APPLIED COMPUTING, MATHEMATICS AND STATISTICS GROUP

Division of Applied Management and Computing

Discrete Lines and Ant Algorithms

Panama Geer, Harry W. McLaughlin and Keith Unsworth

Research Report No: 01/2001
March 2001

ISSN 1174-6696

RESEARCH REPORT

LINCOLN UNIVERSITY
Te Whare Wānaka O Aoraki

Applied Computing, Mathematics and Statistics

The Applied Computing, Mathematics and Statistics Group (ACMS) comprises staff of the Applied Management and Computing Division at Lincoln University whose research and teaching interests are in computing and quantitative disciplines. Previously this group was the academic section of the Centre for Computing and Biometrics at Lincoln University.

The group teaches subjects leading to a Bachelor of Applied Computing degree and a computing major in the Bachelor of Commerce and Management. In addition, it contributes computing, statistics and mathematics subjects to a wide range of other Lincoln University degrees. In particular students can take a computing and mathematics major in the BSc.

The ACMS group is strongly involved in postgraduate teaching leading to honours, masters and PhD degrees. Research interests are in modelling and simulation, applied statistics, end user computing, computer assisted learning, aspects of computer networking, geometric modelling and visualisation.

Research Reports

Every paper appearing in this series has undergone editorial review within the ACMS group. The editorial panel is selected by an editor who is appointed by the Chair of the Applied Management and Computing Division Research Committee.

The views expressed in this paper are not necessarily the same as those held by members of the editorial panel. The accuracy of the information presented in this paper is the sole responsibility of the authors.

This series is a continuation of the series "Centre for Computing and Biometrics Research Report" ISSN 1173-8405.

Copyright

Correspondence

This paper represents work to date and may not necessarily form the basis for the authors' final conclusions relating to this topic. It is likely, however, that the paper will appear in some form in a journal or in conference proceedings in the near future. The authors would be pleased to receive correspondence in connection with any of the issues raised in this paper. Please contact the authors either by email or by writing to the address below.

Any correspondence concerning the series should be sent to:

The Editor
Applied Computing, Mathematics and Statistics Group
Applied Management and Computing Division
PO Box 84
Lincoln University
Canterbury
NEW ZEALAND

Email: computing@lincoln.ac.nz

DISCRETE LINES AND ANT ALGORITHMS

PANAMA GEER
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, NY 12180-3590, USA
geerp@rpi.edu

HARRY W. McLAUGHLIN
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, NY 12180-3590, USA
mclauh@rpi.edu

KEITH UNSWORTH
Applied Computing, Mathematics and Statistics Group
Division of Applied Management and Computing
Lincoln University
Lincoln, Canterbury, New Zealand
unsworth@lincoln.ac.nz

March 26, 2001

This is a report on work in progress. The focus is on the design of an algorithm used to construct discrete lines. It is intended that this is the first step in applying models of complex adaptive systems to more complex geometric constructs. We construct discrete lines using agents (virtual ants). The agents are given very few rules, and otherwise move freely. With this design we allow a particular line to emerge from the movement of the agents rather than model the line first and then display it.

Contents

Figures

Part 1: Introduction

Recently, there has been considerable success in applying models of complex adaptive systems to a diverse range of problems. Examples include the re-routing of computer network traffic, analysis of banking data, and the traveling salesman problem. [2] Each of these problems has been addressed using "swarm intelligence," i.e. algorithms that are based on aspects of the behavior of insects, in this case ants, that live in colonies.

The problems mentioned above are examples of complex problems. Problems fit into the class of complex problems when they are not based on linear relationships. Complex problems typically involve a system of components with nonlinear interactions. In other words, the global results of the system, or the global interactions, amount to more than the sum of the local interactions of elements of the system. Through these nonlinear interactions, *unexpected* behavior results, or emerges (the sum of the of the local interactions is what we refer to as *expected* behavior). If the complex system changes (globally adjusts or reacts to the local interactions) as these nonlinear interactions occur, the system is often referred to as an adaptive system. [8] The term *complex adaptive system* is commonly used to describe these systems. Ant algorithms, along with the more well known genetic algorithms, are examples of algorithms used to address the challenges presented by complex adaptive systems.

An open problem in mathematics is the development of a mathematical model for an adaptive discrete surface. By this, we mean an algorithmic model that generates a surface in such a way that the surface can be easily adjusted. Because swarm-based algorithms are largely directed by local relationships, it is thought that these local relationships would allow for local adjustments. This is really a twofold problem. How does one define a discrete surface and how does one generate a discrete surface? It is hoped that a class of algorithms, which are based on models of complex adaptive systems and are adept at solving problems with nonlinear relationships, will aid in the solution to this problem.

In this report we address two simpler problems. First, how does one define a discrete line? Second, can one use an adaptive algorithm to construct discrete lines? Through looking at the conceptually simpler problem of a discrete line, we hope to lend insight into a way of thinking about evolving geometries like that of a discrete surface generated using models of complex adaptive systems.

A description of what constitutes a discrete line is formulated in this report. Although algorithms do exist for generating discrete representations of continuous lines (e.g. Bresenham's algorithm), there is no consensus on a definition of a discrete line. [5] In defining a discrete line, we set up criteria for measuring the success of an algorithm for generating a discrete line. The definition presented here does not rely on continuous lines, as the lines generated from Bresenham's algorithm do. Our definition relies on simple, local relationships between the cells of the discrete line. In that sense, our definition lends itself to the algorithms described above.[1]

Marco Dorigo *et. al* have had considerable success with algorithms based on the behavior of ant colonies. [3] They report that their ant algorithm solution to the traveling salesman problem (TSP) is comparable to, if not better than, alternative approaches using genetic algorithms and simulated annealing. Encouraged by their success, we investigated the discrete line problem using an ant algorithm approach.

In addition to the success of the work by Dorigo *et. al*, there are particular characteristics of general ant algorithms that support the choice of an algorithm based on the behavior of insect colonies versus choosing another type of algorithm. One of the primary differences, for example, between ant algorithms and genetic algorithms, is in the dynamic versus static behavior of their agents. In genetic algorithms individuals from a population are repeatedly selected, mated (crossover), and mutated. The individuals in the population do not move around investigating the environment in the way that the agents in ant algorithms do. It is this dynamic quality of the agents that initially led us to investigate ant algorithms.

This work proceeds in several parts. Part 2 is devoted to describing what is meant by an ant algorithm. There are several types of ant algorithms that have been used effectively. They are each based on aspects of real ant behavior. One of the ways that natural ants behave is to find the shortest path between locations, for example, a nest and a food source. It is this food foraging behavior that Dorigo *et. al* use to solve the TSP and we will use to generate discrete lines. The work of Dorigo *et. al* on the TSP is briefly presented in Part 2. Part 3, the most comprehensive section, addresses our definition of a discrete line. This definition establishes what it is that

[1] One could argue that a definition of a discrete line is that which is produced by Bresenham's algorithm. However, this definition requires that discrete lines depend on continuous lines, which need not be the case.

makes a collection of cells on a grid appear to be linear. In addition, if two distinct collections of cells appear to be linear, how does one determine which collection of cells constitutes the discrete line, and which does not (or do they both)? Part 4 describes and outlines an algorithm used to find these discrete lines. This algorithm takes inspiration from the work of Dorigo *et. al*. Some conclusions and areas for further study are presented in Part 5.

Part 2: What is an Ant Algorithm?

Optimization algorithms based on ant colony behavior are a recent phenomenon. Marco Dorigo and his collaborators have had success in solving complex optimization problems using agent based algorithms referred to as ant algorithms. Their work using ant algorithms to solve the TSP, provides the basis for our discrete line algorithm. Thus, a brief description of ant algorithms that are based on real ant foraging behavior is presented here. This is followed by a brief description of the work of Dorigo *et. al* [4] using a particular ant algorithm to solve the TSP.[2]

§2.1 A Brief Description

The term "ant algorithm" tends to bring to one's mind many small insects scurrying around busily involved in some process. The name comes from the fact that the general algorithmic procedure was originally based on a very simplified interpretation of the behavior of real ants. Although some ant algorithms are used to model real ant behavior, many are not. In reading the following work, it is helpful to think of ants as a metaphor for understanding the process of ant algorithms. This should be viewed as separate from modeling real ant behavior, which is not the goal in this report. For our purposes, ant algorithms are optimization algorithms which take inspiration from real ant and ant colony behavior.

It is the behavior of an ant colony that provides the power behind an ant algorithm. One can liken this power to that of parallel processing. [9] In order to understand exactly what it is about ant algorithms that makes them so powerful, it is helpful to start with a very simplified description of ant colony behavior. There are many aspects of ant colony behavior that have spawned ant algorithms. These include foraging for food, sorting brood, and co-operative transportation methods.

Our work is based on the process of ants foraging for food. A simplified interpretation of the food foraging process follows. Again, we emphasize that this is an interpretation of the foraging process and our metaphor for understanding the algorithm presented later in this report. Thus, we are not attempting to precisely report on the behavior of real ants.

One can imagine that a colony of ants sends out a portion of its population

[2]The work done in 1996 [4] builds on earlier work from 1991 [3].

from its nest at regular intervals to search for food. The ants disperse from the nest and move randomly across a landscape, each at the same speed. As they move they deposit a substance called pheromone. Once pheromones are deposited they evaporate with time. Eventually one or more of the ants reaches a food source and then returns to the colony. Subsequent ants tend to follow paths with high levels of pheromone. Thus, the trail of pheromone that has evaporated the least will determine the path which is most likely to be followed. These are precisely those paths that are the shortest (i.e. have had the least amount of time to evaporate).

At any given point in the algorithm, the path with the highest pheromone level may not be the true shortest path. This is where the power of colony behavior is demonstrated. After the first group of ants has been sent out, each ant that leaves the colony has several pheromone trails that it could potentially follow. With a relatively large probability, the subsequent ants will follow the shortest path.[3] Those ants that do not follow the shortest path, or only follow it some of time, can potentially find an even shorter path. This path, in turn, will have a higher pheromone level, thus a higher probability of being followed, because the pheromone will have less time to evaporate. The colony behavior allows the ants to reinforce the shortest path to date, while at the same time, due to the probabilistic aspects of their behavior, the colony continues to search for shorter and shorter paths.

§2.2 An Application

Marco Dorigo at the Université Libre de Bruxelles, has done much work with ant algorithms. In particular he and his colleagues have used ant algorithms to solve the traveling salesman problem (TSP). [4] The TSP is a well known problem that for large numbers of cities is complex.

The TSP can be stated as the problem of finding a minimum length closed tour of n cities. Dorigo *et. al* position m ants randomly among the n cities. Over n time steps each ant makes a move to a new city. One can picture the cities as a collection of points, each point connected to every other point by an edge. The choice of a particular ant's move from the current city (point) to another is based on a probability function. This probability function depends on the length of the particular edge connecting the cities (points) and the amount of pheromone present on the particular edge. We can think of the

[3]The actual value of the probability depends on various factors and adjusts frequently.

algorithm as modeling a complex adaptive system. Since the probability function is updated during the running of the algorithm, the probability function allows the algorithm to adapt, and thus model an adaptive system. After the n moves are completed, the shortest path is calculated and the process is repeated. This time in addition to the edge lengths, both the pheromone deposited during the last cycle as well as the evaporation of the existing pheromone is taken into account in an updated probability function. Each repetition of this process is called a cycle. The number of cycles is allowed as user input. Likewise, there are user controlled parameters for the number of ants, the evaporation rate of pheromone, the relative importance within the probability function of the length of the edges, and the relative importance within the probability function of the pheromone level on the edges.[4]

Dorigo *et. al* found that there are certain ranges of these parameters that produce successful results.[5] The ranges of the parameters in the problem correspond to certain characteristics of the optimization procedure. For example low values of the pheromone level, made the procedure similar to a greedy algorithm, while higher values of the pheromone level correspond to stagnation behavior. Stagnation behavior is the situation in which all of the ants make the same tour and thus cease to explore new, potentially shorter, tours.

An important characteristic of ant algorithms is the ability to continuously explore for new, potentially better, solutions. Dorigo *et. al* referred to this as scouting. One way to way to observe scouting within an ant algorithm is to look at the standard deviation of the path lengths after multiple cycles. The path lengths should not all be converging. After each cycle in the TSP, the shortest path was updated. Thus, if a particular cycle yielded a shorter tour than the previous cycle, this new tour was taken to be the new shortest tour. Because it was only necessary to find a true minimum length tour once, scouting was encouraged. Too much scouting however, made the algorithm

[4]More specifically, the parameter representing the relative importance of the length of edges is the reciprocal of the distances between cities. This reciprocal represent the visibility between cities. Because real ants are virtually blind, the use of visibility demonstrates one way that this formulation of an ant algorithm diverges from real ant behavior.

[5]The success of the algorithm was tested against other known methods of solving the TSP, which included a genetic algorithm approach, and a simulated annealing approach. Dorigo and his colleagues report that in each case the effectiveness of the ant algorithm approach was comparable, if not better.

essentially a purely random search. The ant algorithm can be thought of as a directed random search. The probability function and the pheromone level changes, which are updated with each cycle, guide this search.

Part 3: A Definition of a Discrete Line

Our definition of a discrete line proceeds in several parts. In §3.1, we describe an environment for discrete lines. We are working with a specific class of discrete lines which lie in one octant of a square grid and can be denoted by their terminal cell. In §3.2, we discuss several observed characteristics of discrete lines in this environment. These observations are properties that will follow from our definition of a discrete line. These characteristics include connectedness, §3.2.1, cell distribution, §3.2.2, and essential cells, §3.2.3. This is followed by a more detailed description of the essential cells, in the context of the core of a discrete line, §3.3. The core cells are defined and a description of their quantity and location are given. Finally, §3.4 presents criteria for determining whether the discrete line will have the properties mentioned earlier. We simply minimize the lengths of a discrete line. The definition of the length measures used are described in detail. We conclude Part 3 with our definition of a discrete line, §3.5.

It is important to note that the definition of a discrete line is left to §3.5. Thus, our reference to discrete lines in §3.1-§3.4 refers to the eventual goal of the definition in §3.5 and is not meant to imply that the definition is already understood.

§3.1 The Line Environment

A two-dimensional rectangular lattice, with lattice points (vertices) denoted by integer valued coordinate pairs, extends infinitely in every direction. Each square determined by the lattice, with unit side length, is a cell in the square grid. The vertices of these squares are the ordered pairs of integers, mentioned above. Our goal in investigating discrete lines is to characterize what it is about a collection of these cells that makes the collection look like a discrete line. In other words, what makes a collection of these cells appear "straight" to the human eye? When we refer to a discrete line, one can envision a television screen, or a computer display which is composed of pixels. The screen is white when none of pixels are turned on. When any one pixel is turned on, that pixel becomes colored. If the pixels are squares on the grid mentioned above, which ones should be turned on to create an image that appears linear?

In many cases one can easily tell which pixels or cells should be turned on to make the line appear straight. For example a vertical discrete line between

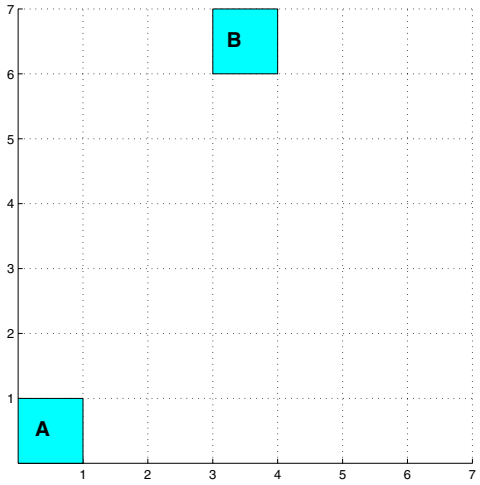


Figure 1: The initial cell, $A(0,0)$, and terminal cell, $B(3,6)$, for a discrete line.

cell $A(0,0)$ and cell $B(0,5)$ is simply a collection of six cells on a square grid forming a column. [Figure 3] The notation $A(0,0)$ can be interpreted as the cell on a square grid, where the ordered pair $(0,0)$ denotes the coordinates of the cell's southwest corner. We adopt the convention of the initial cell, $A(0,0)$, throughout this paper. The cell where the line terminates is denoted $B(e,n)$.[6] The ordered pair (e,n) denotes the coordinates of the southwest corner of the terminal cell, B. [Figure 1] Since the initial cell will always be the same, we will refer to a discrete line by its terminal cell, $B(e,n)$.[7] In the case of $B(3,6)$ it is not as clear which cells should be turned on, as it was in the vertical discrete line case, $B(0,5)$. Should one turn on the cells in the set $\{(1,1), (1,2), (2,3), (2,4), (2,5)\}$ or maybe the cells in the set $\{(0,1), (1,2), (1,3), (2,4), (2,5)\}$? This is where some measure of the straightness, or linearity, of each collection of cells is necessary.

In order to get a feel for what characterizes straightness we examined many collections of cells. We imposed some limits on the problem to initially simplify it. We chose the discrete slope of the collection of cells to be greater than or equal to one. The discrete slope between two cells, $C_1(x_1, y_1)$ and $C_2(x_2, y_2)$, is given by $\frac{y_2 - y_1}{x_2 - x_1}$. The discrete slope for a collection of cells

[6] The choice of e and n refer to easterly and northerly directions.

[7] Note that we use $B(e,n)$ to denote both the discrete line with terminal cell $B(e,n)$ as well as the terminal cell itself.

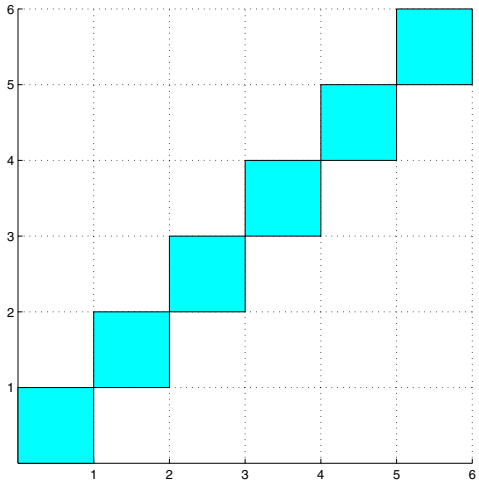


Figure 2: The diagonal discrete line, $B(5,5)$.

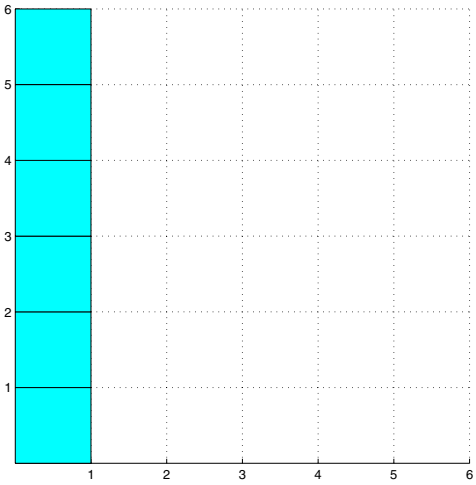


Figure 3: The vertical discrete line, $B(0,5)$.

with initial and terminal cells $A(0,0)$ and $B(e,n)$ respectively, is the discrete slope between cells $A(0,0)$ and $B(e,n)$. Thus we constrained the discrete slope between these cells to be $m = \frac{n}{e} \geq 1$. In other words, the terminal cell will be denoted $B(e,n)$ where $n \geq e$. We assume that characterizing discrete lines in this octant will allow us to extend these ideas to collections of cells with any discrete slope. This has the effect of limiting the environment for a particular collection of cells to an approximate upper triangular region. For example in Figure 1, the triangle with vertices given by the cells $A(0,0)$, $B(6,6)$, and $C(0,6)$ provides a limited region of potential cells for a discrete line from $A(0,0)$ to $B(e,n)$. It follows from what we have stated that any collection of cells which contains a cell in the lower triangular region would not be a candidate for a discrete line. Figures 2, 3, 4, and 5 provide examples of collections of cells that satisfy our later definition of a discrete line.

§3.2 Observations of Collections of Cells

While investigating collections of cells we made numerous observations that differentiated those collections of cells that appeared linear from those that did not. The observations of those collections of cells that appeared linear include

- a connectedness among the cells,

- an evenness in the distribution of cells, and
- the inclusion of some essential cells.

A more detailed discussion of these topics follows.

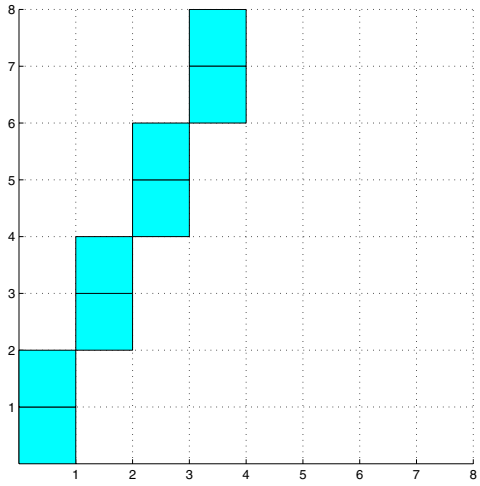


Figure 4: The discrete line, $B(3,7)$.

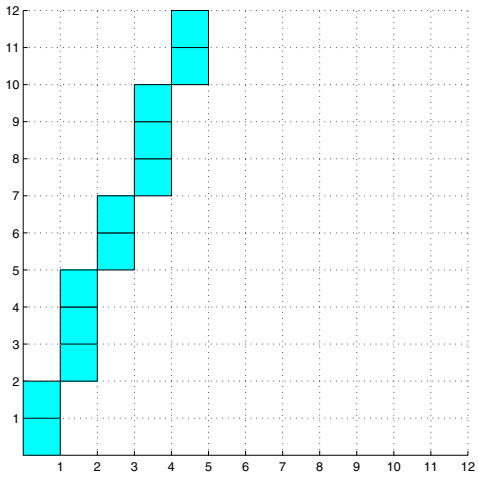


Figure 5: The discrete line, $B(4,11)$.

§3.2.1 Connectedness

It is critical that the discrete line appear connected. It is precisely this connectedness that allows our eyes to differentiate between a discrete line or curve and an arbitrary group of cells on a grid. One can interpret the idea of a connected discrete line as a discrete analogy to continuity for a continuous line or curve. One way to think about a connected set of cells is as follows. A connected set of cells from $A(0,0)$ to $B(e,n)$, denoted $\{C_i |\, i \in \{1,2,...,n+e+1\}\}$, where $A = C_1$ and $B = C_{n+e+1}$, is a set such that every cell, C_i, $1 < i < n+e+1$, shares exactly one edge with cell C_{i+1} and one edge with cell C_{i-1}. It follows from this definition that in the case where $e = n$, the resulting collection of cells is unnecessarily thick. [Figure 6] It is reasonable that a discrete line, with the slope, $m = 1$, between initial and terminal cells, should be the collection of cells $C_i(i-1, i-1)$ for $i \in \{1,2,3,...,n+1\}$, where $A = C_1(0,0)$, and $B = C_{n+1}(n,n)$. Thus, we are looking for a smallest set of connected cells that form a connected discrete line. This requires a slightly different definition of a connected set of cells.

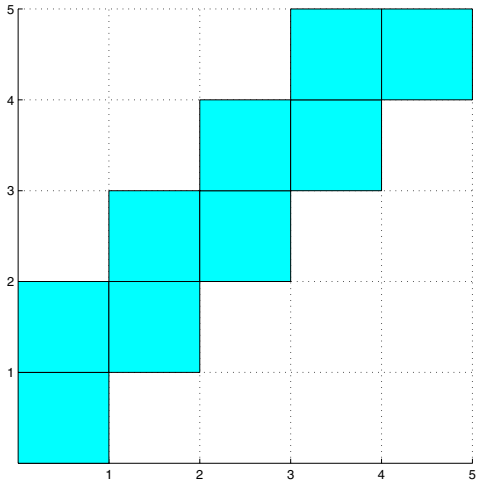


Figure 6: An unnecessarily thick discrete line, $B(4,4)$.

To eliminate these thick, or overly connected, discrete lines, we imposed another constraint on our discrete lines. Our notion of a discrete line from $A(0,0)$ to $B(e,n)$ requires the collection to be connected in such a way as to minimize the number of cells in the collection. The minimum number of cells necessary to allow each cell, other than the initial and terminal cells, to touch exactly two other cells (by an edge or vertex) and still reach $B(e,n)$ is $n+1$. This changes our definition of a connected set of cells.

> A *connected set of cells* from $A(0,0)$ to $B(e,n)$ is a set $\{C_i |\, i \in \{1,2,...,n+1\}\}$, $A = C_1$ and $B = C_{n+1}$, such that each cell C_i, $1 < i < n+1$, shares exactly one horizontal edge or exactly one vertex with cell C_{i-1} and shares exactly one horizontal edge or exactly one vertex with cell C_{i+1}.

The cells depicted in Figures 2-5 fulfill this criteria of connectedness. Thus, our notion of a discrete line requires the collection of cells to be a minimum size set of connected cells.

For a moment we return to ants. Dorigo *et. al* solved the TSP using an ant algorithm. This algorithm sends ants out from various locations and, in some sense, evaluates their movements throughout the process. It is helpful to think of discrete lines in a similar way. Consider starting with some number of ants at cell A and moving them throughout the grid. Since we are only considering discrete lines (collections of cells) with discrete slope ≥ 1 we can

simplify the movement of the ants. At any given cell the ants will only need to move to either the cell that is directly north of their current location (a north move), or to the cell that is directly northeast of their current location (a diagonal move).[8] If we think of ants moving from cell to cell, in a north and diagonal manner (without skipping cells), then the connectedness of the path of the ants, or the connectedness of the discrete line, is assured. The only edges that the connected cells share are horizontal edges. Likewise, the only vertices that the connected cells share are the northeast and southwest vertices.

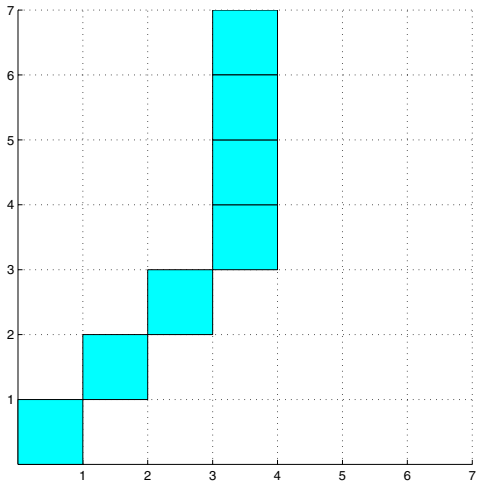


Figure 7: Not a discrete line, $B(3, 6)$.

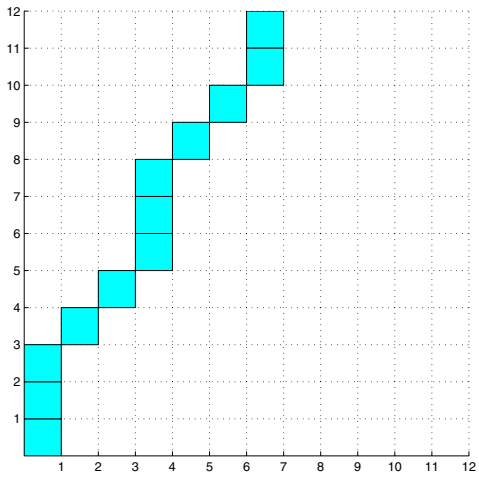


Figure 8: Not a discrete line, $B(6, 11)$.

§3.2.2 Cell Distribution

The concept of connectedness alone is not enough to characterize a discrete line. For example, we suggest that the collections of cells in Figures 4 and 5 each appear as a discrete line, while the collections of cells in Figures 7 and 8 do not. In Figures 7 and 8, there are $n + 1 = 7$ and $n + 1 = 12$ cells, respectively, in each collection. In addition, in both cases the cells are connected either by northeast and southwest vertices or horizontal edges. Thus, Figures 7 and 8 are minimum size, connected, sets of cells. One of

[8] If we had allowed collections of lines like those in Figure 6, then we would have used north and east moves.

the differences between these collections of cells is in their arrangement. One could say that the collections of cells in Figures 4 and 5 are more evenly distributed. There is a symmetry present that makes Figures 4 and 5 appear as discrete lines, a symmetry that is not present in Figures 7 and 8.

What does it mean to *evenly distribute* these cells? If one thinks of the cells in Figures 4 and 5 as appearing in columns, then a pattern arises. The collections of cells that appear to be discrete lines (these are necessarily connected) are composed of columns of cells. We hypothesize that for a given collection of cells these columns can have at most two different heights. If these heights are different, then they differ in height by only one cell. For every discrete line there are $e+1$ columns. Figures 2, 3, and 4 show discrete lines with columns that do not differ in height. Figure 5 shows a discrete line with columns that vary in height between 2 and 3 cells.

> The *column property of a discrete line* states that for a given collection of $e+1$ cells, the heights of the columns of cells, h_1 and h_2, satisfy $|h_1 - h_2| \leq 1$.

The column property is an important characteristic of discrete lines. It requires the columns to be as close as possible in height. If they were not as close as possible in height, then the collection of cells would appear curved, thus not the discrete lines we are looking for. [Figures 7 and 8] However, the column property and the connectedness of the set of cells are still not enough to ensure a discrete line. There are connected sets of cells with the column property that do not appear linear, see Figures 9 versus Figure 10.

Beyond this idea of at most two columns of two sizes, that differ in height by at most one cell, we also consider the idea of arranging these columns of cells. Of course this is only an issue when there are two different column heights. This amounts to the problem of arranging two collections of distinct objects (the shorter columns and the longer columns) as evenly as possible.

What do we mean by *as evenly as possible*? One can represent each column in Figure 9 by a number. If a column height is the shorter of the two heights, it is represented by a zero, otherwise it is represented by a one. Employing this representation we can write the collections of cells of Figures 9 and 10 as the strings 01100 and 01010 respectively. We consider the 1s and 0s associated with the string of Figure 10 to be more evenly distributed than those of Figure 9. The precise criteria for why this is so, remains to be stated.

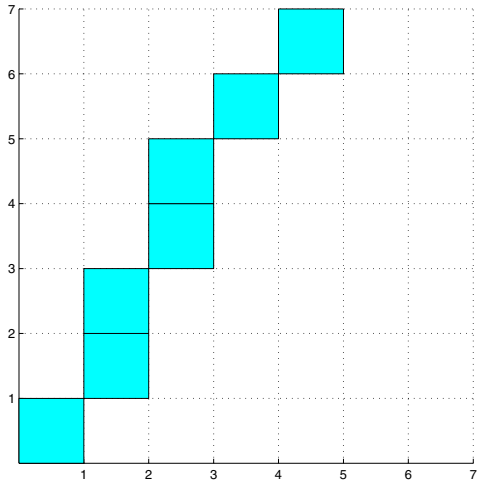


Figure 9: Not a discrete line, $B(4,6)$. This can be represented by 01100 or D N D N DD.

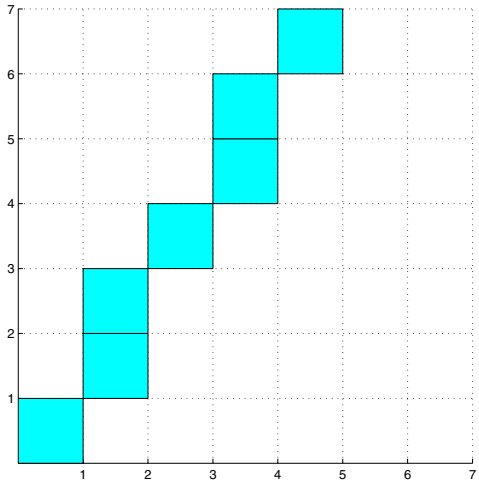


Figure 10: The discrete line, $B(4,6)$. This can be represented by 01010 or D N DD N D.

One can also dynamically generate a connected set of cells. As mentioned earlier, this idea of movement is appealing when considering ants foraging for food.[9] In Figures 9 and 10 the first move from cell A to the next cell is a diagonal move (D). The following move in each figure is a north move (N). Continuing this process the string representations of Figures 9 and 10 are D N D N DD and D N DD N D, respectively. Again, we consider the string of Ds and Ns associated with Figure 10 to be more evenly distributed than that of Figure 9 (at this point one can consider that the string simply "appears to the eye" to be more evenly distributed).

The implementation of our ant algorithm uses string representations composed of Ds and Ns rather than 1s and 0s. In evenly distributing the Ds and Ns, our algorithm evenly distributes the moves rather than the cells themselves. For a given terminal cell, $B(e,n)$, if one compares the collections of cells generated by distributing moves and those generated by distributing cells, there are cases in which different collections of cells result. [Figures 11 and 12] Often when the characters are evenly distributed, the two representations (01-strings versus DN-strings) yield the same collection of cells.

[9] One of the differences between genetic algorithms and ant algorithms is that ant algorithms use a metaphor for ant movement, whereas genetic algorithms are more static in their use of selection, crossover, and mutation. This concept is also discussed in Part 1.

However, there are some terminal cells for which evenly distributing 0s and 1s in a 01-string representation of the collection of cells yields a collection of cells that we consider more visually appealing than the collection produced using a DN-string representation. Thus, it is important to note that our algorithm uses only DN-string representations of collections of cells.

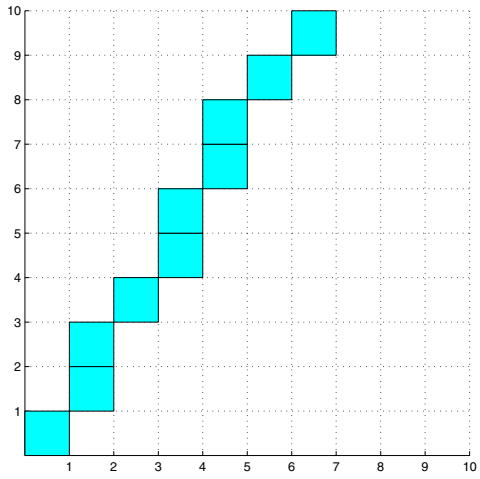


Figure 11: The sequence of Ds and Ns for $B(6,9)$, that appears most evenly distributed is D N DD N D N DD. The sequence of 0s and 1s for 0101100, does not appear to be the most evenly distributed.

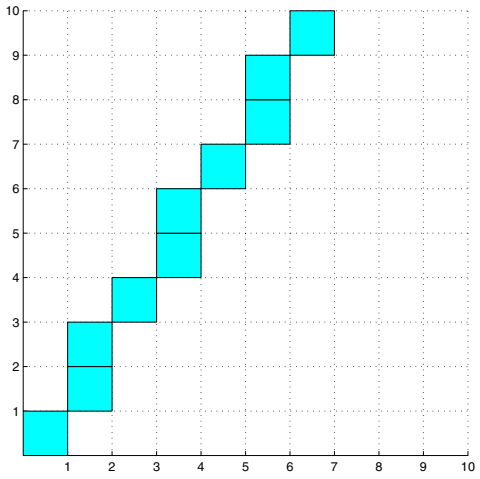


Figure 12: The sequence of 0s and 1s for $B(6,9)$ that appears the most evenly distributed is 0101010. The sequence of Ns and Ds, D N DD N DD N D, does not appear to be the most evenly distributed.

The question remains. How does one characterize this notion of even distribution? What is the measure for the evenness of a distribution of moves or cells? Since the goal of this section is simply to state our observations, the method developed to measure this concept of even distributions of two distinct entities will be discussed in depth in §3.4.

§3.2.3 Essential Cells

There is one further observation that is worth noting. In looking for a characterization of a discrete line one can look for discrete analogies to some of the properties of continuous lines. We are not convinced that there are discrete analogies for all (or maybe even many) of the properties of a continuous line. Earlier we discussed a discrete analogy to continuity in the case of a continuous line. After defining a discrete line, we would like to be able to

both subdivide a discrete line into discrete line segments and extend a discrete line to a "longer"discrete line. How can this be accomplished? These are still open questions. In the process of thinking about them, one encounters the notion of essential cells. Clearly cells $A(0,0)$ and $B(e,n)$ are always in the discrete line, and as such are deemed essential cells. Are there other essential cells? It turns out that there are. We refer to these cells as core cells.[10] We will see that algorithm presented later automatically includes the essential cells. These cells appear in predictable quantities that depend on the terminal cell location, $B(e,n)$. Not only do they appear in predictable quantities, they appear in predictable locations. Both the quantity of core cells as well as their location, suggest potential for extending and subdividing the discrete lines.

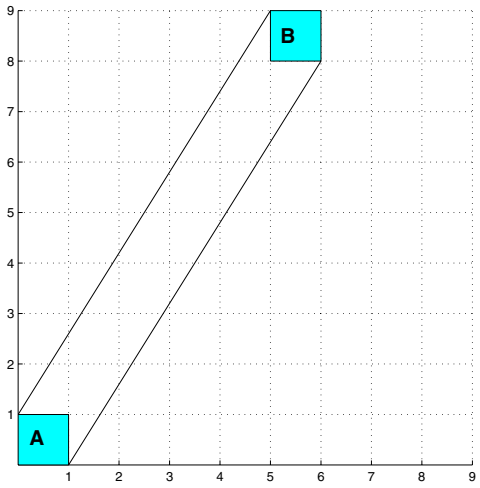


Figure 13: The channel lines for $B(5,8)$.

§3.3 The Core

The initial cell, $A(0,0)$, and the terminal cell, $B(e,n)$, are examples of essential cells in a discrete line. These are also examples of core cells. Given a discrete line, as defined later, the collection of core cells is referred to as the

[10]There are two cases in which the set of essential cells is actually a subset of the set of core cells for a given discrete line. In other words, all essential cells are core cells, however, the converse is not true. All core cells are not essential cells. This has to do with the concept of connectedness discussed earlier and will be explained in more detail in §3.3.

core of the discrete line. In observing many collections of cells that appear as discrete lines, we found that there are essential cells other than just the initial and terminal cells. To have an accurate definition of a discrete line, the discrete line should certainly include these other essential cells. This is where the core cells are incorporated into the definition. To define the core of a discrete line we first describe two other features of a discrete line, the channel lines and the hull. Next, we define the core of the discrete line and proceed to describe algorithms for determining the number of cells in the core, and the locations of these cells.

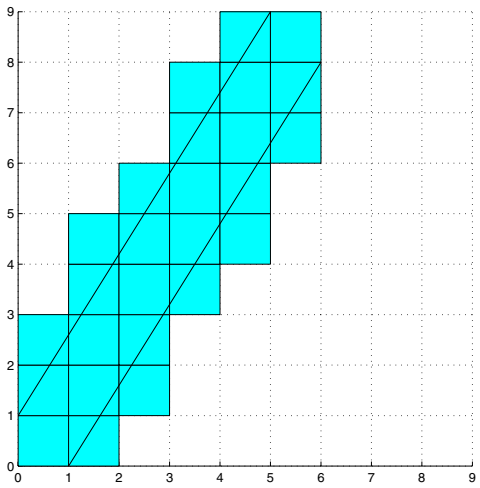


Figure 14: The shaded cells are the hull for $B(5,8)$. Each individual cell in the hull is referred to as a hull cell.

For a particular discrete line, the channel lines along with the initial and terminal cells define a region in the plane that contains all the continuous lines from a point in the initial cell, A, to another point in the terminal cell, B. If the terminal cell is $B(e,n)$, then the region is the interior and boundary of the six sided polygon with vertices at $(0,0)$, $(1,0)$, $(e+1,n)$, $(e+1,n+1)$, $(e,n+1)$, and $(0,1)$. Every discrete line has two channel lines, which are the continuous lines joining certain vertices of the initial and terminal cells of the discrete line. Specifically for $B(5,8)$, the channel lines are the continuous line connecting the vertex $(0,1)$ to the vertex $(5,9)$ and the continuous line connecting the vertex $(1,0)$ to the vertex $(6,8)$ on the square grid. [Figures 13 and 14] The region bounded by the channel lines

and the initial and terminal cells for $B(5,8)$ forms the six sided polygon with vertices $(0,0)$, $(1,0)$, $(6,8)$, $(6,9)$, $(5,9)$, and $(0,1)$.

> The *channel lines* can be defined as the continuous lines ℓ_1 and ℓ_2, such that ℓ_1 connects the northwest vertex of cell A, denoted by the ordered pair (0,1) to the northwest vertex of cell B, denoted by the ordered pair $(e, n+1)$, and ℓ_2 connects the southeast vertex of cell A, denoted by the ordered pair (1,0) and the southeast vertex of cell B, denoted by the ordered pair $(e+1, n)$.

The hull is closely related to the channel lines. The hull of a discrete line from $A(0,0)$ to $B(e,n)$ is a collection of cells, called hull cells, on the square grid.

> A cell is a hull cell if it intersects the interior of the region bounded by the channel lines and the initial and terminal cells (the six sided polygon). The collection of hull cells is referred to as the *hull of the discrete line* from A to B.

The shaded region in Figure 14 is the hull for $B(5,8)$. Note that the initial and terminal cells are hull cells. A discrete line is a subset of its hull. In the case of a vertical discrete line, the hull from $A(0,0)$ to $B(e,n)$ is exactly the discrete line, $B(e,n)$, we define later.

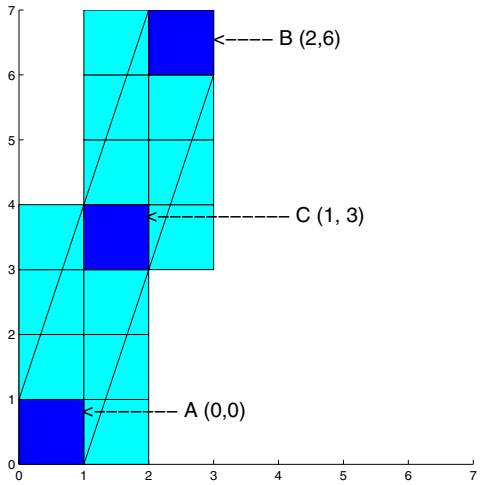


Figure 15: The hull, core and channel lines for $B(2,6)$.

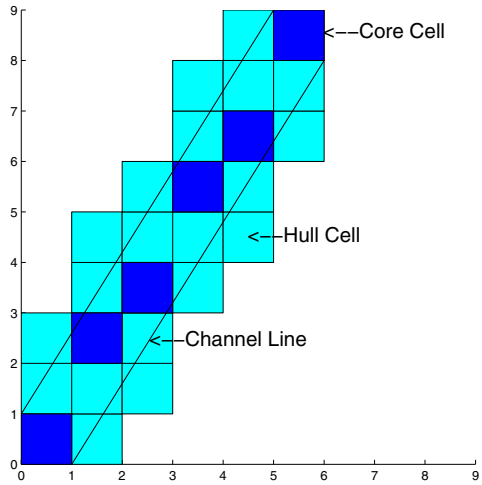


Figure 16: The hull, core and channel lines for $B(5,8)$.

The hull plays an important role in understanding and defining core cells. This is most easily seen by posing a question. For a given discrete line, is there a hull cell, C (other than the initial and terminal cells), for which the hull of the discrete line from A to C and the hull of the discrete line from C to B are both subsets of the hull of the discrete line from A to B? The answer is yes. We refer to a cell C that has this property as a core cell. [Figures 15 and 16] The core of a discrete line is the collection of core cells for that discrete line.[11]

> The *core of a discrete line* is a set of cells, C_i, $i = 1...c(e,n)$, such that for each C_i, the hull from A to C_i and the hull from C_i to B are both subsets of the hull from A to B. We define the initial cell, A, and terminal cell, B, to be core cells. The value $c(e,n)$ is the number of cells in the core.

In the case of the vertical discrete line denoted by $B(0,n)$, the hull, the core, and the discrete line itself all coincide. There are many cases in which the discrete line and the core coincide, however, the vertical discrete line is the only case in which all three coincide.

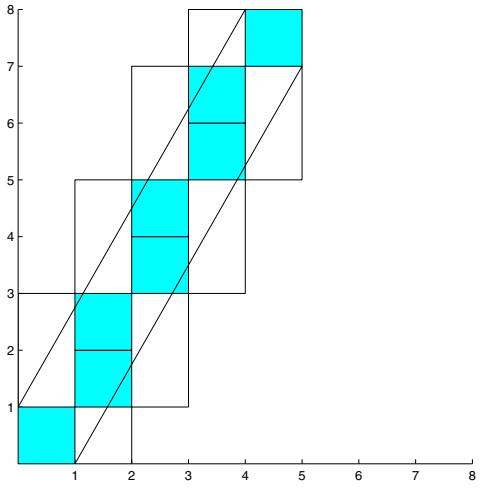


Figure 17: For the discrete line $B(4,7)$, the core and the discrete line coincide.

Earlier the idea of the essential cells was mentioned. If the core of a discrete line is not overly connected, as described in §3.2.1, then the discrete

[11] Note that the core of a discrete line necessarily includes the initial and terminal cells for that discrete line.

line must contain all cells in the core, and thus these cells are essential cells.[12] In fact, the discrete line is exactly the core when the number of cells in the core is $n+1$. [Figure 17] This will be discussed further after methods for finding the number and location of the core cells have been discussed. The number of cells in the core can be established as follows.

> For a discrete line connecting cell $A(0,0)$ and cell $B(e,n)$, let $c(e,n)$ denote the number of cells in the core.
>
> - If $e=0$, then $c(e,n)=n+1$.
> - If $e=1$, then $c(e,n)=2(n+1)$. [13]
> - If $e=n-1$, then $c(e,n)=2n$. [14]
>
> After establishing the above cases, the number of cells in the core can be generated for any $B(e,n)$ by recursively calling 1 and 2 below.
>
> 1. If $2e>n$, then $c(e,n)=c(n-e,n)$.
> 2. If $2e\leq n$, then $c(e,n)=c(e,e+r)$, where $r=n \bmod e$.

A consequence of this is that if n and e are not relatively prime and are not one of the first four cases above, then the number of cells in the core is $k+1$, where k is the greatest common factor of n and e. It follows that, if the number of cells in the core is odd, then the greatest common factor of the n and e is a multiple of 2.

To specify the location of the core cells we look at a pattern in the relationship between the core cells. Figure 16 shows the core of the line $B(5,8)$ as the darkest cells. Notice that the core cells form two groups. The first group includes the initial cell, $A(0,0)$, as well as the cells $A_1(2,3)$ and $A_2(4,6)$. The second group includes the terminal cell, $B(6,8)$, as well as the cells $B_1(3,5)$ and $B_2(1,2)$. Recall that the discrete slope between two cells, $C_1(x_1,y_1)$ and $C_2(x_2,y_2)$, is given by $\frac{y_2-y_1}{x_2-x_1}$. Notice that the discrete slope between A and A_1 is $\frac{3}{2}$, and the discrete slope between A_1 and A_2 is $\frac{3}{2}$. Likewise notice that the discrete slope between B_2 and B_1 is $\frac{3}{2}$, and the discrete slope between B_1 and B is $\frac{3}{2}$. Each of these groups of cells are subsets of the core. We

[12] The core cells are only overly connected in two cases, when $e=1$ or $e=n-1$.

[13] The core is overly connected, that is, $2(n+1)>n+1$, $\forall n\geq 0$.

[14] The core is overly connected, that is, $2n>n+1$,$\forall n>1$.

will refer to them as the A-set and the B-set respectively. If the A-set also contains the terminal cell, B, then there is only one subset of the core. In other words the A-set and B-set are equal.

If we find the discrete slope between any two cells in either the A-set or the B-set, we can specify the location of all of the core cells. How do we find the discrete slopes between two cells in one of these two sets? The following procedure describes the method for finding the locations of the core cells.

> For a discrete line connecting cell $A(0,0)$ and cell $B(e,n)$, $e > 1$, let $c(e,n)$ = the number of cells in the core, and let i, z, p and q be positive integers.[15] If the number of cells in the core is even, then $c(e,n) = 2z$, otherwise $c(e,n) = 2z+1$.
>
> - If n and e are not relatively prime, then, for i from 1 to $c(e,n)$, the location of the core cells is given by
>
> $$(qi, pi),$$
>
> where $\frac{p}{q} = \frac{n}{e}$, and p and q are relatively prime.
>
> - If n and e are relatively prime, then, for i from 1 to z, the locations of the core cells are given by,
>
> $$(\widetilde{\Delta x}(i-1), \widetilde{\Delta y}(i-1)), \quad \text{and}$$
>
> $$(e - \widetilde{\Delta x}(i-1), n - \widetilde{\Delta y}(i-1)),$$
>
> where $\widetilde{\Delta x}$ is an integer such that $\Delta x = \frac{e+1}{z}$ and $|\widetilde{\Delta x} - \Delta x| < 0.5$ or $\widetilde{\Delta x} - \Delta x = 0.5$ and $\widetilde{\Delta y}$ is an integer such that $\Delta y = (\frac{n}{e})\Delta x$ and $|\widetilde{\Delta y} - \Delta y| < 0.5$ or $\widetilde{\Delta y} - \Delta y = 0.5$.[16]

In the first case, when n and e share a positive integer divisor other than 1, the discrete slope between any two cells in one of the two sets (either the A-set or the B-set) is given by $\frac{p}{q}$. In the second case, when n and e are

[15] The cases when $e = 0$ and $e = 1$ are relatively straightforward. If $e = 0$, then the core cells form a single column of $n+1$ cells from $A(0,0)$ to $B(0,n)$. If $e = 1$, then the core cells form two columns, one from $A(0,0)$ to cell $(0,n)$ and another from cell $(1,0)$ to $B(1,n)$. Notice that when $e = 1$, the core cells are overly connected.

[16] It is worth noting that the equation $\Delta y = (\frac{n}{e})\Delta x$ is consistent with the continuous equation for a line through the origin.

relatively prime, the discrete slope between two cells in one of these sets is given by $\frac{\widetilde{\Delta y}}{\Delta x}$.

It was mentioned earlier that all essential cells are also core cells of the discrete line.[17] In initially investigating the core of a discrete line we were looking for discrete analogies to some of the properties of a continuous line. Although we have not crystallized these yet, we do feel that the connection between the core of a discrete line and the idea of essential cells is an important one. The following statements support this.

> If the core of $B(e, n)$ is not overly connected, i. e. $e \neq 1$ and $e \neq n - 1$, then the core is a subset of the discrete line. [Figures 15, 16, and 18]

> If $c(e, n) = n + 1$, then the core of a discrete line, is the discrete line. [Figures 2, 3, 4, 17, and 19]

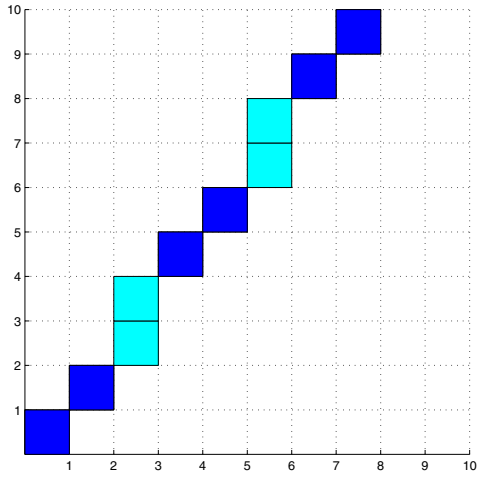


Figure 18: For $B(7, 9)$ the core is a subset of the discrete line. The discrete line is all colored cells, while the core is the darkest of these cells.

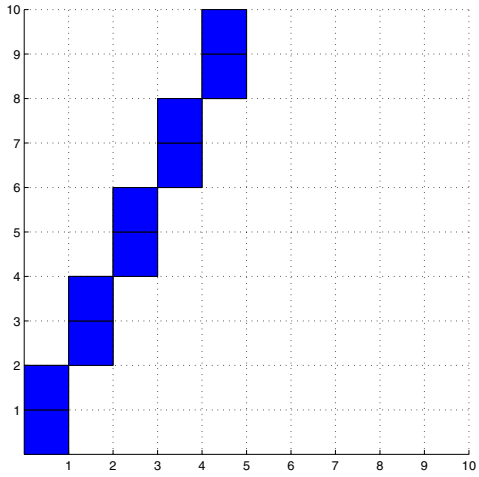


Figure 19: For $B(4, 9)$, the core and the discrete line coincide.

[17] See §3.2.3, Essential Cells.

§3.4 The Rules of Arrangement

We have observed several characteristics of a discrete line. These observations have given us a good idea of what features should follow from our definition of a discrete line. We have already established how to attain some of these characteristics. Now, the question is how do we ensure that *all* of these features are satisfied? Our observations can be summarized as follows.

1. A discrete line is composed of a collection of connected, but not overly connected, cells, §3.2.1.
2. A discrete line is composed of columns of cells, such that these columns differ in height by no more than one cell (column property).
3. The columns of the discrete line must be arranged so as to appear as evenly distributed as possible.
4. If the core of a discrete line is not overly connected, i.e. $e \neq 1$ and $e \neq n-1$, then the discrete line contains its core.
5. If $c(e,n) = n+1$, then the discrete line is completely defined by its core.

In §3.2.1 we ensured the connectedness of the discrete line by writing the line as a string of Ds and Ns and interpreting these Ds and Ns as moves along the square grid in the diagonal and northerly directions respectively. It is a bit more challenging to ensure that the other characteristics are satisfied. We begin with observations 2 and 3, the column property and the even distribution of the columns. We find that through establishing these observations 2 and 3, observations 4 and 5 follow directly. In other words, by ensuring the column property and the even distribution of the columns, for $e \neq 1$ and $e \neq n-1$, it follows that the core is a subset of the discrete line.

> The column property and the even distribution of these columns guarantee that a discrete line will contain the essential cells.

The column property and the even distribution of columns are achieved by minimizing lengths. These lengths can be described as one length measured on different scales. We will refer to the length measured on the first

scale as the first order length, the length measured on the second scale as the second order length, and so on. This notion of length requires that the discrete line $B(e, n)$ is represented as a string. Our algorithm measures the length of the string composed of just the characters D and N.[18] As before, the Ds represent diagonal moves and the Ns represent north moves. A connected collection of cells from $A(0, 0)$ to $B(e, n)$ can be represented as a string of e, Ds and $n - e$, Ns. Thus, for example, connected collections of cells for $B(7, 9)$ can be written as any combination of 7 Ds and 2 Ns. Examples 1-6 show some combinations and Figures 20-25 show the collections depicted on a two dimensional rectangular lattice. The spaces between adjacent Ns and Ds are included for clarity. Four of these examples, specifically those without adjacent Ns (NNs), satisfy the column property. However, only one of these strings, Example 6a, represents a collection of cells that has evenly distributed columns of cells. This in turn corresponds to the string which we suggest has the most even distribution of Ds and Ns.

Examples 1a-6a show strings that represent connected collections of cells where the terminal cell is $B(7, 9)$.

Example 1a:	DDDDDDD NN	**Example 2a**:	N DDDDDDD N
Example 3a:	DDD NN DDDD	**Example 4a**:	D N DDDDD N D
Example 5a:	DD N DD N DDD	**Example 6a**:	DD N DDD N DD

Before the details of the lengths are given, it helps to understand their role. The first order length quantifies the number of adjacent letters that are the same. We minimize the first order length, which has the effect of minimizing the number of adjacent letters that are the same. This in turn has the effect of establishing the column property.[19] In Examples 1a-4a the first order length would not be minimized because the number of adjacent

[18] It is noted that the length measure described here, with slight modifications, would also apply to other string representations of a discrete line. In fact, there are certain cases in which the representation of the line by 0s and 1s, as mentioned in §3.2.2, produces evenly distributed collections of cells that are not available when Ds and Ns are used.

[19] If a collection of cells has minimum first order length, then the collection satisfies the column property. However, the converse is not true. There exist strings that represent collections of cells that satisfy the column property, that do not have the minimum first order length. [Figures 21 and 23]

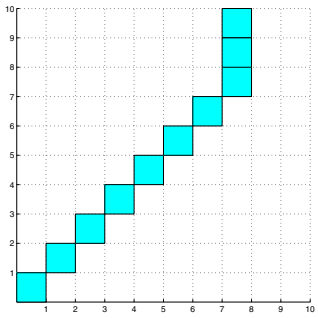


Figure 20: Example 1a, DDDDDDDNN.

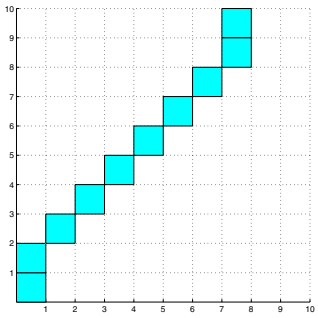


Figure 21: Example 2a, NDDDDDDDN.

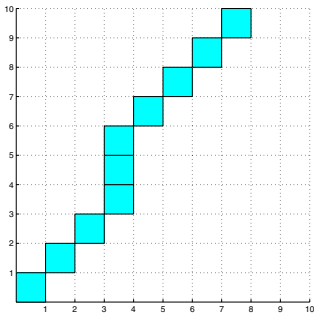


Figure 22: Example 3a, DDDNNDDDD.

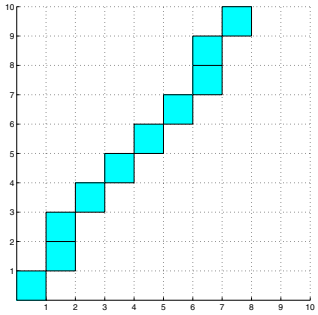


Figure 23: Example 4a, DNDDDDDND.

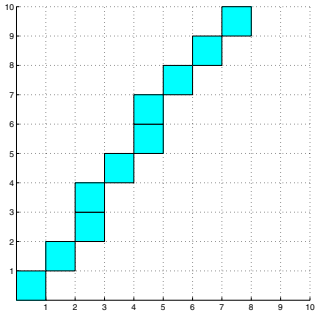


Figure 24: Example 5a, DDNDDNDDD.

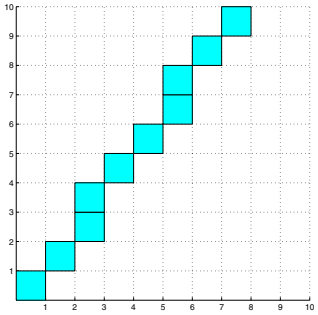


Figure 25: Example 6a, DDNDDDNDD.

letters that are the same is not minimized. Whereas, in examples 5a and 6a, the number of adjacent letters that are the same is minimized. The second order length quantifies the number of adjacent substrings of letters that are the same.[20] As with the first order length, we minimize the second order length. This has the effect of minimizing the number of adjacent substrings of letters that are the same. Example 5a does not have the minimum number of adjacent substrings of letters that are the same, whereas 6a does minimize the number of adjacent substrings of letters that are the same. This in turn has the effect of establishing the first level of even distribution. Minimizing

[20]The term *substrings* here refers to clusters of letters that are the same. For example, single letters (N or D), double letters (NN or DD), triple letters (NNN or DDD), etc. are all examples of substrings. Two substrings are *adjacent* if they are separated only by one different letter substring. For example in NNDDDNN, the double N substrings are adjacent because they are separated by only one triple D substring.

the third order length minimizes the number of adjacent groups of substrings of letters, and so on. These ideas will be discussed further with examples.

We begin by defining the first order length. The number of characters in a string for $B(e,n)$ is n. Each character in the string, denoted $char_i$ for $i = 1, 2, ..., n$, has an associated number. These numbers, denoted num_i for $i = 1, 2, ..., n$, are determined as follows.

Let $num_1 = 1$.
For each $i \neq 1$,

- if the letter in the string for $char_i$ is the same as the letter in the string for $char_{i-1}$, then $num_i = num_{i-1} + 1$,
- otherwise $num_i = 1$.

The first order length, L_1, can be defined as follows:

$$L_1 = \sum_{i=1}^{n} num_i.$$

Examples 1b-6b show the value of num_i for each $char_i$ and the first order length L_1.

Example 1b: DDDDDDDNN
1 2 3 4 5 6 7 1 2
$L_1 = 31$

Example 2b: NDDDDDDDN
1 1 2 3 4 5 6 7 1
$L_1 = 30$

Example 3b: DDDNNDDDD
1 2 3 1 2 1 2 3 4
$L_1 = 19$

Example 4b: DNDDDDDND
1 1 1 2 3 4 5 1 1
$L_1 = 20$

Example 5b: DDNDDNDDD
1 2 1 1 2 1 1 2 3
$L_1 = 14$

Example 6b: DDNDDDNDD
1 2 1 1 2 3 1 1 2
$L_1 = 14$

Notice that Examples 5b and 6b have the same first order length. In addition this is the minimum first order length possible for the string of 5 Ds and 2 Ns. One may ask how we know that this is the minimum first

order length possible since we have only examined six strings of Ds and Ns. The minimum first order length string representing a collection of cells with terminal cell $B(e, n)$ can be found as follows.

Let $minL_1$ = the minimum first order length.

Case 1: If $n = 2e$, then $minL_1 = 2e = n$.

Case 2: If $n \neq 2e$, let $M = \max(e, n-e)$, $m = \min(e, n-e)$, $q = \left\lfloor \frac{M}{m+1} \right\rfloor$, and $r = M \bmod (m+1)$, then [21]

$$minL_1 = m + (m+1)\sum_{i=1}^{q}(i) + (q+1)r.$$

Thus for the string with $B(7, 9)$, the minimized first order length, $minL_1$, is given by

$$minL_1 = 2 + (2+1)\sum_{i=1}^{2}(i) + (2+1)(1) = 14.$$

From Examples 5b and 6b, we see that minimizing first order length does not produce a unique string (or collection of cells). As mentioned before, when the first order length is minimized, the column property is satisfied. However, the first order length does not ensure an even distribution of the columns. One can say that the second order length is used to determine which strings represent a first level of evenly distributed columns. In this particular case, the second order length can be used to find the unique discrete line, thus there is no need to look to a higher order measure of length. In the event that the second order length does not produce a unique discrete line, we would move to the third order length, and, if necessary, continue to higher order lengths from there.

As mentioned earlier, the second order length quantifies the number of adjacent substrings of letters that are the same. Substrings of letters that are the same refer to, in the case of Ds, single Ds, double Ds (DDs), triple Ds (DDDs), etc., likewise with Ns. Two substrings are adjacent if they are separated only by a different letter substring. Because the second order length is measured after the first order length has been minimized, it is only necessary to consider substrings of letters that appear most frequently in the string.

[21] Note that $q = \left\lfloor \frac{M}{m+1} \right\rfloor$ is the greatest integer less than or equal to $\frac{M}{m+1}$.

Thus, if $n - e > e$, the second order length will measure the substrings of Ns and if $e > n - e$, the second order length will measure the substrings of Ds. If $e = n - e$ then the Ds and Ns alternate and the second order length is not used because the strings found by minimizing first order length are considered equivalent.[22]

Examples 5c and 6c show substrings grouped with braces.

Example 5c: $\underbrace{\text{DD}}\ \text{N}\ \underbrace{\text{DD}}\ \text{N}\ \underbrace{\text{DDD}}$

Example 6c: $\underbrace{\text{DD}}\ \text{N}\ \underbrace{\text{DDD}}\ \text{N}\ \underbrace{\text{DD}}$

Notice that in Example 5c, there are two adjacent DDs, however in example 6c, there are no adjacent substrings of the same number of Ds. Again, we do not consider the adjacent substrings of Ns because the total number of Ds is greater than the total number of Ns, thus the distribution of the Ns was established in the first order length.

The second order length is defined in much the same way as the first order length. For a collection of cells with terminal cell $B(e, n)$, that has more than one string with minimum first order length, there will be two distinct substrings of the same letters that need to be distributed evenly. These substrings will differ by only one additional letter, in the same way that the column of cells differ by only one cell in height. In Examples 5c and 6c, these substrings are the DDs and DDDs. The number of each of these substrings is $m + 1 - r$ and r respectively, where m and r are as defined earlier. The number of characters in each of these substrings are q (there are $m+1-r$ of these) and $q+1$ (there are r of these), again where q is as defined earlier. These distinct substrings play the role for the second order length that the distinct letters played in the first order length. There is a total of $m + 1$ substrings and each will be denoted s_i for $i = 1, 2, ..., m + 1$. Each substring has an associated number, denoted num_i for $i = 1, 2, ..., m + 1$ and these numbers are determined as follows.

Let $num_1 = 1$.
For each $i \neq 1$,

- if s_i is the same substring as s_{i-1}, then $num_i = num_{i-1} + 1$,

[22]For example, DNDN is considered the same as NDND.

- otherwise $num_i = 1$.

The second order length, L_2, can be defined as follows:

$$L_2 = \sum_{i=1}^{m+1} num_i.$$

Examples 5d and 6d show calculations of the second order length.

Example 5d: $\underbrace{\text{DD}}_{1}\ \text{N}\ \underbrace{\text{DD}}_{2}\ \text{N}\ \underbrace{\text{DDD}}_{1} = 4 = L_2$

Example 6d: $\underbrace{\text{DD}}_{1}\ \text{N}\ \underbrace{\text{DDD}}_{1}\ \text{N}\ \underbrace{\text{DD}}_{1} = 3 = L_2$

In the case of $B(7, 9)$, the second order length isolates the unique string with the most evenly distributed columns of cells.[23] There are cases in which the second order length will not find a unique string with the most evenly distributed columns of cells [Examples 7a and 8a]. In this case it is necessary to extend this procedure to third order length. In the case of third order length, distinct groups of substrings play the same role as the distinct substrings played in the second order length and the distinct letters played in the first order length. Higher order lengths can be minimized until a unique evenly distributed string is found or the strings with minimized lengths are deemed equivalent.

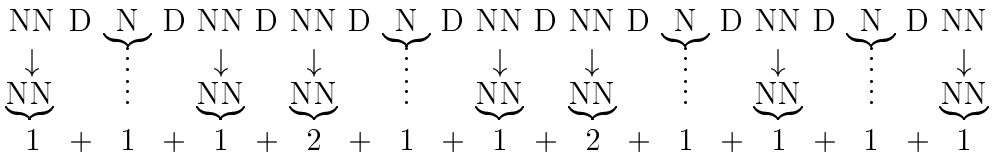
In the case where $e = n - e$ or $n = 2e$, two equivalent strings result from minimizing first order length. The strings have the same number of Ds as Ns. The characters in the string with minimum first order length alternate between Ds and Ns, for example, $B(2, 4)$, DNDN and NDND. These two strings are considered equivalent because one is simply the reverse of the other. The same equivalence occurs with higher order lengths. For second order length, the string D N DD N D N DD is equivalent to DD N D N DD N D. This equivalence occurs precisely when the items being arranged occur in the same quantity. In the first case the items being arranged were the characters D and N, and they occur in the same quantity when $n = 2e$. In the second case the items being arranged were the substrings DD and D. Substrings, of any

[23] The procedure for determining the value of the minimum possible second order length is similar to that shown for the minimum possible first order length.

length, occur in the same quantity when $m+1-r=r$ or $m=2r-1$, where m and r are defined as before.

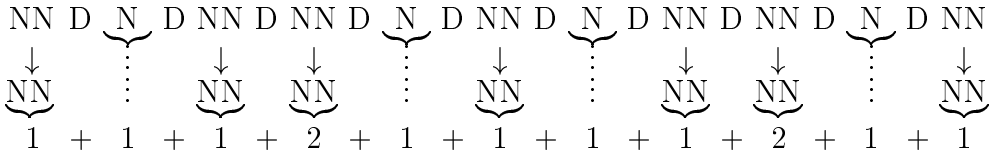
Examples 7a and 8a show calculation of the second order length for which the minimum second order length is not unique.

Example 7a:



$L_2 = 13$

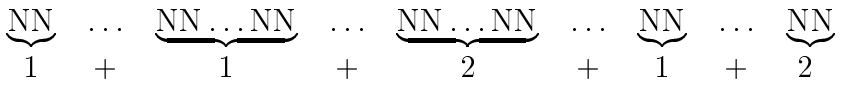
Example 8a:



$L_2 = 13$

As mentioned above, the third order length can be calculated for Examples 7a and 8a by considering the distinct groups of substrings. In this case we consider the groupings of double Ns taken from the second row of Examples 7a and 8a. The groups of substrings as well as the third order length are displayed in Examples 7b and 8b.

Example 7b:



$L_3 = 7$

Example 8b:

$L_3 = 5$

§3.5 Definition of a Discrete Line

A **discrete line** is the connected, minimum size, collection of cells on a square grid represented by a string of Ds and Ns, such that the k^{th}-order length of the string is minimized, where k is the smallest order that yields a unique string.

Part 4: The Algorithm

We refer to the algorithm we use to find the discrete line from $A(0,0)$ to $B(e,n)$ as an ant algorithm. In Part 2 we described ant algorithms, both in general and in the context of the TSP. In Part 3 we defined precisely what we mean by a discrete line. Here we proceed by connecting these ideas, using an ant algorithm to find discrete lines. First, the details of how we applied the ideas of ant algorithms to the discrete line problem are explained, §4.1. This includes a definition of the parameters used and a brief outline of the algorithm itself. Then we report on some of our observations, §4.2.

§4.1 Using Ants to Draw Lines

The ant algorithm we use to generate discrete lines is called *antline*. The environment for *antline* is the same as the environment described in §3.1 for a discrete line. As with our definition of discrete lines, we limit the discussion to collections of cells with discrete slope, $m \geq 1$, between the initial cell, $A(0,0)$, and terminal cell, $B(e,n)$. This has the advantage of allowing us to think of the line as existing in an upper triangular region of the grid, and thus the only moves we consider are diagonal moves (D) and north moves (N). It was this concept of movement that led us to ant algorithms. If we think of dynamically generating the discrete line through a series of moves (Ns or Ds) from the initial cell, $A(0,0)$, one can imagine that there are ants on the square grid making each of the necessary moves.

As stated earlier, we refer to a discrete line by its terminal cell $B(e,n)$. In designing this algorithm, our goal is to send out a group of a ants from the initial cell, $A(0,0)$. These ants then move along the lattice, in this case n moves, and deposit pheromone on the cells that they encounter. Following this, a percentage of the pheromone evaporates and the procedure repeats with the same number of ants, a, leaving the initial cell and each making the same number of moves, n. However, this time, an ant is more likely to follow a path (sequence of moves of Ds and Ns) that has a high level of pheromone. The idea is that through repetition of this process (each repetition is called a cycle), the ants eventually find the ideal path. The ideal path is the collection of cells that satisfies our definition of a discrete line, §3.5.

This is similar to the work by Dorigo and his collaborators on the TSP. In the TSP, the ants move between cities rather than between cells. The length measure in the TSP is Euclidean distance, while in our case the length

measure is the k^{th}-order length described in the definition of a discrete line. In both cases the goal is to minimize the length measures used.

There are seven parameters in the algorithm that are left for the user to choose. These adjust various aspects of the algorithm. The parameters for the algorithm, $antline(e, n, a, NC, \alpha, \gamma, \rho)$, are:

$$\begin{aligned} B(e,n) &= \text{the location of the terminal cell,} \\ a &= \text{the number of ants,} \\ NC &= \text{the number of cycles,} \\ \rho &= \text{persistence of pheromone,} \\ \alpha,\ \gamma &= \text{length parameters.} \end{aligned}$$

Briefly stated, the algorithm moves a ants, from $A(0,0)$, n moves, NC times. The intricacies of the algorithm occur within each cycle. Now that we have roughly established the parameters we can describe the algorithm in more detail.

The Algorithm

$antline(e, n, a, NC, \alpha, \gamma, \rho)$

For each cycle, $c = 1, 2, ...NC$, repeat the following.

- Each ant makes n moves according to the probability functions $prob_N(i,j)$ and $prob_D(i,j)$. The probability function, $prob_N(i,j)$, is the probability of making a north move at location (i,j). Likewise, the $prob_D(i,j)$ denotes the probability of making a diagonal move at location (i,j).

$$\begin{aligned} prob_N(i,j) &= \frac{\eta_{ij}}{\eta_{ij}+\delta_{ij}} \\ prob_D(i,j) &= \frac{\delta_{ij}}{\eta_{ij}+\delta_{ij}} \end{aligned}$$

  η_{ij} = the pheromone associated with a north move at location(i,j).
  δ_{ij} = the pheromone associated with a diagonal move at location (i,j).[24]

- A quantity of the pheromone evaporates. The evaporation is denoted by $1-\rho$. Thus, the parameter $0 \leq \rho \leq 1$ controls the persistence

[24] In the first cycle, $c = 1$, $\eta_{ij} = \delta_{ij} = 0.5$.

of pheromone.[25] For each cycle, c, where c is an integer such that $1 \leq c \leq NC$, the matrices,

$$\eta_c = \rho\eta_{(c-1)} \qquad \text{and} \qquad \delta_c = \rho\delta_{(c-1)}$$

have entries, η_{ij} and δ_{ij} which represent the pheromone associated with a north and a diagonal move respectively at location (i, j) after a percentage of that pheromone has evaporated.

- For all ants that reach the terminal cell, $B(e, n)$, add pheromone to the cells in the collections traversed by each ant (the ants' paths). The quantity of pheromone added is

$$addpher = \left(\frac{1}{L_1}\right)^{\alpha} + \left(\frac{1}{L_2}\right)^{\beta}, \quad \text{where} \quad \beta = \alpha\gamma\frac{\ln L_1}{\ln L_2}$$

and L_1 and L_2 are the first and second order lengths respectively, as defined in Part 3. The parameter β is designed to insure that the second order length is minimized only after the first order length has been minimized. The details of the choice of β will be discussed more later.

It may be helpful, in reading the description of the *antline* above, to discuss the expressions listed in the algorithm. We begin with the probability function. The probability functions determine the movement of the ants in *antline*. There are two matrices that represent amounts of pheromone, η and δ, at a given location on the grid. The total amount of pheromone at a given location (i, j) is the sum of the ij-entries from these matrices, $\eta_{ij} + \delta_{ij}$. Initially, $\delta_{ij} = \eta_{ij} = 0.5$, $\forall\, i, j$. If the terminal cell of the line that we are looking for is $B(e, n)$, then both matrices have dimensions $n \times n$.[26] In the first cycle of the algorithm,

$$prob_N = prob_D = \frac{0.5}{0.5 + 0.5} = 0.5,$$

so the probability of making a north move at location (i, j) is equal to the probability of making a diagonal move at location (i, j). Assume for simplicity that there is only one ant (thus there will be only one sequence of Ds

[25] In other words if the $\rho = .35$, then 35% of the pheromone present remains between cycles and 65% evaporates.

[26] Note that for indexing purposes and because the location of A is (0,0), we assume that the first entries in each of the pheromone matrices are η_{00} and δ_{00}.

and Ns formed for each cycle). Because $prob_N = prob_D$ the north moves and diagonal moves are equally likely. Thus, in the first cycle one can think of the initial sequence of n letters, Ds and Ns, as being formed by a flip of a coin. There is no guarantee that the sequence of Ds and Ns will even reach the terminal cell B.

Regardless of whether or not the sequence of Ds and Ns represents a path that reaches B or not, a quantity of the pheromone evaporates. This is taken into account by multiplying each of the matrices, η and δ, by ρ. The same percentage of pheromone evaporates on every cell in the grid. However, since this is a percentage the quantity of pheromone that evaporates is relative to the amount present at a given cell. The effect of evaporation varies. One could say that it serves to emphasize or de-emphasize the effect of *addpher* on the probability functions. This will be discussed further in §4.2.

If the path formed by the sequence of Ns and Ds does reach B, then pheromone is added to the cells on the path. The quantity added is given by *addpher*. The parameter $\beta = \alpha\gamma\frac{\ln L_1}{\ln L_2}$ is defined in terms of γ, so as to make sure that the first order length dominates in the expression for *addpher*.

$$\begin{aligned}
\left(\frac{1}{L_1}\right)^{\alpha} &> \left(\frac{1}{L_2}\right)^{\beta} \\
\alpha \ln L_1 &< \beta \ln L_2 \\
\beta &> \alpha\frac{\ln L_1}{\ln L_2} \\
\beta &= \alpha\gamma\frac{\ln L_1}{\ln L_2}, \quad \gamma > 1
\end{aligned}$$

In other words, only in the event that after we minimize the first order length, and we find that this does not produce a unique string, do we take into account the second order length. Likewise, if our algorithm took into account the third order length, then *addpher* (and thus *antline*) would have to have a parameter that forced the third order length to only be considered after it was found that there were multiple strings with minimum second order length. Our algorithm accounts for only the first and second order lengths, L_1 and L_2. To generalize the algorithm one would have to add

pheromone according to the following quantity:

$$addpher = \sum_{i=1}^{k} \left(\frac{1}{L_i}\right)^{\alpha_i},$$

where each successive α_i accounts for the decreasing order of importance of the subsequent lengths. In our algorithm α and γ (and hence β) are user defined parameters, thus to follow that example each α_i would depend on user defined parameters as well. However, if this amount of flexibility is not required, it is possible to simplify the length parameters to just one, on which each α_i is dependent. The strings tested in our algorithm were sufficiently short, so that limiting the lengths to just the first and second order was reasonable.

Considering the single ant example, if during the first cycle a single ant creates a path that does not reach B, then pheromone will evaporate uniformly on all cells in the lattice. However, no pheromone will be added, thus keeping an equal amount of pheromone on every cell. It follows that in the second cycle, the probability of all moves remain equal. Now instead assume that in the first cycle the single ant creates a path that does reach B. The pheromone evaporates uniformly as before. Next, for every cell in the path that reached B, *addpher* units of pheromone are added to the existing pheromone at the cells' locations. This means that in the second cycle, the probabilities are not all equal. The amount by which the probabilities differ will vary according to the user defined parameters and the length measures for the collection of cells. To be more specific, assume that the first move of the path that reaches B, found in the first cycle, was a north move. It follows that in the second cycle the probability of making a north move at location (0,0) is larger than the probability of making a diagonal move at (0,0). One can see that if there are multiple ants, and thus multiple paths found in the algorithm, there are more adjustments to the probability functions.

If there are two ants that reach B in the first cycle and their first moves are different (one N and one D), this may or may not change the probabilities associated with moves at $(0,0)$. The probabilities will not change if the first order lengths are the same for these two paths, then $prob_N(0,0) = prob_D(0,0)$. However, the probabilities will change if the first order lengths differ. Consider that the path with a north as a first move has a shorter first order length, then it follows that $prob_N(0,0) > prob_D(0,0)$.

§4.2 Observations of Antline

In experimenting with the algorithm we observed several phenomena. The first question that we asked was how many ants, if any, would reach the terminal cell $B(e, n)$ throughout the runs of the algorithm. For a particular $B(e, n)$, we compare the number of ants that reach B throughout the algorithm to the number of ants that reach B when the paths are generated randomly. Not surprisingly, we find that the number of ants that reach B in each cycle depends on the user defined parameters. In investigating the ability of the algorithm to generate the unique discrete line, we find that there is significant dependence on the distribution of lengths (in the example provided, simply first order lengths). The parameter selections that we experimented with are set up so as to make the effect of the second order length negligible. Thus one can assume that the output in the figures presented is solely based on the first order length.

We begin with the terminal cell $B(5, 11)$ because there is only one collection of cells with the minimum first order length, $L_1 = 11$. Thus, first order length determines the unique discrete line. This is the collection of cells that can be represented by the string NDNDNDNDNDN. In Figure 26 we show what happens when paths are randomly generated. If 100 ants make 11 moves, randomly choosing Ns and Ds, Figure 26 shows the quantity of paths that would reach $B(5, 11)$ over the course of 100 cycles.

Figures 27–34 show experiments in which parameters ρ and α are varied and the number of cycles versus the quantity of cells that reached the terminal cell $B(5, 11)$ are plotted. In some of the figures, the parameters a and NC also differ. This was done to provide a better viewing window of phenomena observed.

Notice that in Figures 27-30, as the parameter ρ increases, the algorithm tends to generate first order lengths for $B(5, 11)$ that vary more and more. In other words, as the persistence of pheromone increases, or the evaporation decreases, there appears to be more scouting.[27] This scouting, or the tendency of the algorithm to continuously explore for new, potentially better, solutions, was observed in the work by Dorigo and his collaborators on the TSP, §2.2. An extreme example of scouting behavior would be to randomly generate collections of cells. [Figure 26] Thus, there is potential for too much

[27] One could argue that instead of "more scouting," there is a larger variety in the scouting. At any rate, there is a clear change in the algorithms performance as ρ changes.

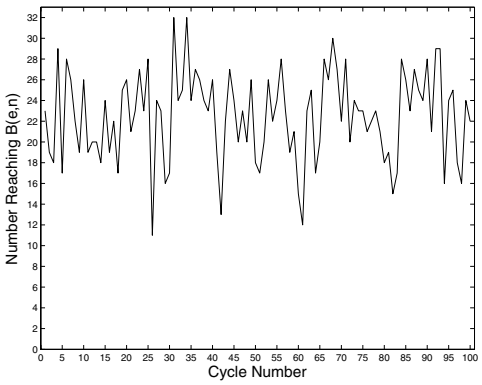


Figure 26: The number of randomly generated collections of cells, out of 100 per cycle, that reach the terminal cell $B(5, 11)$.

scouting. The danger of there being too little scouting is that the algorithm could generate a collection of cells that, although the collection has a low first order length, it does not have the minimum first order length.[28] This suggests that there are some values of the parameter ρ that are more desirable than others.

The same scouting phenomenon can be observed in Figures 31–34. However, in addition to changing the values of the parameter ρ we have increased the value of the parameter α by one. This appears to have the effect of shifting the general shape of the graph horizontally to the right. Increasing the value of the parameter α amounts to lessening the relative importance of the first order length. Said another way, as α increases, $addpher = (1/L_1)^\alpha + \ldots$, decreases. Thus, it takes the pheromone more cycles to accumulate, which results in the ants needing more cycles to reach $B(5, 11)$.

[28]It is important to note that in Figure 27, although all of the 25 ants consistently reach $B(5, 11)$ after about 5 cycles, this does not necessarily mean that the algorithm is displaying stagnation behavior. It is certainly possible that there is significant variation among the paths reaching $B(5, 11)$. That information is simply not available from this graph. Instead, graphs like those shown in Figures 38 and 39 display the absence of stagnation behavior.

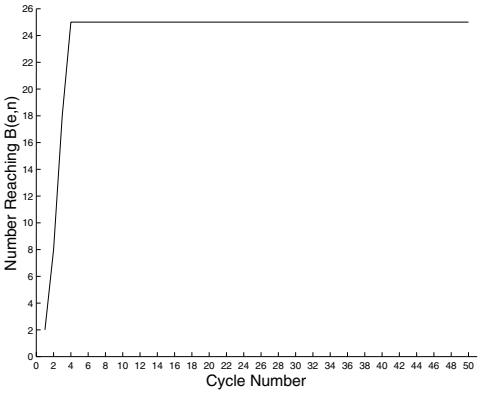


Figure 27: All of the ants reach the terminal cell, $B(5,11)$, after 5 cycles with $\alpha = 1$ and $\rho = 0.25$, $antline(5,11,25,50,1,100,0.25)$.

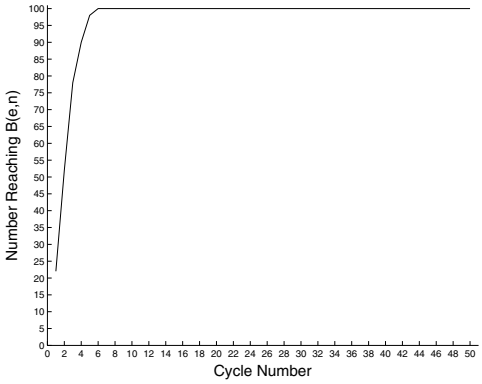


Figure 28: All of the ants reach the terminal cell, $B(5,11)$, after 7 cycles with $\alpha = 1$ and $\rho = 0.5$, $antline(5,11,100,50,1,100,0.5)$.

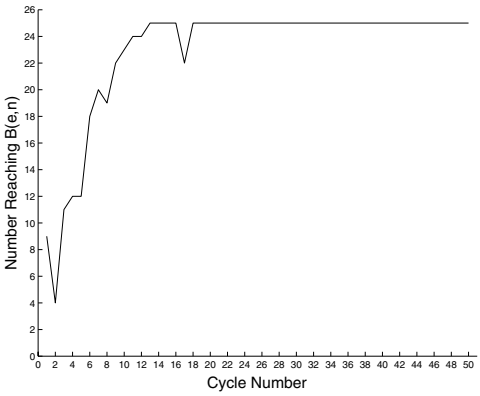


Figure 29: All of the ants reach the terminal cell, $B(5,11)$, after 20 cycles with $\alpha = 1$ and $\rho = 0.75$, $antline(5,11,25,50,1,100,0.75)$.

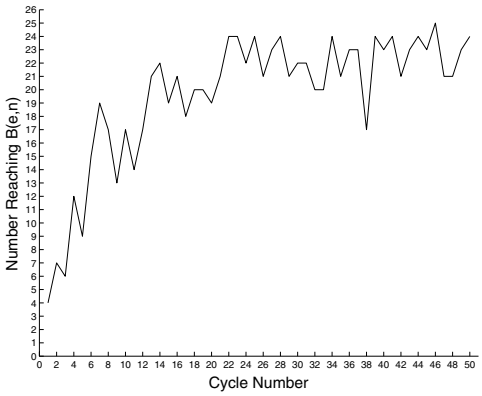


Figure 30: After 50 cycles the number of ants that reach the terminal cell, $B(5,11)$, continues to vary, for $\alpha = 1$ and $\rho = 1$, $antline(5,11,25,50,1,100,1)$.

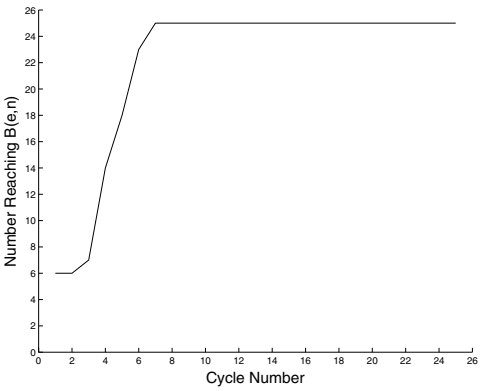


Figure 31: All of the ants reach the terminal cell, $B(5,11)$, after 7 cycles with $\alpha = 2$ and $\rho = 0.25$, $antline(5,11,25,25,2,100,0.25)$.

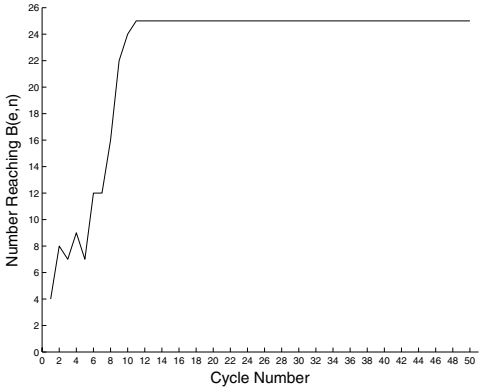


Figure 32: All of the ants reach the terminal cell, $B(5,11)$, after 11 cycles with $\alpha = 2$ and $\rho = 0.5$, $antline(5,11,25,50,2,100,0.5)$.

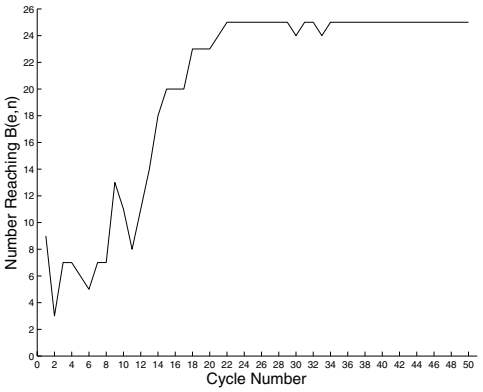


Figure 33: All of the ants reach the terminal cell, $B(5,11)$, after 24 cycles with $\alpha = 2$ and $\rho = 0.75$, $antline(5,11,25,50,2,100,0.75)$.

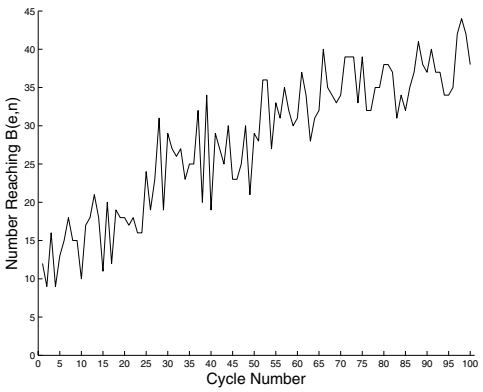


Figure 34: After 100 cycles the number of ants that reach the terminal cell, $B(5,11)$, continues to vary, for $\alpha = 2$ and $\rho = 1$, $antline(5,11,50,100,2,100,1)$.

The overall change within both sets of figures, 27–30 and 31–34, is worthy of discussion. One might expect that because ants are attracted to cells that have higher levels of pheromone, as the persistence of pheromone increases (as it does in each set of figures) so would the number of ants reaching B. In other words, there would not be the variation in the number of ants reaching the terminal cell, B, that there is when $\rho = 1$. [Figures 30 and 34] In fact, both sets of figures display just the opposite behavior. Why is it that when there is no evaporation there is so much variation in the number of ants that reach B? Likewise, why is that when there is 75% evaporation, there is such a sudden and dramatic increase in the number of ants that reach B? The answer to these questions lies in the fact that the important measure for the ants behavior is not simply in the amount of pheromone added or in the percentage of evaporation of the pheromone. Instead, the key is in the relative size of these two quantities.

To explain the subtle balance between ρ and *addpher* we begin with a simplified example. We run the algorithm with the following set of parameters:

$$antline(e, n, a, NC, \alpha, \gamma, \rho) = antline\Big(3, 5, 2, 25, 1, 100, \frac{1}{3}\Big).$$

For this particular choice of $B(e, n) = B(3, 5)$, the minimum first order length yields the unique discrete line we are searching for. Thus, in this case, we can reduce the expression for *addpher* to

$$addpher = \frac{1}{L_1}.$$

We then make the following observations.

- The minimum first order length for $B(3, 5)$ is given by the string, D N D N D, with $L_1 = 5$.
- The maximum first order length for $B(3, 5)$ is given by the string, D D D N N, or equivalently N N D D D, with $L_1 = 9$.
- The range of *addpher* is therefore given by

$$\frac{1}{9} \leq \frac{1}{L_1} \leq \frac{1}{5}.$$

- For each cycle, c, where c is an integer such that $1 \leq c \leq NC$,

$$\delta_c = \rho\delta_{c-1} = \frac{\delta_{c-1}}{3} \quad \text{and} \quad \eta_c = \rho\eta_{c-1} = \frac{\eta_{c-1}}{3},$$

  and initially all entries of δ and η are $\frac{1}{2}$.

Clearly, in order to generate the unique discrete line, the first move should be a diagonal move, (D). Suppose that for the first four cycles neither of the two ants reach $B(3,5)$. The result is

$$\delta_4 = \eta_4 = \left(\frac{1}{2}\right)\left(\frac{1}{3^4}\right) = \frac{1}{162}.$$

This has not changed the probabilities of any moves, $prob_N(i,j) = prob_D(i,j) = 0.5$. Now suppose that in the fifth cycle, one of the ants reaches $B(3,5)$ with the string N D D N D. The first order length for that string is given by $L_1 = 6$. This changes the entries in the matrices associated with the first move to

$$\delta_{00} = \left(\frac{1}{2}\right)\left(\frac{1}{3^5}\right) = \frac{1}{486} \quad \text{and} \quad \eta_{00} = \left(\frac{1}{2}\right)\left(\frac{1}{3^5}\right) + \frac{1}{6} = \frac{41}{243}.$$

This changes the probability that the first move is a north (N) to

$$prob_N(0,0) = \frac{\eta_{00}}{\eta_{00} + \delta_{00}} = \frac{\left(\frac{1}{2}\right)\left(\frac{1}{3^5}\right) + \frac{1}{6}}{\left(\frac{1}{2}\right)\left(\frac{1}{3^5}\right) + \frac{1}{6} + \left(\frac{1}{2}\right)\left(\frac{1}{3^5}\right)} = \frac{82}{83} \approx 0.988.$$

This example demonstrates that for sufficiently long strings, although initially $addpher << \eta_{ij}, \delta_{ij}$, after a certain number of cycles, it is possible (in fact likely, depending on the value of ρ) that $addpher >> \eta_{ij}, \delta_{ij}$. Of course this example depends on the fact that neither of the ants reached $B(3,5)$ in the first few cycles. In this case over 30% of possible combinations of n moves (Ns and Ds) reach $B(3,5)$. However, this percentage is much less for different values of e and n. For a given $B(e,n)$, the number of strings that reach $B(e,n)$ is given by, $\binom{n}{e}$ and the total number of strings possible is given by 2^n. Thus, the probability that a string will reach $B(e,n)$ is

$$prob_B(e,n) = \frac{n!}{e!\,(n-e)!\,2^n}.$$

Now we look at the effect of changing ρ. Since $0 \leq \rho \leq 1$, for values of ρ near zero, the distinction in the size of *addpher* and the entries of η and δ will become more and pronounced. This in turn creates a dramatic difference in $prob_N$ and $prob_D$. In fact, as $\rho \to 0$, it takes fewer cycles for *addpher* to become much larger than the entries of η and δ. This describes the sudden and dramatic increase in the number of ants that reach B in Figure 27. This also explains the phenomena we observe in Figures 30 and 34. As $\rho \to 1$, the distinction between the entries of η and δ is not as pronounced, thus the probabilities are closer to 0.5.

To generalize this idea we can look at the probability of a north move at location (i, j) in the following way,

$$prob_N = \frac{\rho\eta + addpher}{\rho(\eta + \delta) + addpher}.$$

Let $\eta \approx \delta$, then [29]

$$\begin{aligned} prob_N &\approx \frac{\rho\eta + addpher}{2\rho\eta + addpher} \\ &= \frac{1 + addpher/(\rho\eta)}{2 + addpher/(\rho\eta)}, \end{aligned}$$

and

$$prob_D = 1 - prob_N.$$

Figure 35 shows how the relationship between *addpher* and ρ effects the probability functions. Notice that for $\rho\eta >> addpher$, or as $\rho \to 1$, $prob_N \approx prob_D \to 0.5$. Likewise, notice that for $\rho\eta << addpher$, or $\rho \to 0$, $prob_N >> prob_D$.

In both of our examples, we chose the terminal cells, $B(5, 11)$ and $B(3, 5)$, partly because we found that there was only one collection of cells that had the minimum first order length. This is certainly not the case for all $B(e, n)$. However, the benefit (for explanation purposes) is that in finding the discrete lines $B(5, 11)$ or $B(3, 5)$, we can set the parameter γ to be large, making the second order length negligible. In addition, if one could predict the order of the length necessary to provide the unique discrete line by our definition, then the algorithm could be simplified. This raises the question of what the distribution of the first order lengths looks like for a particular collection of cells terminating at $B(e, n)$.

[29] Remember that initially $\eta = \delta$, so this approximate equality is reasonable.

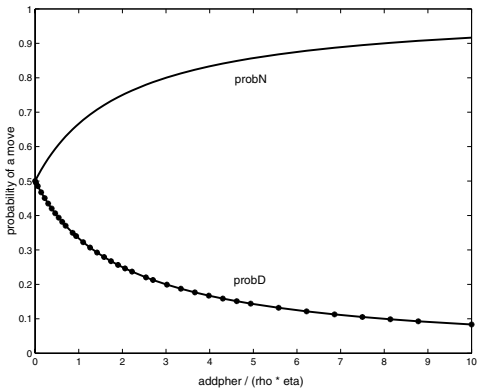


Figure 35: The probabilities versus the relative size of *addpher* and $\rho\eta$.

We found that the shapes of the distributions of first order length for the values e and n were all quite similar. Figures 36 and 37 provide examples of the distributions for two choices of $B(e, n)$. The distribution tends to be skewed so that the median of the distribution is less than the mean. It is also typical that there are two peaks, the larger having a smaller first order length and occurring closer to the mean of the distribution. This can be seen in Figure 37 for first order lengths 15 and 17. In this sense the distribution shown in Figure 36 is slightly atypical because it is relatively smooth and does not display two peaks.

While investigating this we notice the following properties of the string quantities, some of which were mentioned earlier.

- The number of possible strings is given by 2^n.
- The number of strings that reach $B(e, n)$ is given by $\binom{n}{e}$.
- Let the number of strings with the minimum first order length be denoted $num(minL_1)$.
  - If $n = 2e$, then $num(minL_1) = 2$, where the strings are simply the reverse of one another.[30]

[30]In this case, as mentioned earlier, the strings represent the same discrete line. For

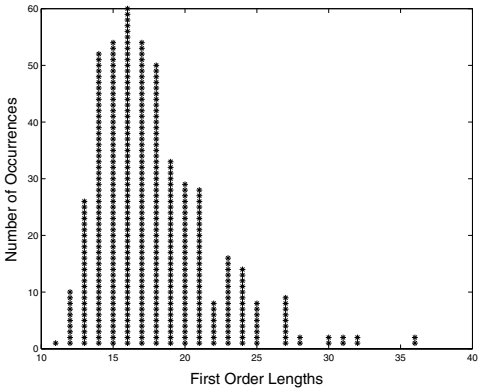


Figure 36: The distribution of first order lengths for collections of cells terminating at $B(5, 11)$.

Figure 37: The distribution of first order lengths for collections of cells terminating at $B(4, 10)$.

- If $n \neq 2e$, then

$$num(minL_1) = \binom{m+1}{R},$$

where $M = max(e, n-e)$, $m = min(e, n-e)$ and $R = M \bmod (m+1)$.

- Let the number of strings with the minimum second order length be denoted $num(minL_2)$, where $M = max(e, n-e)$, $m = min(e, n-e)$ and $R = M \bmod (m+1)$.[31]

  - If $m + 1 - R = R$, or $m = 2R - 1$, then $num(minL_2) = 2$, where the strings are again the reverse of one another.
  - If $m \neq 2R - 1$, then

$$num(minL_2) = \binom{\ell+1}{r},$$

example, given $B(3, 6)$, N D N D N D represents the same discrete line as D N D N D N.

[31] The similarity in the expression of $num(minL_1)$ and $num(minL_2)$ is a direct result of the similarity in the definitions of L_1 and L_2. Just as one can generalize the notion of higher order lengths, one can extend these ideas to define $num(minL_k)$, for integers $k > 2$.

where $u = max(R, m + 1 - R)$, $\ell = min(R, m + 1 - R)$, and $r = u \bmod (\ell + 1)$. [32]

One can also view the scouting phenomenon as well as other characteristics of the distributions of first order lengths by investigating the mean, median, standard deviation and interquartile range over the cycles of the algorithms for varying sets of parameters. Notice the difference in the behavior of the algorithm in Figures 38 and 39. In Figure 39 the scouting phenomenon is more evident. You can observe that in each case there is no stagnation behavior since the standard deviation and interquartile range are never zero. In addition, the algorithm is displaying the shape of the distribution of first order lengths in that the median tends to be smaller than mean.

[32] We have given expressions for calculating the number of strings that have the minimum first and second order lengths. It is also possible to write expressions for the number of strings that have the minimum k^{th}-order length, for $k > 2$. Thus, using this information we can determine the value k, the smallest order that yields a unique string, in our definition of a discrete line.

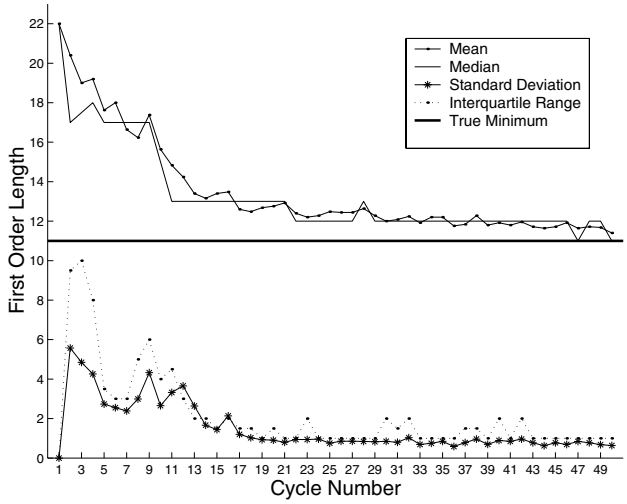


Figure 38: $antline(5, 11, 25, 50, 2, 100, 0.5)$

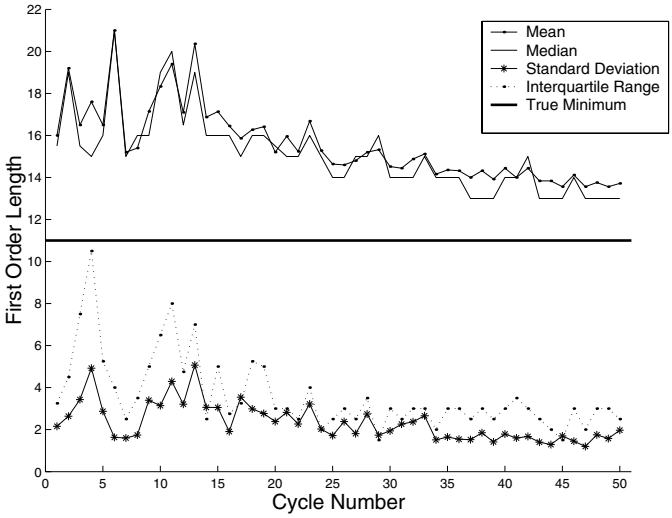


Figure 39: $antline(5, 11, 25, 50, 2, 100, 0.75)$

Part 5: Conclusions and Further Study

In order to present some of the questions and conclusions that have arisen from this work, we find it helpful to summarize our goals and what has been accomplished towards achieving them. The primary goal of this study is to take the first steps in applying models of complex systems to defining and generating geometric constructs. In this case, the geometric construct that we have defined and generated is the discrete line. Our work falls into two categories, the definition of a discrete line and the algorithm used to generate a discrete line.

Our definition of a discrete line produces a connected, minimum size, collections of cells on a grid. By minimizing specified length measures of the string representations of these collections of cells, our definition yields a unique discrete line. This unique discrete line satisfies the three criteria of our definition. First, the collection of cells satisfies the column property. Second, the collection is as evenly distributed as possible. And third, it contains the core cells in all cases except when the core is overly connected.

After defining a discrete line we designed an algorithm to generate such a line. The algorithm is based on an interpretation of the process of ants foraging for food. A fixed number of ants, a, make n moves, for each cycle (up to NC cycles). During a cycle, some ants deposit pheromones along their paths. A quantity of the pheromone evaporates, and then the process is repeated in the next cycle with the ants tending to follow paths with more pheromone. The algorithm is non-deterministic because the ants follow paths according to a probability function that incorporates the length measures in our definition of a discrete line. By choosing the shortest length path after all of the cycles, we generate a discrete line.[33]

What is it that makes our approach different from others, e.g. Bresenham's algorithm [5]? There are several characteristics that stand out.

- Our definition of a discrete line is not based on a continuous line. We do not generate a continuous line and then try to approximate it in a discrete environment.

- The discrete line is generated through an iterative process. For each cycle in the algorithm, the lengths are minimized. Then, through the

[33] As discussed in Part 4, the outcome of the algorithm is dependent on the user defined parameters.

probability functions, the ants use the information from the previous cycle to generate paths in the next cycle.

- We use independent agents, the ants, acting simultaneously to generate the discrete line. In this way, the agents create a parallel computation process.

- The rules governing the agent behavior (in this case the ants' north or diagonal moves) are simple and local. There is no global control over the output of the algorithm, for example, supervision that "corrects" collections of cells that do not appear straight.

- The algorithm is non-deterministic.

As with many initial studies, the results often produce more questions than definitive conclusions. Many of these open questions fall under the category of avenues for further study. With this in mind, we can divide some of the questions and conclusions that arise from this work into those associated with our definition of a discrete line, §5.1, and those associated with the implementation of our algorithm, §5.2. Our final conclusions are left to §5.3.

§5.1 The Definition: Questions and Conclusions

In defining a discrete line we characterized what it is that makes a collection of cells appear straight to the human eye. We did this by focusing on those collections of cells that appeared connected, had an evenness in the distribution of cells, and included some essential cells. There are other intuitively appealing characteristics that we would like to include in our definition of a discrete line. For example, we found that the string representations of the discrete line (composed of Ds and Ns) often form a palindrome. Likewise, the pattern of columns of cells in the discrete line (represented by a string of 0s and 1s) often form a palindrome. On the other hand, not all palindromes fit our criteria for a discrete line. In fact, there are certain discrete lines for which the string representations can never form palindromes. For example, if the number of Ns (given by $n-e$) and Ds (given by e) in a string representation of a line are both odd, then the string cannot be written as a palindrome.[34] This brings up the question of how to characterize those

[34]The simplest examples of strings that cannot be written as palindromes are N D, and N D N D N D.

discrete lines, given by our definition, which have string representations that are palindromes.

We have also considered variations in the level of connectivity of the cells of our discrete line. Our definition requires that the discrete line be a minimum size connected collection of cells. This is done to avoid what we refer to as overly connected collections of cells. If we allow overly connected collections of cells, we could instead define the discrete line to be the hull of the discrete line. Likewise, if we allow disconnected collections of cells we could define the discrete line to be the core of the discrete line. If one considers these variations in the level of connectivity of the cells of our discrete line, the criteria for collections of cells that appear to the eye to be a discrete line are closely dependent on the resolution of the lattice on which the collections are displayed. For example, when one observes a dotted line from a close distance it is clearly dotted, but as that distance increases, the dotted line appears more and more continuous. Similarly, a thick line appears thin from a distance. Future study may include experimentation with other notions of connectivity and different resolutions.

The choice of a square lattice seemed natural at the onset of this study. However, knowing that the only regular polygons that tile the plane are squares, triangles and hexagons, this choice comes into question. We have begun investigations into whether our definition of a discrete line on a square lattice translates to a notion of a discrete line on a hexagonal lattice. This is particularly appealing because of the difference in connected collections of cells on a hexagonal lattice versus those on a square lattice. Two cells on a square lattice can be connected if they share only a vertex and no edges, however, every connected pair of cells on a hexagonal lattice must share an edge.

§5.2 The Algorithm: Questions and Conclusions

Within the design of the algorithm we made choices which are worthy of more investigation. Many choices in the design of our algorithm were guided by ant foraging behavior. We have attempted to follow this guide in our design to the extent that it benefits the output of our algorithm. We could instead concentrate on improving the performance of our algorithm. We suspect that some of the comments that follow could lead to ways to improve the performance of the algorithm at the expense of lessening the algorithms similarity to ant foraging behavior.

Observations

- There is a subtle interplay between the pheromone added, the evaporation, and the probability functions. The amount of pheromone added is dependent on the lengths measured (first order, second order, etc.). The quantity of pheromone that evaporates at a cell is dependent on the amount present at that cell. The probability functions are controlled by the quantity of pheromone present at a cell, that is, the sizes of the entries η_{ij} and δ_{ij} relative to one another.[35] These sizes are in turn dependent on the balance between the amount of pheromone added and the quantity of evaporation.

- The number of ants that cross a particular cell is implicitly accounted for in the algorithm. As many ants make a particular move at cell (i, j), the pheromone at that cell accumulates. As mentioned earlier, this may or may not have a direct effect on the probability functions. However, the larger this accumulation becomes, the less influence *addpher* will have.[36]

- By varying the user controlled parameters, which in turn control the balance between *addpher* and the entries η_{ij} and δ_{ij}, there is some control over the amount of scouting versus stagnation behavior displayed by the algorithm.

- One can interpret the role of the number of ants, a, in the algorithm as analogous to the number of students in a classroom working on a problem. Following that analogy, the number of cycles, NC, can be interpreted as the number of opportunities that the students have to solve the problem. If one substitutes agents for students, increasing a amounts to increasing the number of agents who can learn to find the ideal path, while increasing the number of cycles amounts to increasing the number of opportunities for those agents to learn.

[35] The quantities η_{ij} and δ_{ij} represent the pheromone associated with north and diagonal moves respectively. For a more detailed description see §4.1, The Algorithm.

[36] The quantity *addpher* represents the quantity of pheromone added to cells. For a more detailed description see §4.1, The Algorithm.

Questions and Further Study

- Could the importance of the terms of *addpher* be governed by something other than powers of α and β?
- The persistence of pheromone is represented by multiplying the matrices η and δ by the scalar ρ, $0 \leq \rho \leq 1$. This makes the quantity of pheromone that evaporates dependent on the amount that is present at the current cell. What would be the effect in the algorithm of changing the role of evaporation so that a fixed quantity of pheromone evaporates regardless of the amount present at the current cell?
- If the quantity of pheromone added depended directly on the quantity of pheromone present, how would that effect the behavior of the algorithm?
- How would the algorithm be affected if the evaporation of pheromone at a given cell depended on the lengths (first order, second order, etc.) of the paths containing that cell in previous cycles?
- If we think of the algorithm as a directed random search, can we control how directed the search is? In other words, are there optimum values, or ranges of values, for the parameters a, NC, α, γ and ρ?
- Does increasing the number of ants, a, and cycles, NC, always lead to a higher probability of getting the unique discrete line? Is there a point at which increasing these parameters has no effect on the algorithm's performance?
- The cost of increasing the parameters a and NC can be measured by the time it takes for the algorithm to run. At what point does this time become prohibitive?

§5.3 Where Does That Leave Us?

In our work we have developed a definition of a discrete line. Our definition characterizes mathematically what it means for a collection of cells in a discrete environment to "look straight" without being based on some approximation to a continuous line. The collections of cells on a square lattice are represented by a two letter alphabet. We have defined a series of length

measures that, when applied to these strings and minimized, yield a unique collection of cells that we call the discrete line, $B(e, n)$.

We then developed a non-deterministic algorithm that can be used to generate discrete lines. This algorithm is inspired by the behavior of ants foraging for food. The algorithm sends a agents (ants) out from the initial cell, to make n moves, NC times. As the agents move across the landscape, pheromone is deposited on some of the cells they encounter. Throughout the process of depositing pheromone, the evaporation of pheromone, and the probabilistic movement of the agents, a minimum length path is found. When reasonable parameters are chosen by the user, the algorithm is very likely to generate the discrete line as we have defined it.

The process of defining a discrete line and developing an algorithm to generate discrete lines has provided insight into extending these ideas to other geometric constructs. We are looking into using the same process of minimizing successive length measures, applied to larger alphabets, to define a discrete line composed of cubes in three dimensions. In addition, we are considering ways to extend these ideas to curves in two and three dimensions, to a definition of discrete curvature, and to a notion of discrete orthogonality and discrete angle.

Acknowledgments

This study was undertaken while Ms. Geer was visiting the Applied Management and Computing Division, Lincoln University. The authors wish to acknowledge the financial support that was provided by the division in order to make the visit possible. In particular, Ms. Geer would like to thank the members of the staff of the Applied Management and Computing Division at Lincoln University for helping to make the visit both productive and pleasurable.

References

[1] E. Bonabeau, M. Dorigo, G.Théraulaz, *Swarm Intelligence, From Natural to Artificial Systems*, Oxford University Press, 1999.

[2] E. Bonabeau, G. Théraulaz,"Swarm Smarts," *Scientific American*, March, 2000, pp. 54-61.

[3] M. Dorigo, web site, http://iridia.ulb.ac.be/∼mdorigo/ACO/ACO.html.

[4] M. Dorigo, V. Maniezzo, A. Colorni, *The Ant System: Optimization by a colony of cooperating agents*, IEEE Transactions on System, Man and Cybernetics-Part B, Vol. 26, No. 1, 1996, pp. 1-13.

[5] D. Hearn, M. Pauline Barker, *Computer Graphics*, 2nd ed., Prentice Hall, Englewood Cliffs, 1994.

[6] J. Holland, *Emergence, From Chaos to Order*, Perseus Books, Cambridge, Massachusetts, 1998.

[7] J. Holland, *Hidden Order, How Adaptation Builds Complexity*, Perseus Books, Reading, Massachusetts, 1995.

[8] P. Hraber, T. Jones, S. Forrest, "The Ecology of Echo," *Artificial Life 3*, Massachusetts Institute of Technology, 1997, pp. 165-190.

[9] K. Kelley, *Out of Control, The New Biology of Machines, Social Systems and the Economic World*, Perseus Books, Reading, Massachusetts, 1994.

[10] I. Stewart, *Life's Other Secret, The New Mathematics of the Living World*, John Wiley & Sons, Inc., New York, 1998.