explicit obstacles, this technique is not applicable to nonuniform cost terrain.

D. Ant Colony Optimization

Ant colony optimization (6) (ACO) is also an interesting option to consider, as it allows us to choose how much time we want to spend optimizing the resulting path. Similarly, we can choose a memory maximum by limiting how many agents are used in its simulation. ACO was originally made to solve the travelling salesman problem but is also sees application as for graph search in general, which extends to pathfinding

[2]Heightmap-generated terrain is not truly 3D. It produces exactly one terrain point per grid coordinate and is thus not able to represent all 3D features such as overhangs and bridges. It is therefore sometimes referred to as 2.5D.

over grids. The algorithm imitates real-life ants leaving their colony in search for food. Digital ants move from the source node throughout the graph while leaving behind pheromones. Other ants detect these pheromones and follow them with a probability based on the pheromone's strength. Pheromone strength, in turn, is based on the path length. This causes the shortest paths to be the most likely to be reinforced again and again. ACO has no definite termination condition, rather improving over time until it stagnates at a (locally) optimal solution. ACO has been applied to heightmaps before (7), and can even be used for continuous spaces (8).

E. Slime Mould

Next, we cover slime moulds (9), a multi-agent algorithm. This pathing technique is based on the behaviour of a real-life slime mould, physarum polycephalum, which can produce rather efficient networks in its search for food. Conceptually, it has some parallels to ACO, with agents leaving and following pheromone trails. A difference is that slime mould algorithms do not feature a specific start and goal node, rather multiple points of interest (POI) in general. These POIs propagate pheromones, attracting agents to approach it. The algorithm works by spreading large number of agents around the scene, after which they move around randomly or follow the strongest pheromones they can find. The pheromones are stored in a grid and are diffused and spread out after each update, causing agents to detect trails from larger distances and merging close-together paths. Agent positions are stored with floating point coordinates. However, each agents still occupies exactly one grid point or 'cell' and are unable to pass through cells occupied by other agents. As agents either follow one another or approach POIs, the algorithm can create efficient path networks connecting multiple POIs. Agents could sample from the heightmap in the same manner as they do from the pheromone map, making for an easy extension to nonuniform cost terrain traversal. Furthermore, just like ACO, we can adjust memory consumption when needed by reducing the number of agents used.

F. Gathering Technique

In a series of videos of Pezzza's Work (10), an ant simulation method for finding smooth paths from a colony to pockets of food scattered across a discrete domain was presented. As mentioned by its creator, this hobby project has no scientific basis. The videos call the method 'ant simulation', but to prevent confusion with ACO, this method will be referred to as the 'gathering technique' for future reference. The simulation functions like a modified slime mould technique. It has similar dynamics with agents leaving pheromones and using sensors to detect them to determine their behaviour. Unlike slime moulds, however, agents all originate from the same start node, and attempt to return there after finding a food source instead of another POI. Furthermore, only agents leave pheromones. These pheromones are not diffused like in slime mould simulations. Lastly, the gathering technique makes use of two types of pheromones to distinguish agents moving to and from the food sources. This way, searching agents can follow trails of agents who are returning after having found a food source and vice versa.

G. Authoring Hierarchical Road Networks

This paper (11) describes the creation of road networks over 3D terrain consisting of different road types. Firstly, highways are generated between large settlements, then primary roads between smaller ones and finally secondary roads between the smallest settlements. The network is constructed by first generating a graph of possible connections, then culling redundant roads and finally joining adjacent roads based on the Fréchet distance between them. The paths are generated according to earlier work (12), using a n-neighbour A* system which additionally can build bridges over water and tunnels through mountains, taking into account road curvature restrictions. After a path consisting of grid points is found, a smooth path is constructed from these points using clothoïd curves.
I only discovered this paper in the last week before the proposal date, meaning that it was too little too late to adjust my plans to their contents. However, as it is highly relevant work related to the goal of this thesis I decided to still mention it here.

III. Analysis

To pick a suitable baseline technique we need to compare the techniques mentioned above. Table I gives an overview of the main characteristics of each technique, while below any miscellaneous features and/or problems are mentioned. Finally, the most suitable one is chosen.

*A. A**

A* is one of the few techniques that is compatible with heightmaps out of the box. Due to A*'s admissibility property, it and its variants are the only techniques on this list that can already satisfy the Winding condition with the use of a maximum inclination and/or steepness aversion heuristic. However, A* has memory issues. It saves all explored nodes, and as such has a exponential memory complexity. Such a characteristic does not meet the Large Scale requirement. Furthermore, the discrete nature of the resulting paths does not conform to our smooth path requirement. All mentioned variants suffer one or both of these issues.

Iterative-deepening A* only has a polynomial time complexity and Simplified Memory Bounded A* can set a memory limit for its search, meeting the Large Scale requirement where A* could not. However, these techniques still do not meet the primary Smooth requirement.

Many of the any-angle techniques are not compatible with nonuniform cost terrains (like heightmaps). Even compatible techniques like Field D* (13) still carry the same memory issues as A*. Furthermore, while an any-angle approach is a step up from discrete paths, it is still not suitable for creating smooth paths as they are unable to create curves.

Incentivized joint graph traversal adds a form of realism to the resulting paths by having two paths share parts of their trail. However, the caviat of this technique is that paths need to be explicitly chosen to share paths, which makes it unfit for creating multiple paths that connect or overlap in a natural looking way. Bidirectional search A* still has the discrete path issue and is thus not useable. As mentioned before, the method is to be designed for videogames where environments are generated upon play. In this case, preprocessing and regular processing would always execute directly in sequence, defeating the point of preprocessing.

B. Navmesh

Navmeshes aren't suited for nonuniform cost environments, as movement cost can only differ between polygons, not within them. Theoretically it is still possible to apply them to heightmaps, with each polygon representing a grid coordinate. However, this would defeat the purpose of the polygonal abstraction of the terrain and would behave exactly like grid-constrained any-angle techniques, along with their limits.

C. Potential Field

As the technique is dependant on following the positive potential of the goal, long-distance paths would require a very strong potential. This would cover a much larger area than smaller simulations, increasing memory demands exponentially. Furthermore, potential fields can get stuck in local minima when positive and negative potentials cancel each other out. With a positive field spanning over a larger area, the number of local minima will surely increase too. For these reasons I assume the potential field technique to miss the Large Scale requirement.

D. Ant Colony Optimization

Together with slime mould and the gathering technique, ACO is a path optimization technique that relies on randomness to a certain degree. This means that in a complex environment, an optimal solution will likely not be found within reasonable time, but an approximately optimal solution might. With this characteristic, these techniques attain the Imperfect requirement. ACO's performance is strongly dependant on the size of the environment, as it can drastically increase the search space. However, the effect can be reduced by simply limiting the search space by laying initial pheromones between the start and goal state (14). By implementing more such effective extensions for limiting the search space, ACO, slime mould and the gathering technique could possibly increase their convergence speed to such an extent that they satisfy the Large Scale requirement.
Discrete ACO has the same issue as A*, where bends can only be created as a postprocessing effect. Continuous ACO is only able to construct continuous (thus smooth) path solutions based on a continuous domain. This would require the discrete heightmap to be converted to a continuous representation to be used.

E. Slime Mould

Slime mould looks promising for the purposes of smooth path generation over heightmaps. With agents reading pheromones from a grid, reading heightmap values from a grid would be a feasible extension. As the only technique in this list that generates a pathing network at once, it is also the only one meeting the Joining requirement. However, the technique still lacks control, as it can only connect all POIs to one another into one big network. In a videogame environment, this is not always desirable. Additionally, slime mould simulation does not actually produce an actual path; rather a collection of pheromones that agents can follow while freely moving. A path creation extension based on agent movement would create smooth paths, however.

F. Gathering Technique

Just like slime mould, extending the gathering technique to consider the heightmap should be possible. However, it shares its issue in not producing an actual path. Here too, a path creation extension would solve this problem.

G. Authoring Hierarchical Road Networks

As mentioned before, creating smooth paths from discrete representations is differentiated from 'pure' smoothness, meaning the technique does not quite meet the 'Smooth' requirement. It's road joining post processing pass earns it the 'Joining' requirement. Next, by down-sampling the 3D environment, the technique is able to generate roads over large terrains. This does impact path quality somewhat, but does not result in major path inefficiencies, meeting the 'Large Scale' requirement. Lastly, the underlying path generation algorithm uses stochastic point sampling to boost performance when generating tunnels or bridges. As a consequence, these road path have an element of randomness to them. However, as it is not apply to the whole path, it does not meet the 'Imperfect' requirement.

H. Baseline choice

As seen in table I, there is no technique present that can meet all the requirements based on a discrete world representation as input.
(11) is the most suitable technique so far. However, this technique will not be considered as a baseline option as I only discovered it in the last days of the proposal stage, leaving me too little time to completely change my thesis plans.
Therefore, any method chosen would need to be augmented tremendously, or a completely new approach will have to be created. The former was chosen, with the gathering technique as the chosen baseline. This, because it is already able to create smooth paths from discrete terrain and an extension to heightmaps seems very feasible, in contrast to creating smooth terrain-considerate paths from discrete ones, as would be the alternative for the other prime candidates, A*(variations) and ACO. Slime mould was also highly considered, but adjusting it for increased path creation control would likely result in a

TABLE I: Algorithm comparison

Algorithm	World representation	Heightmap Considerate	Smooth	Large Scale	Joining	Winding	Imperfect
A*	Discrete	✓	×	×	×	✓	×
IDA*/SMA*	Discrete	✓	×	✓	×	✓	×
Field D*	Discrete	✓	×	×	×	✓	×
Navmesh	Polygonal	×	×	✓	×	×	×
Potential fields	Continuous	×	✓	×	×	×	×
ACO	Discrete	✓	×	✓*	×	×	✓
Continuous ACO	Continuous	✓	✓	✓*	×	×	✓
Slime mould	Discrete	✓*	✓*	✓*	✓	×	✓
Gathering technique	Discrete	✓*	✓*	✓*	×	×	✓
Authoring Hierarchical Road Networks	Discrete	✓	Smoothed (post-processing)	✓	✓	✓	×

Checkmarks indicate that the base algorithm already meets the corresponding requirement, crosses mean they do not. An asterisk indicates that a requirement can theoretically be met by extending the algorithm.

method very similar to the gathering technique, making it a redundant option.

IV. Method

Below, the baseline is explained in more detail, the minimal comparable version v0.5 is presented, alongside its extensions and further extensions that will be added over the course of the thesis.

A. Baseline

As the gathering technique is essentially a slime mould modification, the new technique will be called Directed Alternating Physarum[3] Agents (DAPA). This new technique will direct agents to automatically-determined goal points. The agents arrive at the goal using slime mould-like pheromone mechanics, after which they alternate their navigation to instead search for their starting point. A video of Sebastian Lague (15) also covers (a modification of) the gathering technique, but in more detail than Pezzza's Work (10). Therefore I based the implementation of this slightly adjusted version which uses slime mould's sensor model (9) for the agents, see fig. 2.

The baseline already contains the following features. It works on uniform cost terrain with walkable surfaces and impassable obstacles. One or more ant colonies and food pieces are placed on the terrain. At the start of the simulation, ant agents oriented in all directions are placed on the colonies. Each simulation iteration the agents move forward, but can bend to the left or right depending on random deviation and their sensor readings. Each agent has three sensors in front of it which measure the pheromone strength within their region, see figure 2. The strongest sensing sensor dictates what direction an ant wants to head in. Over time, the walking direction of the ant is adjusted to that of the strongest sensing sensor, making for a smooth motion. Unlike slime mould, ant agents do not occupy physical space and as such do not take inter-agent collision into consideration, only colliding with unwalkable terrain. As mentioned before, agents can leave two types of markers. Once food is found or brought back to the colony, the ant will start leaving 'to food' or 'to home' markers respectively. Finally, all pheromones lose freshness over time, meaning that if a food source is depleted, the trail towards it will slowly vanish as well. Because ants prefer fresher pheromones, these vanishing trails will not be followed in vain if any other active trail is still in use in the vicinity. In theory, shorter paths are preferred, as it means a path has less time to degrade while ants make trips over them. Furthermore, as agents walk over them, paths are automatically shortened, as random movement deviation creates shorter and preferred versions of the path over time. However, during the implementation of the baseline, this behaviour did not emerge. I inquired Pezzza's Work about this and came to understand he used the same solution I did to remedy these shortcomings myself (see: *Distance counters*, in 'Implemented extensions'). However, he also offered insight on his agent's sensor model, which helped in attaining this behaviour. More on this in *Probabilistic sensor* in 'Thesis extensions'.

Terminology: The parallel to ant-like simulation becomes inaccurate for DAPA in the context of path generation in videogame environments. Therefore, I will be referencing to certain elements more appropriately as seen in table II.

TABLE II: Terminology change

Old term	New term
Ant	Agent
Colony	Node
'to home' pheromone	'to start' pheromone
'to food' pheromone	'to goal' pheromone

For ease and accuracy of reference, terminology used for the baseline will be changed to the above.

[3]Physarum being the mould species the slime mould technique is based upon.

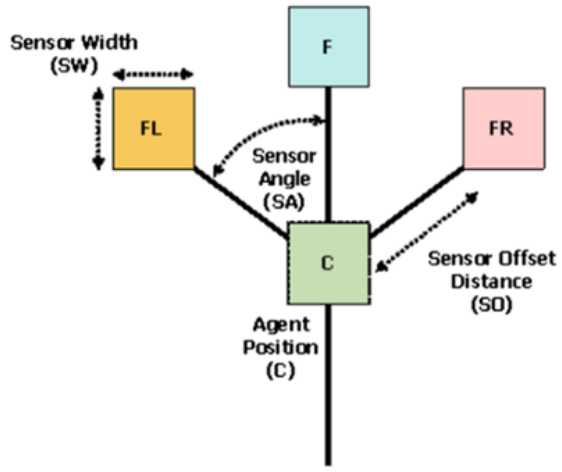

Fig. 2: Agent sensor model
Agent (C) with three sensor areas in front of it: front left (FL), front (F) and front right (FR). Sensors are set at an angle SA at a distance SO from the agent.

B. DAPA v0.5 and v1

As mentioned in the introduction, the project has multiple pathing requirements. The baseline is only able to solve 1 of the requirements by itself. However, it has the potential to fit all of them with multiple extensions of various complexity. The completely extended version - the result of this thesis - will be called DAPA v1. For future improvement measurements, we would like to compare v1 to the baseline to see its performance improvement. However, as the baseline is not meant for generating specific paths, it is difficult to apply it for that purpose. As such, I introduce DAPA v0.5 as a minimal functioning version for the purpose of generating specific paths with this method, meant to compare against the fully functional v1.

C. Minimal version extensions

Below, a list of extensions is given. A distinction will be made between implemented extensions and those that will need to be developed during the thesis. First, the implemented features are mentioned.

1) Directed pathing: This extension was added such that the baseline became fit to generate specific paths on demand. To generate a path between a chosen start and end position, a node is placed on the start position and one food piece was placed on the goal position. Multiple agents are released from the node and directed to only search for the specific goal food piece. Once reached, the agents search for a way home as usual, though without picking up the food. This way, agents will keep pathing between the two points until the simulation is stopped as the food will never move or disappear.

2) Bidirectional search: To increase the baseline's fitness for the Large Scale requirement, bidirectional search was implemented to reduce the search space. The one food piece is replaced with another node. Both nodes direct agents to find the other one. While there are 2 nodes, the program still only needs 2 types of pheromones as before. Agents from different nodes must be able to use the other nodes pheromones, otherwise the 'meet in the middle' concept of bidirectional search does not work. Luckily, as one node's 'to start' pheromone functions identically to the other's 'to goal', these can be used interchangeably.

3) Distance counters: The Heightmap Considerate requirement demands that paths minimize vertical and horizontal travel distance. As mentioned before, my implementation did not feature the path shortening and short path preference behaviour that was expected. Therefore, the baseline was extended to remedy this.

Originally, agents base their movement on pheromone freshness alone. This caused a populism effect, whereby agents ignore short new paths in favour of busy paths, which constantly get new pheromones placed on them. By adding a distance counter to each agent, their pheromones indicate how far an agent has travelled before laying the pheromone. The counter resets upon reaching a node. This solves the populism problem, as it enables agents to distinguish short from frequented paths.
In combination with distance counters, pheromone freshness has an averse effect on path quality, decreasing both the consistency of pathfinding and slowing down path improvement over time. This is likely because the populism effect discourages exploration by encouraging the conservation of the currently established path. As such, pheromone freshness will no longer be considered for pheromone evaluation. Now, it will be used solely as a lifetime count, with the pheromone being disabled and eventually removed when it reaches 0.

4) Exploration limitation: As mentioned before, previous research (14) shows that limiting the search space is beneficial for convergence time in ACO. I expect the same result for the gathering method, which could make it more suitable for large-scale environments. This extension adds two guiding elements to the baseline: an ellipse of initial pheromones and a bounding box, both positioned loosely around the start and goal node. The initial pheromone ellipse makes sure agents move approximately in the direction of the other node. These pheromones are very low value, and any other pheromone of the same type is preferred over them. When the ellipse has completely dissipated, it is possible for agents to stray too far away from either node, getting 'lost' in the large terrain. This severely decreases the chances of the agent finding its goal node, which means computing resources are wasted on it. To prevent this, the bounding box provides a hard limit to the search space. Agents reaching its edge have their walking direction mirrored on that edge.

5) Heightmap interaction: To make the baseline entirely Heightmap Considerate, it needs to be extended to also mind vertical path length. For walking paths, it is preferred to walk along flat terrain as opposed to uneven, steep or mountainous surfaces. In short: the lowest summed absolute vertical change is preferred during travel. Using the heightmap, agents can measure their elevation change during movement and add this to a 'spent energy' counter and add this value to dropped pheromones. Just like the path length, agents can read the energy spending required if they want to follow a given pheromone trail. With this agents can distinguish 'short' paths going straight over mountains from somewhat longer detours around them, with the latter being preferred. Furthermore, as uneven terrain also requires more vertical movement, these should also be avoided if possible.
Gradual slopes should be preferred over steep, sudden jumps in height, thus the spent energy counter exponentially increases energy consumption for larger height differences. The extent of this behaviour can be regulated by multiplying the spent energy counter with an effort weight parameter before adding the score to dropped pheromones.

6) Final path constructor: After the simulation is terminated, the result is a map of pheromone trails. To extract a clear path from this data, all agents are removed and a final batch of agents is released from either end of the path. These agents have no random deviation and use the current slime mould sensor model to make sure the optimal pheromone is never missed. When all agents arrive at their destination, agent's path with the best score will be the final result.

D. Thesis extensions

Below is a list of the remaining extensions required for the project, alongside methodology. These have been implemented during the thesis.

1) Between-path sharing: This extension enables multiple nodes to make use of overlap in their paths, which enables the baseline to meet the Join requirement. Instead of creating a separate pheromone map per path, all pheromones leading to a settlement are stored per node and are accessible to any agent searching for it. This way, if multiple nodes have the same goal they can use each other's search results to aid in finding a path. As a consequence, there is a chance that multiple nodes utilize some of the same pheromones, which causes path joining. As a bonus, the sharing of pheromones increases the searching resources for all nodes involved, which should lead to faster convergence times and more exploration, increasing path quality. However, this might also add a bias towards path sharing, as other options might be left unexplored after a initial path is quickly set up.

2) Long-distance pheromones: As pheromones dissipate over time, long distances can cause them to disappear before a sustainable path can be based on them. To extend the baseline's quality in terms of the Large Scale requirement, this issue will be addressed by increasing pheromone lifetime based on expected path length. Initially, this would create issues with paths sharing pheromones with different freshness standards, but as freshness is no longer considered when agents search pheromones it will be no problem.

3) Selective node pairing: DAPA v0.5 is capable of simulating multiple paths at the same time. However, it attempts to create paths between all nodes by default. Just as with slime mould, this is not desirable as it lacks the control to (dis)connect specific nodes. To replicate the structure of real-life road networks it is preferred to only connect settlements relatively close to one another. Using a Delaunay triangulation on the settlements, the edges created will represent potential paths. User will be able to add/remove potential paths after this process. All settlements that have a potential path within a predetermined range will have a path generated. If all edges to a settlement are outside of range, the shortest among them is generated anyway. Lastly, if the edges to two nodes are too close together, the longer edge will be removed, as a detour via the shorter edge to the node further away would be more natural.

4) Probabilistic sensor: Attaining the Winding requirement is a difficult task with DAPA v0.5's pheromone information setup. This is because agents will ignore terrain while not detecting pheromones to increase exploration. Subsequent agents can then judge the path quality. This means that for winding behaviour, an agent must at some point make the winding motion before others can deem that as the most preferred path. This is currently highly unlikely to happen.
This is where Pezzza's Work's original sensor model could offer a solution. Unlike the currently implemented slime mould sensor model from Sebastian Lague, this one is probabilistic. Agents have a 'sensing cone' in front of them from which they sample a random space every simulation iteration. Because agents are blind to any pheromones outside this space, they will miss pheromones otherwise detected. On average, however, they will keep following trails, especially if they are already well established. The difference in behaviour is that agents now have a higher chance to somewhat sway of the beaten path, increasing exploration. If this behaviour is maintained over numerous iterations on a slope where lower inclination movement is preferred, small diversions turn into bends, which turn into the winding behaviour required. A nice bonus is that with only one sensing area per agent the processing time will also be reduced.

5) Copy pheromones: DAPA v0.5's paths change too little from their initial shape, even if this would be beneficial. The quality of the paths is therefore not always adequate. This is because agents only drop pheromones describing the quality of their total path traveled. An agent joining and improving a popular path halfway will thus be ignored if the path it travelled before was of low quality. This extension aims to solve this problem with a simple solution: agents copy the properties of any same-type higher-quality pheromones they walk over, meaning that any improvement made is guaranteed to produce higher value pheromones than the ones present.

6) Termination condition: The final extension is again for the Large Scale. Many Large Scale extensions are needed, as the search space increases drastically as the distance between nodes does. In DAPA v0.5, users can indicate how many iterations the entire simulation must perform. However, because shorter paths will need fewer iterations for a good path than long paths on average, this number will always be suboptimal for most paths. To reduce wasted computing time on short paths and increase the quality of longer paths by way of more iterations, a termination condition will be introduced. If this condition is met, the simulation for the specific path is stopped. Once an agent has reached its goal node, DAPA will keep track of the best path result between the start and goal node. If the path quality stagnates for too long, it is safe to assume the simulation has reached a (local) optimum that will not change much in the future, indicating that the simulation for that path can be stopped.

V. Experiments

A number of experiments will be performed to measure to which degree the requirements have been met with DAPA v1. The aforementioned 'Authoring Hierarchical Road Networks'(11) would make for a good technique to compare against, as it shares the most requirements with DAPA v1 out of all techniques considered. However, no code is available of this technique in practice, and there was insufficient time left to create an implementation from scratch. That is why, as mentioned in the previous section, DAPA v0.5 will be used to compare against v1. In table III, we have an overview of the difference in extensions between the two versions.

The terrains upon which the experiments will be held are 256x256 by default unless stated otherwise. In terms of scale we determine 1x1 grid space to correspond to $1m^2$. Nodes will be denoted with capital letters, e.g. 'A' and 'B'. When only 2 nodes are used, they will be placed 200m apart unless stated otherwise. Any path characteristics will be measured from the final generated path created by the final path constructor extension. This feature is a post-processing extension and thus will not impact the simulation itself. In table IV an overview of all experiment measurements is shown.

Experiments may be comprised of multiple parts, which I will refer to as *tests*. Different tests within an experiment make the same measurement for the same requirement, though under different circumstances.

Experiment results will be based on the data collected from 100 simulations per parameter test, unless stated otherwise. This means that experiments with multiple tests - such as the considerate experiment testing different effort weights - execute 100 simulations for each one.

Unless mentioned otherwise, the simulation uses an effort and distance weight of 1. When randomly generated terrain is used, both DAPA versions are applied to each of these terrains, meaning that the terrain input is varied yet identical for either version.

As the distance between nodes gets bigger, the chances of finding a path while maintaining the same number of agents decreases significantly. To find the appropriate parameters for the different node distances and their corresponding map sizes (256-2048m^2), I have tested different agent numbers, aiming to get at least 90% success rate for v1. Again, 100 simulation results are averaged each time to approximate their effectiveness. To approximate general performance, the simulations are executed on randomly generated terrain. This resulted in agent quantities of 300, 1000, 2500 and 6000 for node distances of 200, 400, 800 and 1600m respectively. Although 6000 agents is still not enough to reach 90% success rate, diminishing returns and a vast increase in compute time lead me to eventually choose this number as a compromise.

Lastly, due to time constraints, the maximum number of iterations will be approximated to 1000 iterations for a 200m node distance path, doubling with alongside the node distance. This might leave small simulations running for too long and large ones too short, but the significant time needed to study the interaction between node distance, number of agents and maximum number of iterations in its entirety is not available.

1) Heightmap Considerate: For this experiment, the goal is to measure to what degree v1 and v0.5 take sudden and gradual terrain changes into consideration when generating paths. Furthermore, as the effort weight changes the impact of terrain on the simulation, these measurements will be done with effort weight values of 0.01, 0.5, 1 and 10 for comparison. To measure the consideration of different terrain changes, heatmaps will be produced of the paths generated by both DAPA versions and compared to one another. Afterwards, I will speculate the results and what decisions led to them. To test sudden and gradual terrain change consideration, two experiments will be set up.

The first one will contain two nodes featuring a sloped wall with an apex of 37.5m exactly between them, extending perpendicular to one edge of the terrain, see figure 3A. This gradual slope gives the DAPA versions the option of scaling it to reduce path length or avoid it to reduce the effort of traversing the path. This way, the consideration of gradual terrain change can be well observed.

For the second experiment, consideration of sudden terrain change is tested. A flat terrain is used where one node is elevated by 37.5m. The node is accessible by either moving up a steep ridge or a slight slope that is on the side of the node, see figure 3B.

I expect the wall will be directly scaled in the first experiment when the effort weight is below 1, while higher values the mountain might be somewhat scraped but averted overall. In the second experiment, paths should cover the slope rather than the jump in every case, unless the effort weight is set to extremely low levels.

2) Smooth: While the difference between grid-bound and free flowing paths is clearly noticeable, it is difficult to quan-

TABLE III: DAPA extension difference

	Directed pathing	Bidirectional search	Distance counters	Exploration limitation	Heightmap interaction	Final path constructor	Between-path sharing	Long-distance pheromones	Selective node pairing	Probabilistic sensor	Copy pheromones	Termination condition
DAPA v0.5	✓	✓	✓	✓	✓	✓	×	×	×	×	×	×
DAPA v1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

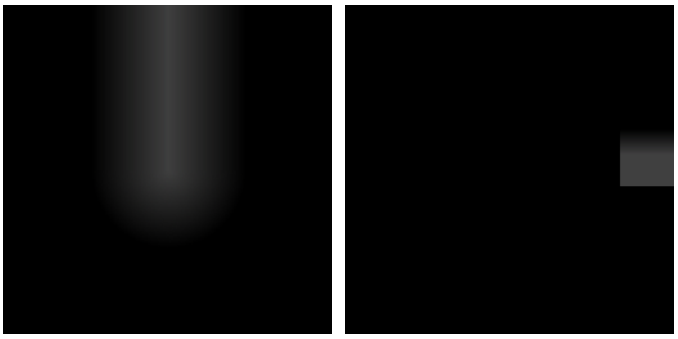

Fig. 3: Considerate terrain layouts. A: first setup, sloped wall. B: second setup, elevation with side slope.

tify smoothness. To approximate it, the final path's movement will be measured. The results will be represented in a scatter plot which shows how often bends of different horizontal angles are made along the path. Furthermore, the change in bend curvature over the course of a final path will be measured. A smooth path should gradually curve to bend around obstacles.

These measurements will be made for paths with node distances of 200, 400 and 800m to get results on different scales.
Later during experimentation, the need for a fifth test arose. This test includes the same terrain as we will see featured for the Imperfect obstacle test. This time, the terrain is used to measure the amount and size of bends in paths when curves are induced by the obstacle on the map. To force the avoidance of the obstacle, the effort weight is set to 1000 for this test.

The expected outcome of this experiment is a generally high % of the path containing small curvature bends and a low change in curvature.

3) Large Scale: In the following experiment, the computation time and memory performance of the two DAPA versions will be compared.

This will be done in two tests, both varied in scale (node distances of 200-1600m) and performed on randomly generated terrain featuring steep mountains, hills and plains. By performing the simulation on varied terrains, a better average performance should be measured than for a static map choice. The first test features a generic 2-node setup to test the single path performance for different node distances.
In the second test, 3 nodes are set up in a triangle, all set apart by the corresponding node distance. This experiment tests the influence of multiple paths and path length on performance. For this setup, three times as many agents will be used, as three paths will be generated.

Because the experiment returned some unexpected values, the first test was done once more at 10 iterations with the alteration of measuring the average maximum memory usage per test instead of the absolute maximum.

The goal is to produce valid results on the 3 node 2048x2048 setting within 5 minutes with a memory usage of 2GB or less.

Expected results for this experiment would be a general decrease in computation time due to the combination of the *Termination condition*, *Probibalistic sensor* and *Copy pheromones* extensions, with a slight increase in memory usage due to the longer-living pheromones the *Long-distance pheromones* extension provides.

4) Joining: When joining occurs, a segment from two paths should overlap or stay in close proximity to one another. To show the degree of joining, we measure how big this segment is by checking how many consecutive path points are within 3m of the closest point from another path.
During experimentation, it became clear that multiple joined segments might appear between paths. As such, all joined segments between paths will be measured (instead of only the first one starting from the goal node), as well as the average number of segments per test.

For this experiment, three nodes will be placed in a triangle formation on a flat terrain, with A in the middle of the map and AB and AC being 100m apart. Multiple test sets will be performed where angle BAC is 180°, 90°, 45°, 30°, 15° and 5°. Paths along edges AB and AC will be formed, but not BC. This gives paths AB and AC the opportunity to join paths whenever they come close to one another.

This measure should be much higher for an algorithm with the joining property than one without it, while it should still remain near-0 for paths that do not come close together. DAPA v0.5 is only able to create connections between all settlements, but the extra path BC will have no effect on the course of AB and AC, as v0.5 contains no degree of joining. This means we can safely generate and ignore any final paths for BC.

5) Winding: For this experiment, winding behaviour will be defined as 'movement at a low incline relative to a scaled slope, moving back and forth along the slope in a zig-zag

pattern of varied lengths'.
As long as a path does not move backwards, the vertical path length should remain the same between winding and non-winding behaviour. However, we do expect horizontal path length to change, along with the maximum and average inclination, according to this definition. These characteristics will be measured in the following experiment to determine to what degree winding can be detected for v0.5 and v1.

The terrain setup for this experiment will be a gradient slope from A to B. The experiment will be held with inclination angles ranging from 0° to 50°, incrementing in steps of 10°. Due to technical limitations of the terrain generation program I made, this experiment will have to be performed on a 128x256 terrain, with A and B 100m apart, parallel to the slope and the slope moving along the shortest axis. This way, a 50° slope is possible while staying under the artificial height maximum of the program, while giving the agents enough space to wind along the longer axis.

The expected result from this experiment is for v1 to showcase stronger winding behaviour than v0.5 due to the increased agent exploration and path improvement of the *Probabilistic sensor* and *Copy pheromones* extensions.

6) Imperfect: To measure the deviation from the optimal path, the standard deviation of the path length difference is calculated over multiple simulations for both scenarios.

As no perfect pathfinding algorithm was found which matches the thesis' requirements, we have no means of determining the optimal smooth path to compare against[4] By imitating uniform-cost terrain scenarios, however, we can get an optimal reference length with straight line paths.
Uniform-cost terrain is imitated by only allowing flat traversable terrain. Walls are represented by elevated segments. By increasing the effort weight to 1000, any and all height changes should be avoided at all costs.
This does mean that only the horizontal imperfection can be measured, as uniform cost terrain is 2D. However, this will still give a strong indication of DAPA's imperfection, as the only introduced imperfection in the technique is that of random horizontal deviation of the agent's movement.

For this experiment, two scenarios will be considered.
Firstly, an empty, flat terrain with nodes A and B. We will test this with a node distance of 200m, 400m and 800m to see the impact of scale on imperfection. Consequently, terrain size will also scale from 256, 512 to 1024m^2, respectively.

Second, a flat terrain with a 20x80m obstacle with a height of 37.5m aligned perpendicularly between A and B is tested, see figure 4. This scenario will show the impact of curves on the deviation of perfect path length.

For this experiment the expected outcome would be for the path deviation to scale with node distance. Furthermore, path deviation should be lower for v1, as the *Copy pheromones* extension should improve the quality of generated paths.

[4]An exception would be the last approach(11) mentioned in the methods section, though it is not optimal. As there is no public implementation of the approach available and because there is far too little time to recreate it from scratch, it is not possible to use it for comparison.

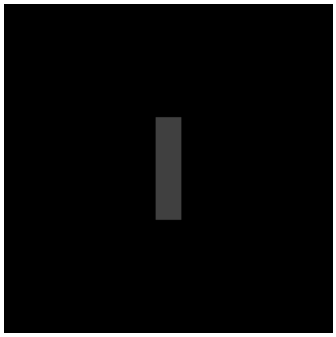

Fig. 4: Imperfect terrain layout. The obstacle in the middle forces paths to curve around it.

VI. Results & Discussion

Below, we present the results from the above mentioned experiments. Many of these results are displayed in scatter plots. These plots represent individual simulation results as grey dots, which are grouped together in buckets for clarity. The domain size of each bucket is 1 unit, where decimal results are rounded down before being added to the graph. The groups are normalized, where the number and size of dots indicate the frequency of occurrence compared to the most frequent result. The most frequent result is represented with 6 full size dots. Blue dots indicate the average value, purple the standard deviation[5] added/subtracted to/from the average, red dots indicate the maximum value and orange dots show the standard deviation of the maximum value.

It should be noted that DAPA version 0.5 performs significantly worse than v1 for long paths (800m+), with often times no path being found before the iteration limit is reached[6]. With many of the experiments being applied to these terrains, it becomes difficult to collect enough data on successful v0.5 path results. As a consequence, the presented v0.5 data for larger maps is (significantly) less representable than desired. To show the degree of generalizability for each experiment and version, I will mention the number of successful paths that each experiment test is comprised of.

1) Heightmap Considerate: To start off, we will look at the results of the Heightmap Considerate experiment. Ideally, we want to see the resulting paths to more strongly avoid slopes with increasing effort weights as they try to balance path

[5]It should be taken into consideration that the standard deviation is expressed in squared units, e.g. $\%^2$. Presenting them alongside the other values in the graph is thus not entirely accurate, though they have been left in as an indication of relative deviation between test results.

[6]This is in part due to an elusive bug in the final path construction step which could not be solved in time. Though, solving this would probably only improve the success rate of the final path construction by at most 50%. As we will see in the results, this would still be far too little for adequate overall success rates.

TABLE IV: Experiment overview

Requirement	Measurements
Heightmap Considerate	Path routing heatmaps for different effort weights and terrain setups.
Smooth	Path curvature and curvature change for different node distances.
Large Scale	Speed and memory performance for different node distances.
Joining	Total joined lengths between paths for different path proximities.
Winding	Vertical and horizontal path length, max. inclination.
Imperfect	Average deviation from perfect path for different node distances and for paths with induced corners. Uniform-cost terrain is simulated with flat terrain, high walls and high effort weight.

distance and effort. Furthermore, slopes should be preferred over sudden jumps in height.

In the first experiment test, we tested the behaviour of the two DAPA versions in minimizing distance and effort depending on effort weight. The second test means to test the versions' behaviour when facing slopes and sudden jumps in height, depending on effort weight as well. We see the results of these experiments in figure 5 (v0.5: 96, 65, 53, 41 out of 100 paths successful, v1: 100, 100, 97, 46) and figure 6 (v0.5: 76, 48, 44, 37 of paths successful, v1: 100, 91, 90, 77) respectively.

For both tests, we notice that v1 has less concentrated results than v0.5 overall. This is probably the result of the *Probabilistic sensor* extension of v1. By adding more randomness to agent movement, fewer straight paths are walked, meaning that more effort is required to make them as straight as v0.5's. Furthermore, the randomness also causes an increase in exploration but not thoroughness, meaning that path with a wider variety of initial shapes have a chance of emerging.

For the first test, v1 does not react as strongly to increased effort weight as v0.5. We can even see this at an effort weight of 0.01, where v0.5 already has a slightly down-curving overall shape, while v1 makes no noticeable adjustments. Notably, however, v0.5 still seems to incorrectly cross straight over the slope for effort weight 10 more often than v1, even though these v1 results contain more than twice as many samples. The difference might be the cause of the *Termination condition* extension being set too sensitive. If the termination condition deems the path improvement during the simulation below a certain level, it stops the simulation and begins final path construction. If it is set too sensitive, it could cut off path that had the potential of curving along with the terrain more. This could have also explained why v1 performs better at effort 10: the higher weight makes for higher path scores. Higher scores means higher differences between path scores as well, which holds off the termination condition from cutting the simulation short. However, looking at the average number of iterations per simulation (1228, 1207, 1197 and 1181 for v0.5, 1204, 1205, 1205 and 1206 for v1, respectively), it is clear that this latter conclusion is incorrect.

For the second test we notice the opposite of the first one: v1's paths much more consistently react to a higher effort weight than v0.5. This is likely due to the *Copy pheromones* extension, as this should drastically increase the simulation's path improvement capabilities. With the extension, agents are better able to adjust a defined path to take into account the terrain, as seen in the v1 results of figure 6. While this was not effective enough for the first test, it might have had a larger impact here due to the area that needed to be improved being much smaller. Instead of the whole path needing to be improved to the shape of the hill in the first test, here, it is only needed to improve a small area of very high contrast possibilities (being the slope and the other steep sides up to the node).

It would seem that both versions contain desirable aspects concerning Heightmap Considerate behaviour. Though, as neither performs desirably in both tests, the Heightmap Considerate requirement is not met.

2) Smooth: For the measure of smoothness, the experiment tested the change in angle between the path points for different map sizes. Furthermore, the change in angle while moving over the path has also been measured.
Reviewing the results for the first time, I realize that the measure of average curvature change was not a good indicator for smoothness in combination with the terrain setup, as slightly curved paths can be just as smooth as straight ones, but give different results with this measure. This did not combine well with the terrain setup of an empty, flat surface, as it allowed for a great deal of variation in path shape, muddying the comparison of path properties. For a slightly more comparable experiment setup, I added a test using the same obstacle layout as the Imperfect experiment, where paths will be more comparable as they avoid the wall in a similar manner. The results are added to figure 7.

Looking at figure 7 (v0.5: success rates of 98, 84, 6 and 52, v1: 100, 100, 98 and 92), we see that v1's distribution of bend size is a lot more gradual than v0.5. While the crude distribution of v0.5's 1024x map results could be attributed to the low number of samples collected (only 6/100 successful paths), the same cannot be said for its 256x and obstacle map results. It would seem that v0.5 mainly goes straight ahead along its shorter paths, while longer paths are harder to optimize for both versions, resulting in more bends and thus a wider spread of bend sizes.
The v1 results on the obstacle map are notably different from v0.5, which more resembles the profile of the other results. Instead, v1 has a clear higher averaged spread of curves.
V1's results indicate smoother curves overall if we take into consideration that, for a curve to be smooth, slightly sharper or duller curves need to occur about as frequently. The smoother distribution of v1 shows that this is the case.

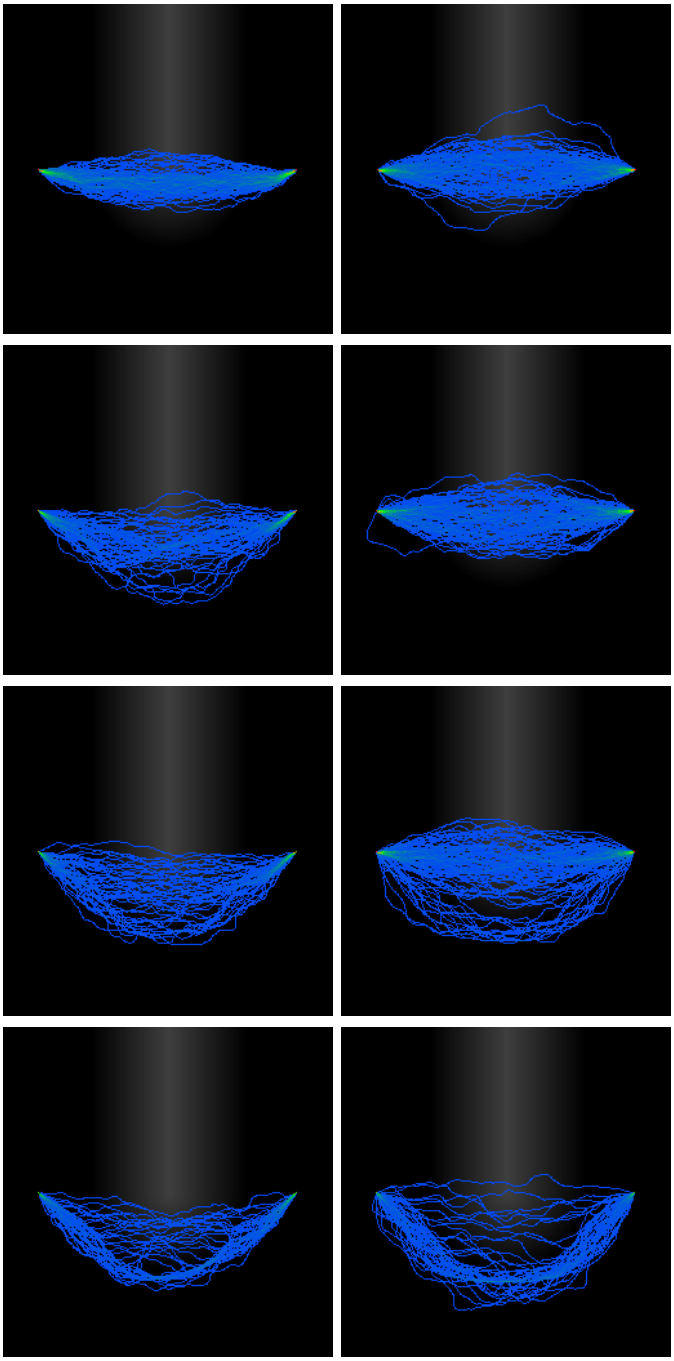

Fig. 5: Heightmap Considerate first setup results. Left column: v0.5, right column: v1. Top to bottom: results with effort weights of 0.01, 0.5, 1 and 10 respectively

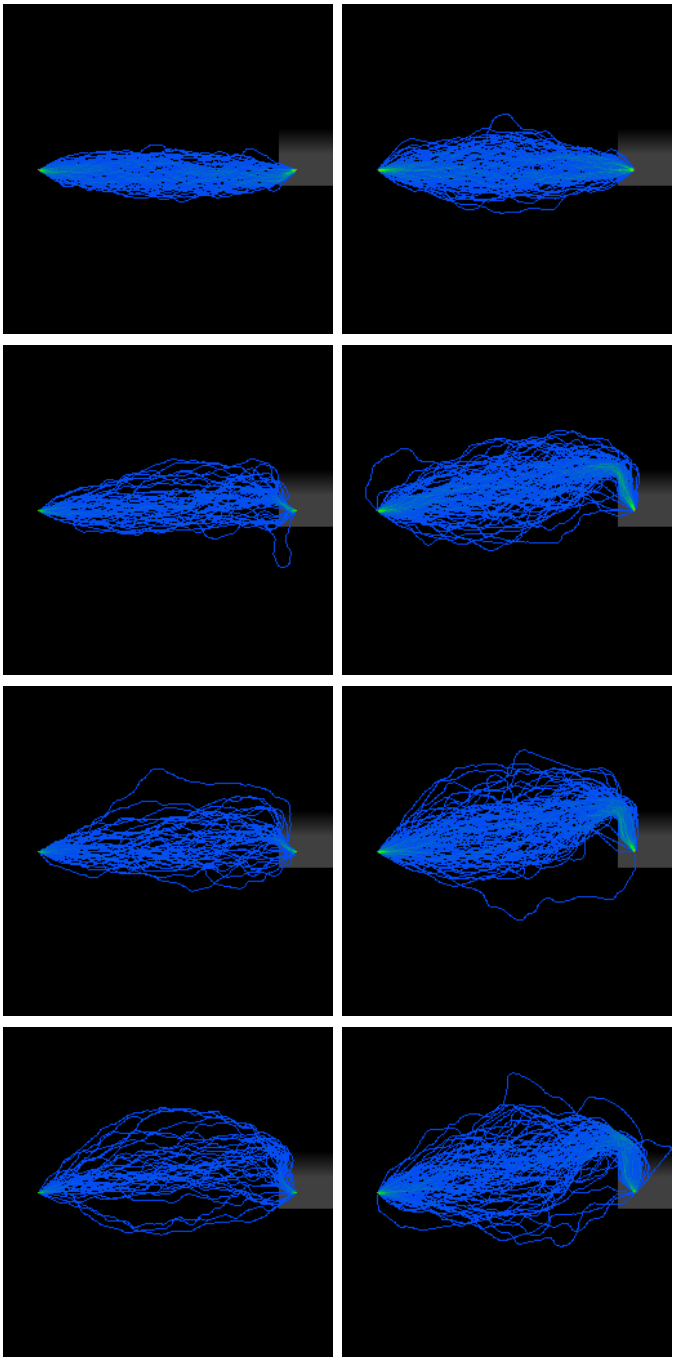

Fig. 6: Heightmap Considerate second setup results. Left column: v0.5, right column: v1. Top to bottom: results with effort weights of 0.01, 0.5, 1 and 10 respectively

From table V it is clear that the average curvature change has increased between versions, while the standard deviation has stayed relatively the same. This is most likely due to v1's agent sensors working differently than v0.5's (see *Probabilistic sensor* in *Thesis extensions*). V1 only has 1 sensor randomly positioned in a cone in front of the agent. This randomness probably results in the creation of fewer straight paths, meaning that the paths on average make more turns which causes the difference between v0.5 and v1.

On the contrary, the obstacle test shows a slight decrease in curvature change. As the terrain incites paths to create comparable paths with curves around the obstacle, I can say with more certainty that this indicates smoother behaviour around induced curves as opposed to curves created by random agent movement.

In either case, the change in curvature is very minimal. Though v1 makes sharper curves in general, these curves appear gradually. With that, I deem v1 to attain the Smooth requirement.

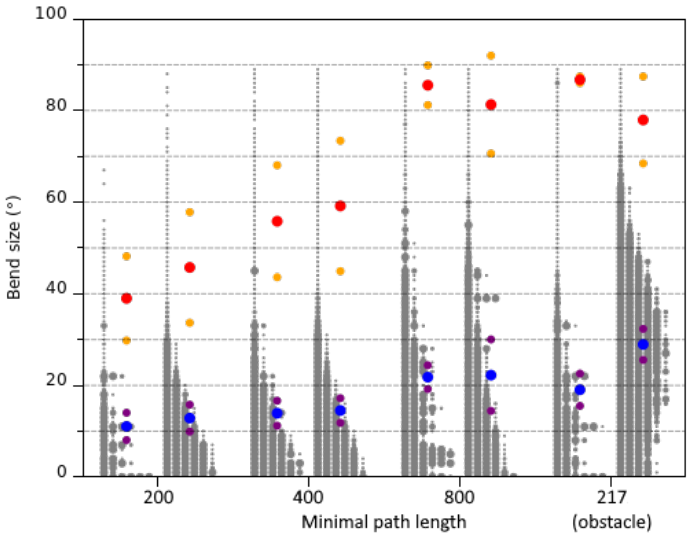

Fig. 7: Smooth results. Left columns: v0.5, right: v1.

TABLE V: Smooth curvature change (°)

	v0.5		v1	
Minimal path length	**Average**	**Std**	**Average**	**Std**
200	1.58	0.55	3.76	0.52
400	2.34	0.46	4.28	0.35
800	4.01	0.47	4.27	0.49
217 (obstacle setup)	4.62	0.63	4.38	0.50

3) Large Scale: The Large Scale experiments have all been executed on a Lenovo ThinkPad w540 with an i7-4800MQ processor and 8GB of RAM. The execution of the experiment has taken more than time expected, with the vast majority of the time spent on calculating the 1600m distance paths. After simulating 100 single paths for both versions, it was clear that attempting to do the same for the three path configuration of the experiment would simply take too much time to be practical. Because of this, I decided to reduce the number of simulations for the three path test down to 10. As this test is not as reliant on specific results and is rather a ballpark indication of the influence of path length on computation time and memory usage, this reduction should not be too much of a problem.

From the experiment results in table VI it is clear that v1 is significantly faster than v0.5. It seems that the *Termination condition* and *Probabilistic sensor* extensions have had their intended effect. For single paths, the latter decreased the number of iterations with 18.1-45.5%: v1 averaged 671, 1318, 2971 and 7089 iterations for the different node distances, whereas v0.5 needed 1226, 2419, 4633 and 8658 iterations on average.
For the triple path setup, iteration amounts were lowered by 19.0-47.3%: 1246, 2484, 4804 and 9022 versus 778, 1310, 2932 and 7304.
Meanwhile, the decreased number of pheromone reads per iteration due to the *Probabilistic sensor* seems to have caused an improvement of computation speed of 12.3-70.7% for single paths, looking at the differences in average iterations/s in table VI. For triple paths, the speedup ranges from 38.1-131.3%. The larger speedup for longer paths has most likely to do with the number of agents becoming the bottleneck of the program. As pheromone reading is the most taxing part of agent behaviour, a lot of performance can be gained with the optimization that the *Probabilistic sensor* performs on this segment. Very notable is that, on average, v1 is faster constructing 3 paths than v0.5 from any distance. Furthermore, looking at the increase in simulation time between single and triple paths, we see that while v0.5 sees time increases of 26.7-316.6%, v1 scales much better with 15.6-230.8%. This is likely the result of aforementioned performance enhancing extensions, as well as the *Between-path sharing* extension helping paths work together for faster overall pathfinding.

It should also be noted that v1 sees much more success constructing paths than v0.5, especially with longer ones: v0.5 has success rates of 86, 66, 31 and 7/100 on the single path test, while v1 boast rates of 96, 99, 92 and 77/100. For the 1600m node distance test (on 2048m^2 terrain), this means an 11$\times$ increase in success chance, and around 3$\times$ for 800m. We can thus confirm that the *Long-distance pheromones* extension works as expected, that is to keep pheromones active for long enough to give agents a chance to find other's paths towards their goal over long distances. For comparison, in the triple path setup v0.5 completed 21, 13, 5 and 1/30 paths over 10 simulations, while v1 successfully generated 26, 30, 29 and 26/30.
Looking at the success rates of the Heightmap Considerate and Winding experiments, however, we can see that v1 still struggles with high effort weights as success rate drastically decreases alongside it.

TABLE VI: Large scale computation times

	v0.5			v1		
Node distance (single path)	**Average(s)**	**Std(s^2)**	**Average Itera-tions/s**	**Average(s)**	**Std(s^2)**	**Average Itera-tions/s**
200	13.1	0.9	93.4	6.4	0.6	104.9
400	48.9	4.1	48.5	17.1	1.2	77.1
800	292.0	16.5	15.9	109.9	6.3	27.0
1600	2123.9	86.1	4.1	1010.1	66.0	7.0
Minimal path length (triple path)						
200	16.6	1.0	76.4	7.4	0.4	105.5
400	154.8	10.6	16.0	35.4	2.7	37.0
800	892.5	105.8	5.4	253.6	23.4	11.6
1600	2623.4	11.9	3.4	1224.1	43.0	6.0

In terms of maximum memory usage, we see in table VII that both DAPA versions perform about the same under both single and multiple path conditions.
The first thing of notice is very low single path average memory usage for both versions relative to the maximum and the triple path results. It is unclear how these values can reach such low numbers, much less why they decrease with node distance.
Another point of interest is the significant memory spikes that only seem to occur for single paths. This is likely in part due to a flaw of the experiment: the maximum is only simply the largest memory reading during all of the simulations for a given test. The maximum values could thus be outliers, as no average maximum per simulation is calculated. With only 10 iterations for the triple path test, it could be chance that these outliers were not encountered here

To verify this, the single path test was re-executed at 10 iterations while documenting the average maximum memory usage between iterations instead of the absolute highest peak across all iterations in a test. From these results, seen in table VIII, it becomes clear that the average maximum is much more consistent with the average memory usage, just like the triple path results indicate in figure VII. A peculiar detail to note is that for node distances 800 and 1600m, v0.5 and v1 have a lower average maximum than average memory usage respectively. It is unclear what causes this. Furthermore, with this adjusted experiment, it is safe to say that the high maximum memory usage values of the initial experiment were only outliers.

V1 is able to stay below the 2GB memory usage goal, but cannot create the three 1600m paths under 5 minutes. The criteria has thus not been met, but significant strides have been made towards it considering the increased success rate and computation time.

TABLE VII: Large scale memory usage (mb)

	v0.5		**v1**	
Node distance (m) (single path)	**Average**	**Maximum**	**Average**	**Maximum**
200	5.8	277.3	27.6	281.3
400	1.0	581.3	25.0	585.3
800	4.7	1445.3	9.5	1449.3
1600	0.2	4917.4	5.7	4913.4
Node distance (triple path)				
200	147.6	149.3	151.4	153.3
400	173.2	181.3	175.8	185.3
800	244.1	277.3	247.5	281.3
1600	500.5	645.4	500.8	657.4

4) Joining: As per the joining requirement, paths should join if they run close to one another, rather than having multiple separate paths side by side. In figure 8 (98, 93, 86, 83, 94 and 96 samples for v0.5, 100, 99, 99, 99, 99 and 100 samples for v1) we can see the impact of the *Between-path sharing* extension for both DAPA version for different between-path angles between the two generated paths. Note

TABLE VIII: Large scale memory usage (mb), adjusted test

	v0.5		**v1**	
Node distance (m) (single path)	**Average**	**Average maximum**	**Average**	**Average maximum**
200	151.6	151.7	151.4	151.7
400	175.7	175.7	177.2	177.3
800	244.6	244.5	244.5	244.5
1600	498.2	501.4	508.5	501.4

that no values of 0% joining are present: 2-3 path points around the shared goal of the paths are always within 3m of one another. Because this experiment can only be measured when both paths are successful, the described situation is always present.

As is to be expected, path joining values only rise in the tests where paths come close together. From 30° on, we see an increase in joining % from v1 over v0.5. However, the standard deviation increases as well. There are two possible reasons for why this is the case.

Firstly, this dispersion of results may be the consequence of the experiment measure chosen. As mentioned in the experiment section, for Joining we originally measured the number of consecutive points that are close together from both paths. However, when a long joined segment contains a single path point that is not close to the other path, this number gets cut short as only the first segment is registered and the second is ignored.
To find out if this reasoning was valid, the experiment was extended and executed again. This time, the experiment measures the total % of joined path, instead of only measuring until the first >3m disconnect occurs. Additionally, the average amount of joined segments per simulation and their lengths was measured. The same characteristics were measured for the segments of path disconnection between connected parts. The results of this extended experiment are already included in figure 8. The new measurements show that all joined paths across all simulations for v1 and v0.5 in this experiment have exactly one segment, with the average segment count being one, leading to the average segment length matching the total joining length.
From this we can conclude that the increase in dispersion is not the result of increased joined path segmentation.

Second, as discussed in the Heightmap Considerate results, the *Probabilistic sensor* extension increases exploration, meaning that the initially generated paths AB and AC might not emerge close together. This lowers the chances for agents from either path to find and make use of the other's path.

From figure 8 it can be concluded that the degree of joining has certainly increased from v0.5 to v1. However, as we can see in figure 9, the improved joining still leaves room for unwanted separation between joined paths, albeit small. Due to this, together with the inconsistency of the % of path shared, I deem the joining requirement not to be met, though the results are promising.

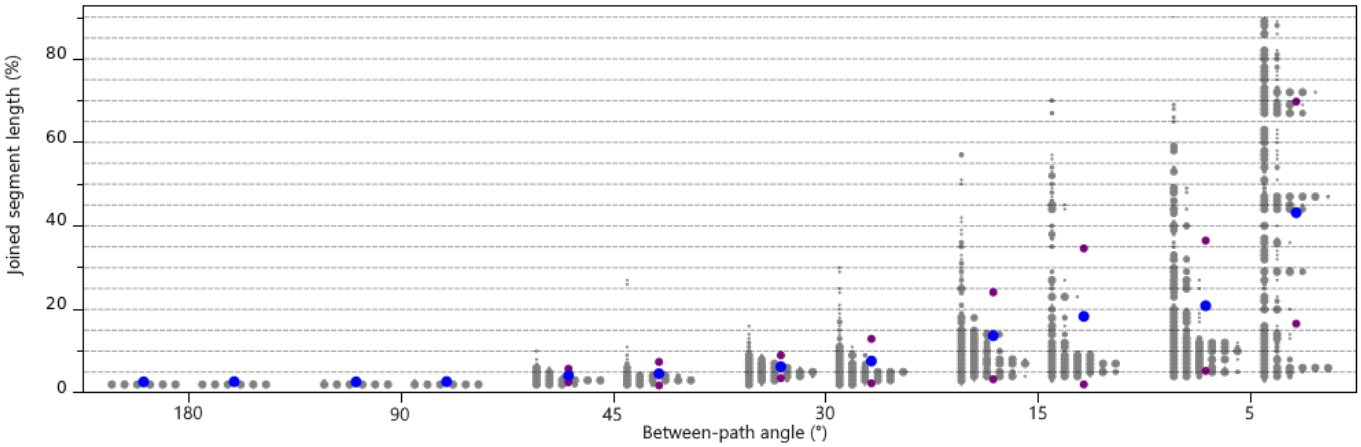

Fig. 8: Joining results. Left: v0.5, right: v1.

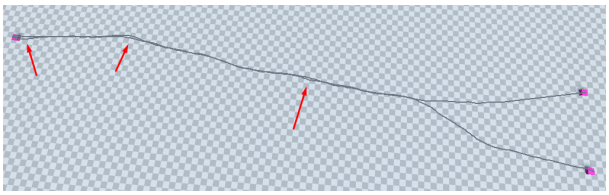

Fig. 9: Unwanted separation behaviour

5) Winding: It should be noted that the results of v0.5 for this experiment are somewhat incorrect, as they included rather specific inclination values of over double the maximum possible slope of different tests. This bug only occurred a handful of times, depending on the test. However, as the large values influenced the calculation of averages and maximal values, they have been left out of the graphs presented below. It is not known if this bug also impacts values below the maximum incline limit, but I expect the lower inclination spikes in e.g. figure 10D to be a result of this. The same experiment code is used for the calculation of the incline angles. Both versions use the same function for calculating agent and path point elevation, meaning that the problem likely stems from how agent movement is coded in v0.5.

Secondly, due to downward rounding, all highest inclinations fall into the bucket just below the maximum inclination. The result of floating-point rounding errors and/or inaccuracies cause some values to fall into the actual max inclination bucket by a margin of 10^{-4} or less.

In the Winding experiment description, I mentioned that a winding path should ideally have an increasing horizontal path length as it moves somewhat perpendicularly to the slope, scaling with slope steepness. Furthermore, the vertical path length should not be affected by inclination, as paths should always be moving up, resulting in nearly the same vertical movement every time. Paths should always be moving up because moving away from the goal does not serve path optimization and will thus be filtered out before the simulation is over. Lastly, the average and maximum inclination of the path should decrease with slope steepness as the agents have a stronger aversion to scaling steeper slopes, especially with higher effort weights.

First of all, the graphs of image 10(A)-10(E) show that the expectation of decreasing maximum inclination was insufficiently thought out. It stands to reason that the maximum inclination of a path is always equal to the slope incline. In fact, it **must** be. As the two nodes are placed exactly behind one another along the slope, only diagonal movement for slope minimisation is not enough to reach the other node. A diagonal path must eventually turn around, or else keep moving further away from the goal. At such a turn, the full steepness of the slope must be traversed. The alternative to this is a straight line across the slope towards the goal. Logically, this includes moving against the full incline of the slope.

In the same graphs we see a pattern in v1 of slightly higher average inclinations, but lower standard deviations and a cleaner spread of inclination frequency compared to v0.5. From table IX and figure 10(D) we can see that the uneven spread of inclination frequency is likely not due to under-sampled results: both results are based on approximately the same number of samples, yet the difference in spread of inclination frequency remains. As mentioned before, the spikes of low inclination could be the result of a bug in v0.5's movement code. By increasing the frequency of non-average values, these spikes lower the average and increase the standard deviation. If this is the case, I can only conclude that v1 has a more solid spread of increment frequency, likely due to the increased path quality consistency of the *Copy pheromones* extension.

As expected, we see in table X that the vertical path length remains basically unchanged when changing the effort weight. Furthermore, we notice that v1 has shorter paths than v0.5 across the boards, with only few exceptions. For the 30-50°

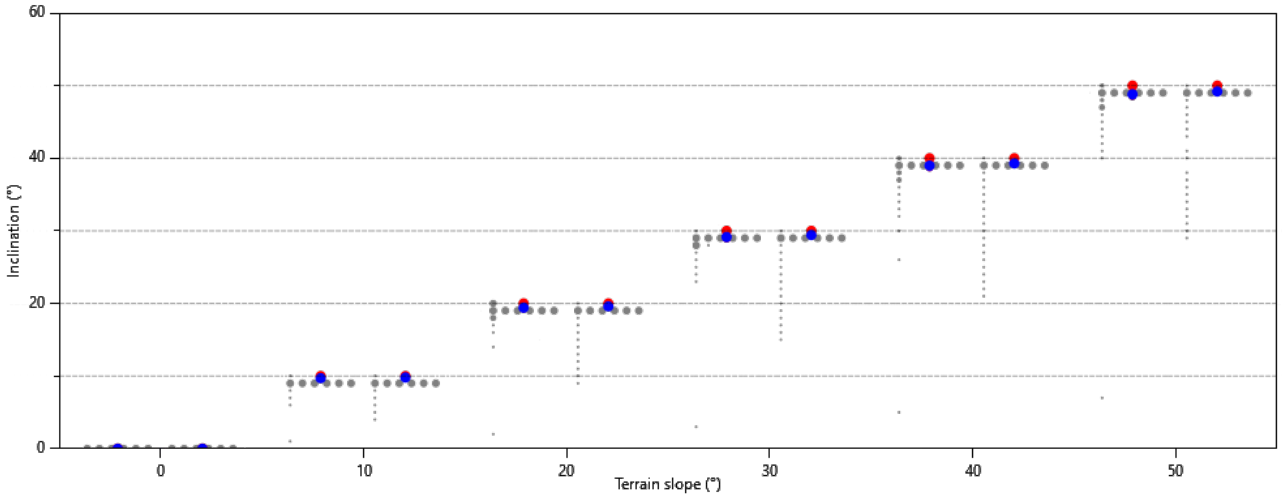

Fig. 10: (A) effort weight 0 results

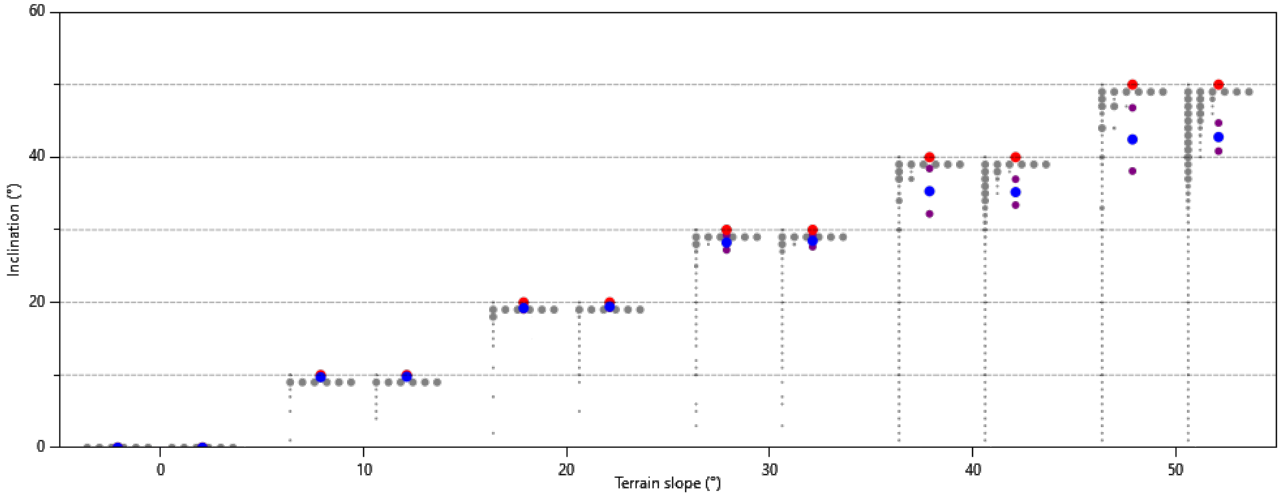

Fig. 10: (B) effort weight 1 results

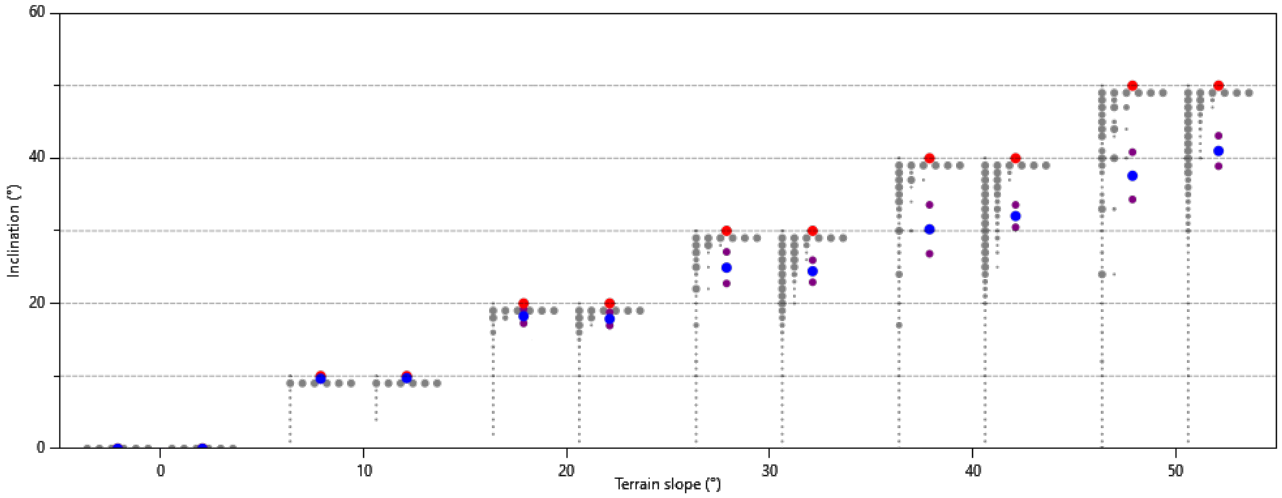

Fig. 10: (C): effort weight 5 results

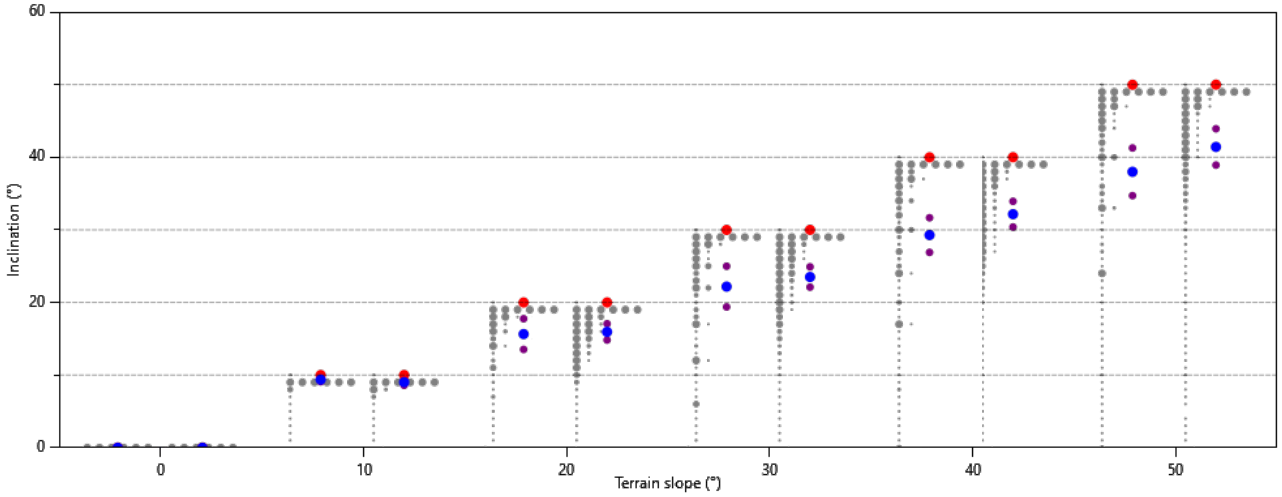

Fig. 10: (D): effort weight 25 results

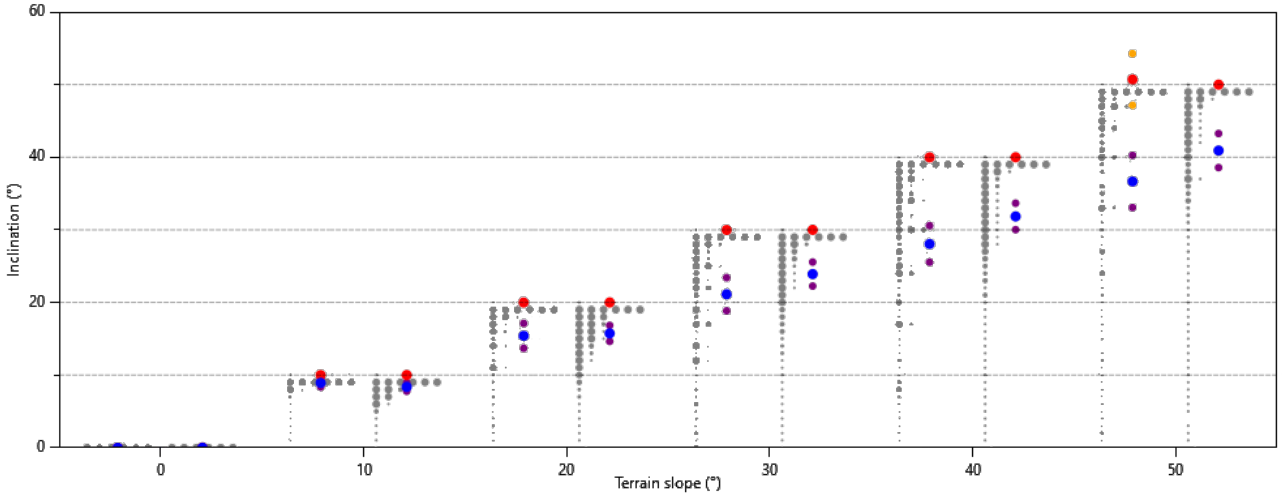

Fig. 10: (E): effort weight 100 results

Winding experiment results. Left columns: v0.5 results, right columns: v1 results.

slopes, we even see that v1's paths actually decrease in length as the effort weight increases to 25 and 100, while being longer than v0.5 at the same effort values for the 10° slope.

A possible explanation could be that the pheromone scoring system enables this behaviour. Each pheromone is scored as follows: (agent's spent effort × effort weight) + (agent's travelled distance × distance weight). When using this formula, the score will obviously be dominated by the effort part if it's weight is set high enough. For low-slope terrain, the high scores motivate agents to move in lower inclines. However, perhaps at high effort weight and high slope, these scores become so large that the score-minimizing behaviour would become to simply interact with the terrain as little as possible. In other words, the agents make shorter paths to receive fewer effort penalties.

TABLE IX: Winding sample counts

Effort weight \ Slope inclination (°)	**0**	**10**	**20**	**30**	**40**	**50**
0	100/100	100/100	100/100	100/100	99/100	100/100
1	100/100	100/100	96/100	89/98	79/94	61/84
5	100/100	100/100	84/97	64/84	60/76	42/51
25	100/100	92/98	71/76	55/59	51/53	39/44
100	100/100	92/91	61/61	43/54	36/43	25/51

Left values: v0.5, right values: v1

If this were the case, however, then an even shorter path would be expected, as we see that v1 is capable of on lower effort weights. Additionally, this behaviour should also occur for v0.5, as it uses the same scoring system.

Thus, it remains unclear what could lead to this behaviour. In any case, both versions clearly still prefer steep ascension in their paths, as we see the (technical) maximum value is the most frequent in all figures 10A-10(E). Furthermore, horizontal movement only increases slightly as incline rises. Both indications combined show that the Winding criteria has not been met.

6) Imperfect: As per the Imperfect requirement, the final path should be no more than 110% of the optimal path's length. From image 11 (v0.5: 99, 90, 6 samples, v1: 99, 100, 99), it is clear that the average path deviation increases dramatically with the distance between nodes. Furthermore, we can see that the path inconsistency also increases with node distance: the standard deviation from the mean ranges from 1.41% to 4.69% for v0.5 and 2.16% to 9.18% for v1. When it comes to path length, v1 has higher average deviation and standard deviation than v0.5. However, with v0.5 having only 6 samples for the path length of 800, it is not possible to confidently tell how different the two versions actually perform at this length.

The increase in overall deviation could very well be the result of the *probabilistic sensor* extension. The increase in randomness of movement does increase exploration, but perhaps at the cost of consistency. Overall, however, the Imperfect requirement is met for unobstructed paths of 400m and shorter, while 800m paths cannot reliably attain 110% or less of perfect path length. It is extremely likely that this trend continues with longer paths.

Figure 12 (v0.5: 99, 50, v1: 99, 93) shows adequate improvement of average deviation between v0.5 and v1. Curiously, standard deviation actually also decreases for the obstacle terrain, in contrast to the path length results mentioned above, from 9.18% to 6.20%. It should be noted that both versions have still somewhat crossed the walled terrain in their generated paths, despite the high effort weight of 1000. On average, v0.5's Obstacle paths added another 15,47% (39m) deviation with vertical path movement, where v1 added 1,39%

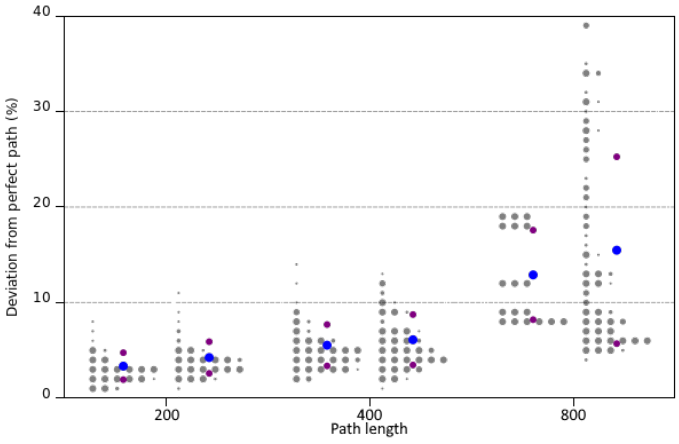

Fig. 11: Imperfect results: influence of node distance.

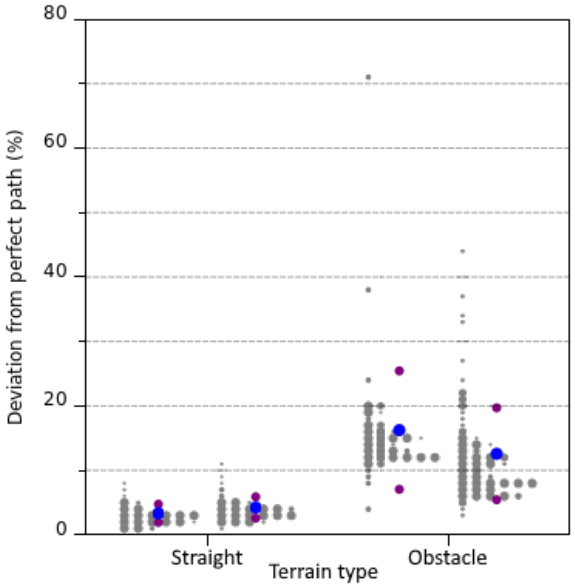

Fig. 12: Imperfect results: influence of curves.

(4m). Keeping in mind the obstacle height of 37.5m, the paths still seem to mostly only graze the walls. For v0.5, this can be attributed to it's difficulty with improving paths, which led to the addition of the *Copy pheromones* extension and is likely the reason for the decreased trespass and decreased average deviation of v1.

Seeing as straight paths maintain a $\leq$10% average deviation up to 400m node distance, one could argue that the Imperfect criteria is met. However, with the vast influence curves seem to have on average deviation and the trend of increasing deviation with node distance, I strongly doubt if the 10% threshold will be maintained in practical applications over non-flat terrain. As such, the Imperfect criteria has been failed.

VII. Conclusion

Below, we will review once more the research questions of this thesis.

For research question 1 (RQ1), the best suitable baseline was to be selected that attained as many of the Heightmap Considerate, Smooth, Large Scale, Joining, Winding and Imperfect requirements as possible as to create realistic walking path networks. As such, various pathfinding algorithms have been covered to see how many requirements they could fulfil. Afterwards, an unusual method was chosen as a baseline, referred to as the 'gathering method'. Though not scientifically based, the method drew parallels to the slime mould method which also fulfilled multiple requirements. The chosen baseline already approximated the Heightmap Considerate, Smooth and Imperfect criteria.

Though deemed the most suitable, the baseline cannot be applied to the circumstances of the experiments. Thus, as a point of comparison, a minimal version of DAPA was

TABLE X: Winding path lengths

Effort weight \ Slope inclination (°)	**0**	**10**	**20**	**30**	**40**	**50**
0	0/0	17.6/17.5	36.4/36.2	57.8/57.4	84.0/83.4	119.2/118.5
	101.9/101.3	102.1/101.3	102.0/101.2	102.2/101.3	102.0/101.3	102.3/101.3
	101.9/101.3	103.6/102.8	108.4/107.5	117.5/116.4	132.2/131.6	157.2/156.0
1	0/0	17.6/17.5	36.4/36.2	57.7/57.4	84.0/83.5	119.9/118.7
	102.0/101.3	102.1/101.5	103.3/102.1	106.3/104.9	116.7/113.4	126.4/123.5
	102.0/101.3	103.6/103.0	109.6/108.4	121.1/119.7	144.9/141.3	176.9/172.5
5	0/0	17.6/17.5	36.4/36.3	57.9/57.6	84.9/83.7	120.5/118.7
	102.0/101.3	103.0/102.2	109.7/111.1	123.4/123.4	141.8/128.7	147.4/130.9
	102.0/101.3	104.5/103.7	115.7/117.0	137.1/136.6	168.2/154.6	194.9/178.4
25	0/0	17.6/17.5	36.6/36.3	58.1/57.6	84.9/84.0	120.4/118.8
	102.1/101.2	106.0/108.4	131.9/124.6	141.6/130.8	145.9/129.3	145.1/132.4
	102.1/101.2	107.5/109.8	137.4/130.0	154.6/143.6	171.7/155.4	192.9/179.9
100	0/0	17.6/17.6	36.7/36.3	58.2/57.6	84.6/83.7	121.0/118.8
	102.2/101.2	111.9/120.4	133.4/129.5	148.7/128.6	153.4/127.9	152.4/130.8
	102.2/101.2	113.4/121.7	138.9/134.8	161.4/141.7	178.3/154.0	199.7/178.7

Averaged results of the Winding experiment. Top values: vertical path length.
Middle: horizontal length. Bottom: full path length.
Left values: v0.5, right values: v1

developed: DAPA v0.5. This allowed me to measure the increase in performance compared to a base version of the approach. To meet all requirements, several extensions were developed. As v0.5 already came relatively close to fulfilling the Heightmap Considerate, Smooth and Imperfect criteria, the extensions focus on improving Large Scale, Joining and Winding characteristics.

For RQ2, the following extensions were chosen to extend the v0.5 version of DAPA to attain the remaining requirements. The Large Scale requirement needed the most attention, as performance scaled quadratically with path length which is a big problem for large world traversal. For this purpose, the *Long-distance pheromones*, *Termination Condition*, *Copy pheromones* and to some extent the *Probabilistic sensor* extension were added. *Long-distance pheromones* enables paths to form over longer distances, the *Termination condition* cuts the simulation short if a quality path emerges early on and *Copy pheromones* increases the likelihood for this to happen by speeding up path improvement. *Probabilistic sensor* changes the way that agents interact with pheromones, resulting in faster iteration time, though at the cost of less consistent agent movement.

Joining was enabled by the addition of *Between-path sharing*, with which agents can interact with pheromones from any path heading to the same goal. This made for some good joining results. However, the behaviour is still not consistent enough, as often times no joining takes place where it could. Furthermore, the extension is not strict enough, as it allows for joined segments to slightly detach from one another at random spots.

The Winding requirement was the hardest one to tackle, as generated paths of the simulation have trouble making significant changes over time. With the *Probabilistic sensor* and *Copy pheromones* extensions, agent exploration and path improvement should have increase respectively, which would lead to winding behaviour in an effort to minimize effort. However, this did not have as much impact with regards to winding as desired.

Lastly, the *Selective node pairing* extension increases control in which paths are to be generated. Though this extension does not contribute to the requirements, it is a valuable tool for any practical application of the technique with networks of more than two nodes.

For RQ3, we want to measure the effectiveness of v1 at fulfilling the criteria as opposed to the baseline (now being v0.5). For this purpose, experiments were set for each requirement, and suitable measurements were determined.
From these experiments, it became clear that both DAPA versions have different desirable Heightmap Considerate behaviour with v0.5 reacting well to low frequency terrain changes while v1 reacts well to high frequency changes. Only a combination of the two would satisfy the requirement, meaning that v1 by itself does not fulfil it adequately.
Furthermore, Smooth has proven to be met, seeing the gradual change in curvature when moving over the path.
Next, while v1 boasts significant performance improvements over v0.5, both in computation time and success rate, the Large Scale requirement of creating 3 simultaneous 1600m paths in 5 minutes could not be met.
Joining was not consistent enough for practical use. Though joining lengths were promising, the deviation in joining length was too strong, thus not attaining its requirement.

Winding experiments show that attempts to increase the winding behaviour were not effective enough as the general incline of the path did not decrease nearly enough as the slope got steeper, failing the requirement.
Finally, the Imperfect experiment showed acceptable results for straight paths with node distances of 400m or less, but deviated too much with longer or curved paths, failing the criteria on larger scales and more practical applications on non-flat terrain.

Overall, most requirements could not be met to satisfaction, with only the Smooth requirement being at a desirable level. Looking back, however, the criteria for this thesis might have been set too high. The amount of augmentation v0.5 needed and still needs to meet them is vast.
For any pathfinding algorithm, map scale has a large performance impact. Considering the nature of my approach, it was too optimistic to set the 5 minute and 2Gb requirement for Large Scale.
It should not be seen as a total failure, however, as significant improvements over v0.5 have been made for all requirements save Heightmap Considerate.

While DAPA is unsuited for navigating mazes or working with impassible obstacles, it remains a novel approach applicable to heightmaps that can create smooth path networks over relatively large distances. Though not as capable as desired, it still fills a niche of pathfinding for large open worlds with some path joining properties.

VIII. Future work

Due to the many near-missed requirements, plenty of future work is available to address these issues.

To start of, a lot of deviation-related differences between v0.5 and v1 are suspected to be the result of the *Probabilistic sensor* extension. It would be interesting if its benefits could be maintained while the increase in deviation is removed by experimenting with its maximum sensor angle and sensor size parameters, or else looking at v1's behaviour when using the original sensor setup. This could have benefits for the Heightmap Considerate, Joining and Imperfect criteria.

For Large Scale improvement, one could have a look at (12) on navigating a subsampled version of the heightmap. While more crude, it could serve as a basis for a second higher-resolution pathfinding pass.
Another performance improvement would be to replace the initial pheromone ellipse mentioned in *Exploration limitation*. The simulation slows down a lot due to the large amount of pheromones added with this ellipse, especially on larger scales. If the ellipse could instead be replaced with an elliptical boundary check in the agent behaviour, then the simulation could speed up significantly while containing agents more effectively then the current approach.
Thirdly, by scaling agents' random movement strength with detected pheromone score, popular paths could be explored more, causing even stronger path improvement behaviour.

Something similar to this pheromone-induced random movement could also be used to approach the Winding requirement, by increasing wandering behaviour based upon strong increases of effort in detected pheromone. This should lead to agents moving side to side along slopes, after which the *Copy pheromones* extension makes sure that other agents notice the improved path alteration.

To prevent small disconnections in joined paths at in Figure 9, a technique can be used that was applied in (11), where close path segments are fused into one, based upon Fréchet distance.

Lastly, to improve path creation success rates on larger paths, multiple final path agents could be released instead of only two. The different agents could have different wandering weights, adding diversity to their traversal over the pheromone map, decreasing the chances of accidentally straying from the main pheromone trail.

IX. Acknowledgment

Big thanks to the Youtuber Pezzza's Work for sharing his insights in the exact workings of the gathering method, his advice in tackling some of the requirements of this thesis and general enthusiasm towards the project.

References

[1] J. Beneš, A. Wilkie, and J. Krivanek, "Procedural modelling of urban road networks," *Computer Graphics Forum*, vol. 33, 02 2014.
[2] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, pp. 100 – 107, 08 1968.
[3] F. Teichteil-Königsbuch and G. Povéda, "Collaborative common path planning in large graphs," 09 2020.
[4] A. Goldberg, H. Kaplan, and R. Werneck, "Reach for a *: Efficient point-to-point shortest path algorithms," 11 2005.
[5] R. Arkin, "Path planning for a vision-based autonomous robot," vol. 727, 01 1986.
[6] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," 01 1991.
[7] L. Wang, J. Kan, J. Guo, and C. Wang, "3d path planning for the ground robot with improved ant colony optimization," *Sensors*, vol. 19, p. 815, 02 2019.
[8] K. Socha, *ACO for Continuous and Mixed-Variable Optimization*, pp. 25–36. Springer, Berlin, Heidelberg, 2004.
[9] J. Jones, "Characteristics of pattern formation and evolution in approximations of physarum transport networks," *Artificial life*, vol. 16, pp. 127–53, 04 2010.
[10] P. Work, "C++ ants simulation 1, first approach," May 2020.
[11] A. Peytavie, E. Galin, E. Guérin, and B. Benes, "Authoring hierarchical road networks," *Computer Graphics Forum*, vol. 30, 05 2011.

[12] E. Galin, A. Peytavie, N. Maréchal, and É. Guérin, "Procedural generation of roads," *Computer Graphics Forum*, vol. 29, 2010.
[13] D. Ferguson and A. Stentz, "Field d*: An interpolation-based path planner and replanner," vol. 28, pp. 239–253, 01 2005.
[14] J. Zhao, D. Cheng, and C. Hao, "An improved ant colony algorithm for solving the path planning problem of the omnidirectional mobile vehicle," *Mathematical Problems in Engineering*, vol. 2016, 01 2016.
[15] S. Lague, "Coding adventure: Ant and slime simulations," May 2021.