GRADUATE THESIS APPROVAL FORM AND SIGNATURE PAGE

Instructions: This form must be completed by all doctoral students with a thesis requirement. This form MUST be included as page 1 of your thesis via electronic submission to ProQuest.

Thesis Title: WoC-Bots: Swarms of Biologically Inspired Prediction Agents

Author's Name: Sean Patrick Grimes

Submission Date: 05/24/2023

The signatures below certify that this thesis is complete and approved by the Examining Committee.

Role: Chair Name: David Breen
Title: Professor
Department: Computing
Approved: Yes Date: 05/24/2023

Role: Member Name: Dario Salvucci
Title: Professor
Department: Computing
Approved: Yes Date: 05/24/2023

Role: Member Name: Vasilis Gkatzelis
Title: Associate Professor
Department: Computing
Approved: Yes Date: 05/24/2023

Role: Member Name: Mark Zarella
Title: Senior Associate Consultant
Institution: Mayo Clinic
Approved: Yes Date: 05/24/2023

Role: Member Name: Ehsan Khosroshahi
Title: Assistant Teaching Professor
Department: Computing
Approved: Yes Date: 05/24/2023

WoC-Bots: Swarms of Biologically Inspired Prediction Agents

A DISSERTATION PRESENTED
BY

SEAN P. GRIMES

AND SUBMITTED
TO THE FACULTY
OF
DREXEL UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

APRIL 2023

Acknowledgments

Many people have helped, motivated, and influenced me to this point, I want to thank everyone who has supported and encouraged me throughout this process. I am grateful for your help. I would like to acknowledge some individually for their help and contributions.

I want to thank my committee members, in particular my advisor, Dr. David Breen, whose knowledge, guidance, and mentorship enabled me to pursue research topics I was truly excited by. I also want to sincerely thank my other committee members; Dr. Mark Zarella for his time and assistance in exploring how this method can be applied in the medical community; Dr. Vasilis Gkatzelis for introducing me to game theory and environmental design; Dr. Ehsan Khosroshahi for enlightening conversations about the world of artificial intelligence; and Dr. Dario Salvucci for his thoughtful and helpful feedback and guidance.

A special thanks to Dr. Andrew W.E. McDonald for inviting me to explore the world of computer science research and for his considerate and (often) helpful feedback and advice.

In addition, I want to thank my in–laws, Cynthia, Michael, and Kristin for their thoughtful encouragement. I am exceptionally thankful for my parents, Karen and Joseph, for their never ending support, without which I would not be where I am today. Finally, I am eternally grateful for my wife, Chelsey, who has provided incredible support, demonstrated considerable patience, and constantly inspires and encourages me.

Table of Contents

List of Tables

List of Figures

Abstract

WoC-Bots: Swarms of Biologically Inspired Prediction Agents

Sean P. Grimes
Advisor: David E. Breen

This dissertation presents Wisdom–of–Crowds–Bots (WoC-Bots), biologically–inspired, simple, and modular agents which work together in a multi–agent environment to collectively make binary predictions. Building on the theoretical underpinnings of Wisdom of Crowds, WoC-Bots represent a knowledge–diverse crowd where each agent is trained on a subset of available information. A honeybee–derived swarm aggregation mechanism was developed to elicit a collective prediction with an associated confidence score. Due to the multi–agent architecture, WoC-Bots can be distributed across multiple compute nodes, reducing training and inference time. Importantly, this architecture demonstrates significant key advantages over traditional classification methods while maintaining comparable predictive performance. Specifically, new and previously unknown input features can be included in an existing classification problem without retraining existing agents. New input features, combined with existing features, are encapsulated into newly generated agents before agents are injected into an existing classification task. Further development led to a "meta–swarm", where an external prediction is used as the core belief of an agent, replacing a simple multi–layer perceptron network. The external prediction requires zero knowledge of source data, maintaining the localization and privacy of the data used to generate the prediction, enabling collaboration between institutions unable to share their data externally.

Chapter 1. Introduction

Predicting the likely outcome of future events is an important classification task for many different fields. Disease outbreak and risk factors, business success, economics, and many other applications can benefit from better, more flexible, forecasting. Healthcare professionals are aided by these algorithms, improving patient outcomes [2, 67], while Hollywood production studios direct advertising and marketing budgets based on the outputs of these complex algorithms [27]. Significant work has been done in this area, including artificial neural networks (ANNs) [66, 60, 90], support vector machines (SVMs) [78, 22], logistic regression [127, 30] and decision tree–based methods Random forest [19], XGBoost [23], and AdaBoost [100], among other algorithms. While all of these algorithms can perform well on their own [62, 106, 47], as with all algorithms, they do have limitations.

Logistic regression requires all observations to be independent and can be extremely sensitive to the input data, easily over–fitting on training data with few samples and a large number of features [89]. SVMs perform poorly in noisy environments, requiring significant data cleaning and preparation, and struggle with large datasets as training time grows very quickly [5]. Ensemble methods, including AdaBoost, XGBoost, and Random forest are computationally complex and cannot expand the number of input feature without retraining weak learners [39]. Similarly, deep neural networks (DNNs) cannot expand the number of inputs once trained and typically have long training times, which reduce their ability to adapt to changes in input features or availability of new input features [136]. Additionally, DNNs require vast amounts of input data to effectively determine relationships between inputs and outputs [108]. An alternative method in development at HP-Enterprise[1] uses swarm intelligence, decentralized and independent learners, before aggregating anonymous "learnings" [3, 75]. While this is similar to federated learning, where data privacy is protected by sending anonymized "learnings", i.e. not the original data, to a centralized model and averaging the inputs [65, 92], it fundamentally differs by replacing a central server, distributing learnings to all edge devices using blockchain technology to track updates to the known model, allowing each peer in the network to act as an equal [46].

The swarm intelligence–based method developed by HP-Enterprise has many attractive qualities, similar to federated learning; edge nodes, which could be individual hospitals, doctor's offices, or other research

[1] https://www.hpe.com/us/en/hpe-swarm-learning.html

institutions, can work with and train learning models without sharing private patient information with other institutions, while still updating a known model between all participating institutions. Additionally, in the HP-Enterprise system, as each edge node is working with a subset of overall data, there is no need for large, centralized, and expensive compute hardware [83]. However, this method requires consistency for all models at the edge nodes, with the same input features used to train each of the models. The participating institutions need to work with, for example, a deep neural network using the same input features; therefore it is not possible for one institution to use Random forest, another to use deep learning, and another to use AdaBoost [97]. The motivation of this dissertation is to develop a classification algorithm that builds on existing methods to offer new and improved functionality that enables distributed learning and classification while allowing for many different types of learners to be included and aggregated in order to drive an overall classification from the complete set of available inputs.

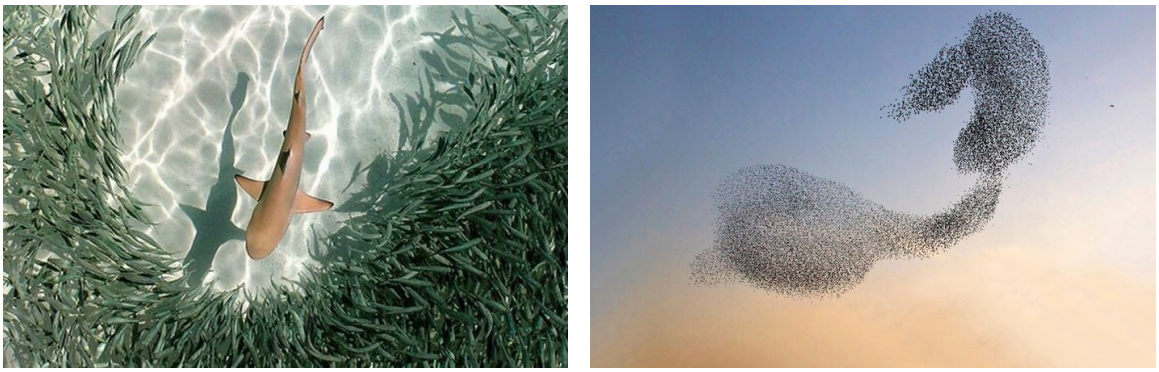

Figure 1.1: Schooling fish avoiding a predator from the International Journal of Advanced Robotic Systems [116] (left) and birds flocking while migrating and foraging from The Smithsonian [36] (right).

This dissertation presents a biologically–inspired, multi–agent architecture for binary classification problems using swarm intelligence to aggregate decentralized predictions from multiple sources. Swarm intelligence is common throughout the animal kingdom, allowing relatively simple animals to collectively perform complex tasks and movements through emergent behavior [84]. Figure 1.1 shows fish schooling together, avoiding a predator (left) and birds flocking together while foraging (right). The algorithms and methods presented throughout this document demonstrate significant key advantages over traditional monolithic neural network architectures while maintaining comparable performance in terms of accuracy, precision, and recall. Specifically, this work allows for new, previously unknown features to be incorporated into a prediction problem without retraining the existing set of agents by encapsulating additional features into newly generated agents, saving significant training time as new information becomes available. The multi–

agent design, naturally, allows for distribution across multiple compute nodes, decreasing both training and inference time when additional hardware is available. Work towards both the incremental feature addition and distribution steps led to the development of a "meta-swarm", where distributed information sources and predictions can be encapsulated as agents, collaborating in the classification problem while maintaining the localization and privacy of the source data used while making the initial predictions.

The work presented in this dissertation allows for individual predictions to be incorporated without revealing the (potentially) private data used to make those predictions. Each agent in this method is a small, simple agent built around a core belief; typically the core belief comes from a simple and shallow multi–layer perceptron (MLP) network. However, the belief can also be derived from an external prediction source, forming the basis for the initial prediction for that agent, without revealing any information about how that prediction was formed. Incorporating information and predictions privately from distributed sources is vital to progress within the healthcare industry where there are significant restrictions for information sharing due to patient privacy concerns [15].

The social and aggregation aspects of this work are built around theories from Wisdom-of-Crowds (WoC) which, fundamentally, state that a diverse crowd of non–expert agents (human, computer, or otherwise) can produce predictive results with small predictive error using a sufficiently competent aggregation mechanism [82]. A computer implementation of a honeybee–derived swarm aggregation algorithm is used as a sufficiently competent aggregation mechanism, not only giving a prediction, but also an associated confidence score for the prediction, derived from how quickly the swarm converges on an answer. Results for this method are discussed in detail and compared with more traditional classification methods throughout this document, typically showing similar performance against the existing "state-of-the-art". Additionally, the meta-swarm approach is demonstrated using a real world problem, showing consistent return on investment (ROI) in sports betting over a 20–month period investigating multiple major sports.

Chapter 3 presents a modular multi–agent architecture and environment used to create and train agents, and interact socially; This chapter presents work using this multi–agent architecture, comparing results with traditional classification methods, and demonstrating a custom–designed aggregation mechanism which builds upon a weighted vote method using trust and prior performance. Chapter 4 demonstrates a nature–based aggregation mechanism, based on swarm intelligence, specifically a honeybee optimization algorithm. Figure 1.2 demonstrates the basic steps of a single bee performing a waggle dance. The waggle dance will advertise to other scout bees the direction, relative to the sun, and the distance of the food source, and im-

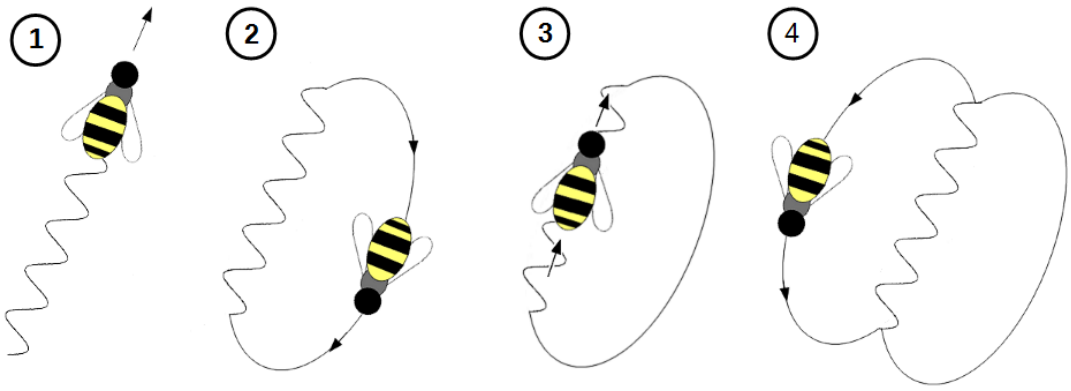

Figure 1.2: Simplified waggle dance process from the Harvey Mudd College Bee Lab [61]. Steps 1 and 3 demonstrate the bee dancing, while steps 2 and 4 demonstrate the bee returning to the starting point

portantly for this work, the quality of the food source. The longer a bee dances, the more bees are recruited to exploit the food source. Similarly, in the computer version presented in this work, agents with stronger prediction beliefs and a history of correct predictions are more likely to recruit additional agents to their prediction. This algorithm improves the predictive performance and improves the stability of the complex system compared with simpler weighted vote and unweighted vote mechanisms. Chapter 5 discusses incremental feature addition and presents results using patient data from Chapter 4 and new cellular morphology features, improving classification performance. I show how the three main components of this method, the initial training, the social interaction, and the swarm aggregation mechanism, can be distributed across multiple hardware nodes in Chapter 6, demonstrating run time performance improvements and presenting the initial development of the meta–swarm method. Chapter 7 presents accuracy, precision, and recall comparisons with the WoC-Bots method, meta–swarm, and other state–of–the–art classification methods. Real world results from applying this research to sports betting predictions over a 20–month period are also presented.

1.1 Scientific Merit

The specific contributions of this research are the following:

- A modular, multi–agent–based architecture for binary classification problems,
- An algorithm that allows for incomplete data with minimal predictive performance degradation,
- A classification algorithm which can be incrementally updated with new, previously unknown features without retraining already trained components,
- A classification system which can be distributed across multiple compute nodes to decrease training and inference time,
- An algorithm that can incorporate predictions from distributed sources – other classification methods, or direct predictions – using a meta–swarm, without compromising the privacy of the original source data.

This dissertation is structured as follows. Chapter 2 presents a chemotaxis–inspired architecture for self–organizing shape formation, based on living cells which emit chemicals into their environment, which are detected by neighboring cells that respond to the chemical concentration by moving along a gradient as needed for complex shape formation [11, 9]. Each agent is simple, and through the use of local–only interactions can be directed to produce a macroscopic, global behavior. An issue with this type of design is determining the correct local interactions without global control. One method used to overcome this limitation in the subsequent work presented in this dissertation is social interaction, a way of disseminating global information with local–only interactions. This work was the impetus to my interest in decentralized, multi–agent architectures and the primary driver behind the swarm–based aggregation mechanism described in Chapter 4.

Chapter 3 introduces the multi–agent architecture and interaction environment. The environment is designed to be modular and flexible, with all aspects of behavior designed as modular algorithms which can be 'plugged in' and modified as necessary. Data is distributed to each agent, which is used to train the agent's internal MLP classifier. Once MLP training is complete, a social interaction arena is generated and each agent is initialized into the the arena based on an initialization algorithm. Moving agents within the arena can be biased to promote more frequent interaction, information dispersal, and prediction formation, which

prepares agents for the opinion aggregation process. All aspects of behavior are enabled through modular algorithms which allow for extensive flexibility and behavioral augmentation to fit specific needs or experimentation. Results using the initial configuration of this architecture are described and presented, showing a robust classification method despite using a dataset with significant missing data.

Chapter 4 presents results classifying breast cancer patients as either node–negative or node–positive, that is, classifying the presence of cancer in the surrounding lymph tissue. Agents with missing data do not participate, but the interaction and swarm steps can continue with the remaining agents. The swarm–based aggregation mechanism presented in this chapter further reduces predictive variance and improves overall stability.

Chapter 5 demonstrates the ability to introduce new features to an existing classification problem without retraining existing agents, a time and computation intensive operation. The variable number of participating agents naturally allows for additional agents to participate, which enables feature addition. New features are included in the classification tasks after the initial training is complete by generating new agents with a combination of new and existing features, and injecting the newly generated agents during the interaction and opinion aggregation phases, leading to improved classification performance during experimentation.

Chapter 6 presents work on distributing the three phases of this method, the initial training, the social interaction, and the swarming phases, across multiple compute nodes. Work has been completed on all three phases. Within this chapter I present the meta–swarm, originally developed as a method to reduce the computational requirements when distributing the swarm phase. This method is described in detail, including results on runtime and classification performance. Experimental results show an overall speedup with a sufficiently large input, and most datasets show good predictive performance when using the meta–swarm method for swarm aggregation.

Chapter 7 contains detailed comparison results, investigating the classification performance of the method described in this dissertation and comparing it with 'state–of–the–art' methods including XGBoost [23], AdaBoost [100], Random forest [19], Logistic Regression [127], and a deep neural network (DNN) implemented in DeepLearning4J (DL4J). These methods were selected for comparison based on their state–of–the–art status and because they are commonly used in binary classification tasks [134]. All methods were tested on three datasets, the Hollywood data described in chapter 3, the breast cancer data described in chapter 4 and an additional dataset predicting airline passenger satisfaction, described in chapter 7. Included in the comparison results is an additional use of the meta-swarm, using predictions from WoC-Bots and the

other classification methods to improve the variance between accuracy, precision, and recall. Also included in this chapter are real world results from applying this method to sports betting from January 2021 through November 2022, across most major sports, focusing on moneyline and spread bets.

Chapter 2. Chemotaxis-based self-organization

The work presented throughout this dissertation is largely based in biological behavior, at the cellular, insect, and human level. This chapter presents work based on the cellular level, an agent–based approach for self–organizing shape formation which is inspired by the chemotaxis of living cells. Cells emit chemicals into the environment and are capable of detecting and responding to the overall chemical concentration within the environment. As with cells, the agents presented here are simple and use local-only interaction to direct complex, macroscopic behavior.

2.1 Introduction

Motivated by the ability of cells to form into specific shapes and structures, in previous work we developed chemotaxis-inspired software agents for self-organizing shape formation [11, 9]. The actions of the agents, which we call Morphogenetic Primitives (MPs), are based on the behaviors exhibited by living cells. Cells emit chemicals into the environment. Neighboring cells detect the overall chemical concentration at their surfaces and respond to the chemical stimulus by moving along the chemical field's gradients [33]. Similarly, in our system the agents emit a virtual chemical, with its concentration defined by an explicit mathematical expression. A set of agents start with an initial random configuration and stochastically follow the gradient of the cumulative concentration field. These chemotaxis-based local interactions direct the agents to self-organize into user-specified shapes (Figure 2.1). Since the behaviors of MPs are based on local information and interactions, they could provide a distributed, scalable approach for controlling the movements of a robotic swarm.

In some cases, we have observed though that the agents do not spatially self-organize into a unique shape, but instead form two or more stable final configurations. If MPs are used to control the motions of individ-

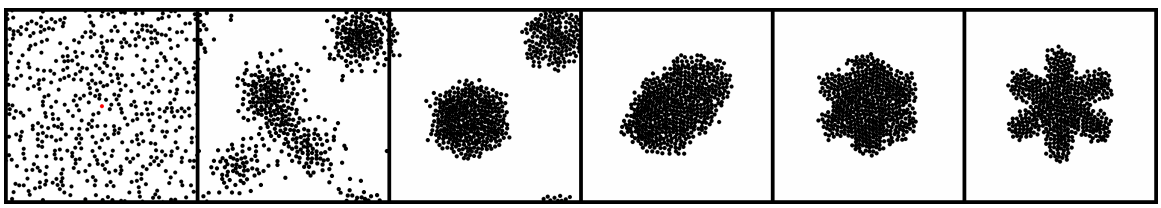

Figure 2.1: Morphogenetic Primitives self-organizing into a star shape. Initially published in [9].

ual robots, it would be extremely useful to direct the outcome of these bifurcating spatial self-organization processes towards a consistent outcome. This would allow us to guarantee that all of our MP aggregations produce a single, desired shape. This is a property that is essential for robust and predictable swarm control algorithms, one that would make the algorithm reliable for engineering applications. Towards this end, we have analyzed whole swarm populations at a global level in search of macroscopic, distinguishing attributes. This analysis identified features, based on statistical moments of the agents' positions, that have significantly different values for different outcomes of a swarm aggregation. In earlier work we discovered that these statistical moments can be used to accurately predict the outcome of the self-organization process at an early stage of the shape aggregation [12]. Given these differentiating moments, the work described here investigated techniques for directing the outcome of our self-organizing system via biased, random initial conditions in order to consistently produce a desired final configuration.

Through our study of the dynamics of a swarm's statistical moments during the aggregation process we noted the connection between initial conditions and the final shape configuration of the swarm. We discovered that biased, random initial conditions that meet specified constraints, i.e. have well-defined statistical properties, robustly yield simulations with a unique final outcome. For those agent interactions that ultimately produce bifurcating/multiple shapes, we have identified for each shape, the most distinguishing macroscopic statistical moment of the evolving swarm. It is possible to generate random distributions of MPs that have specific statistical moments. Our work empirically shows that for bifurcating self-organizing, non-linear, dynamical systems (e.g. a swarm of Morphogenetic Primitives) one final outcome can be consistently generated by enforcing a constraint on the value of a single moment when generating the swarm's initial conditions. Given this feature of our system, we are able to control the final outcome of the simulation by simply thresholding the value of a statistical moment for a particular starting distribution, i.e., we constrain the random initial conditions to have specific statistical properties.

2.2 Related Work

Research on distributed agent-based systems that can form spatial patterns and shapes, as well as swarm behaviors, has been conducted for several decades. Reynolds [91] proposed the seminal model for simulating flocking and schooling behaviors based on the local interactions of "boids". Fleischer and Barr [38, 37] explored a cell–based developmental model for self–organizing geometric structures. Theraulaz and Bonabeau [113, 114] presented a modeling approach based on the swarming behavior of social insects. They

combined swarm techniques with 3D cellular automata to create autonomous agents that indirectly interact in order to create complex 3D structures. Viscek et al. [121] investigated a particle-based model related to the Reynolds model and found that macroscopic phase changes occurred in the particle system when introducing noise in the local interactions. Jadbabaie et al. [54] further explored the Vicsek model and provided a theoretical explanation for the model's observed behavior, as well as convergence results for classes of switching signals and arbitrary initial heading vectors.

The initial work in this area of research created distributed, locally-interacting agent-based systems, then observed and characterized their behaviors. Later work explored techniques for directing these distributed, self-organizing systems. Eggenberger Hotz [32, 51] proposed the use of genetic regulatory networks coupled with developmental processes for use in artificial evolution and was able to evolve simple shapes. Bonabeau et al. [21] applied genetic algorithms to the stigmergic swarm-based 3D construction method of Theraulaz and Bonabeau in order to evolve interactions that produce user-acceptable structures. Nagpal et al. [73, 74] presented techniques to achieve programmable self-assembly. Cells are identically-programmed units which are randomly distributed and communicate with each other within a local area. In this approach, global-to-local compilation is used to generate the program executed by each cell, which has specialized initial parameters. Stoy and Nagpal [107] presented an approach to self-reconfiguration based on directed growth, where the desired configuration (which is stored in each module) is grown from an initial seed module. Spare modules move along recruitment gradients emanating from attached modules to create the final shape. Gradients derived from global potential fields have also been investigated for directing robot swarms. Both Rimon and Koditschek [93] and Hsieh and Kumar [52] demonstrated that robot paths and controls can be computed from these fields, which lead the robots to form a pre-defined shape.

Shen et al. [103] proposed a Digital Hormone Model for directing robot swarms to perform such tasks as surrounding a target, covering an area and bypassing barriers. The model relies on local communications between identical agents, but it also has each agent move towards a single global target. Swarm chemistry, proposed by Sayama [99, 98] and based on Reynolds' model, is an approach for designing spatio-temporal patterns for kinetically interacting, heterogeneous agents. An interactive evolutionary method, similar to Sims' [104], has been used to define system parameters that lead to agent segregation and structure formation. Mamei et al. [71] proposed a distributed algorithm for robots that are attracted to and aggregate around targets sensed over short distances. By electing leader(s) as barycenter(s), propagating gradients of varying structure and using these gradients as instruction conditionals, a swarm of simulated robots are able to self-

organize into a number of simple shapes such as a circle, ring, and lobes. Von Mammen and Christian [123] described swarm grammars, an agent-based extension of Lindenmayer systems, that are capable of adapting to their environment and evolve agent parameters in order to create structures that incorporate aspects of developmental design and morphogenesis. The field of Guided Self-Organization (GSO) [88] has developed techniques for steering self-organizing systems towards desired outcomes, while still attempting to not constrain the system's configuration space during its evolution.

Doursat [29, 28] proposed a model for artificial development which combines proliferation, differentiation, self-assembly, pattern formation and genetic regulation. Via genetic-like regulation at the agent level, the agents can self-organize into a number of patterned shapes and structures. Werfel et al. [124] proposed a decentralized multi-agent system approach, inspired by mound-building termites, for building user-defined structures. A user specifies a desired structure, and the system automatically generates low-level rules for independent climbing robots that guarantee production of the structure. A single "seed" brick is used as a landmark to identify where the structure is going to be built, and defines the origin of a shared coordinate system for the robots. Gerling and Von Mammen [41] provided a context for this type of work in a summary of self-organized approaches to construction.

Our previous and latest work on agent-based shape formation stands apart from related work in that it utilizes a chemotaxis-based interaction paradigm inspired by the behaviors of living cells which leads to the formation of tissues. Given the goal of recreating the properties of cells, MPs were designed with principles that should make them scalable and robust [9]. These design principles define MPs as identical, distributed agents that are not directed by a 'master designer', exchange information locally, carry no representation of the shape to be formed, and have no information about their global location. The macroscopic shape of the swarm emerges from the aggregation of local interactions and behaviors. Our approach is therefore novel compared to previous work in that it contains all of the following features. 1) All morphogenetic primitives are randomly placed in the environment, are identical, and perform the same simple actions, unlike Nagpal et al. [73, 74]), Mamei et al. [71], and Doursat [29, 28]. They require no differentiated behaviors or customized initialized states. 2) No initialization of spatial information is needed in the computational environment, unlike Stoy and Nagpal [107], Shen et al. [103], and Werfel et al. [124]. 3) Individual MPs do not know their location in any external/global coordinate system, unlike Stoy and Nagpal [107], and [124]. 4) MPs do not contain or utilize a representation of the predefined global shape that is being composed, unlike Stoy and Nagpal [107], Rimon and Koditschek [93], and Hsieh and Kumar [52]. 5) We utilize genetic

programming to discover the MP concentration field functions that lead to the formation of a user-specified shape. Chemotaxis then provides a straightforward mechanism for determining the motion of MPs, in contrast to the difficult-to-program approaches of Shen et al. [103] and Sayama [99, 98]. Note that our new work described here enhances our previously developed system [9] to make MPs more robust, consistent and reliable.

2.3 Background Material

2.3.1 Agent-based Shape Formation

Like Pfeifer et al. [87] we turn to biology and self-organization for insights into the design of autonomous robots, robotic swarms in our case. Our previous work in self-organizing shape formation [9, 13] is inspired by developmental biology [42] and morphogenesis [25], and builds upon a chemotaxis-based cell aggregation simulation system [35]. Morphogenesis is the process that forms the shape or structure of an organism through cell shape change, movement, attachment, growth and death. We have explored chemotaxis as a paradigm for agent system control because the motions induced by chemotaxis (one of the mechanisms of morphogenesis) may produce patterns, structures or sorting of cells [102].

Morphogenetic Primitives are initially placed inside a $2D$ environment with a random uniform distribution. Each MP is represented by a small disc and emits a 'chemical' into the environment within a fixed distance relative to its own local coordinate system. Every MP emits the identical local chemical field. An MP detects the cumulative chemical field at eight receptors on its surface, and calculates the field gradient from this input. MPs move in the direction of the field gradient with a speed proportional to the magnitude of the gradient. By employing these relatively simple chemotaxis-inspired behaviors MPs are able to self-organize into user-specified macroscopic shapes.

This process is schematically presented in the bottom $2/3$ of Figure 2.2. In the middle left of the figure, a close-up of an MP is provided showing its eight chemical sensors and the range of the finite chemical field that it emits. Numerous MPs are randomly placed in the computational arena, and are provided as initial conditions to a chemotaxis-based cell aggregation simulator. The simulator then computes an aggregation simulation based on the one chemical field that is associated with all MPs. The self-organization of the agent swarm is shown in the middle right of the figure. The bottom flowchart of the figure outlines the steps taken by the cell simulator for each cell. The bottom left image shows a representative MP chemical field, with the chemical concentration visualized with gray-scale colors. Isolines are added to highlight the structure of the

chemical field. The image to its right shows a cumulative chemical field given the contributions of all of the MPs in the arena. The top part of the figure will be explained later in the chapter.

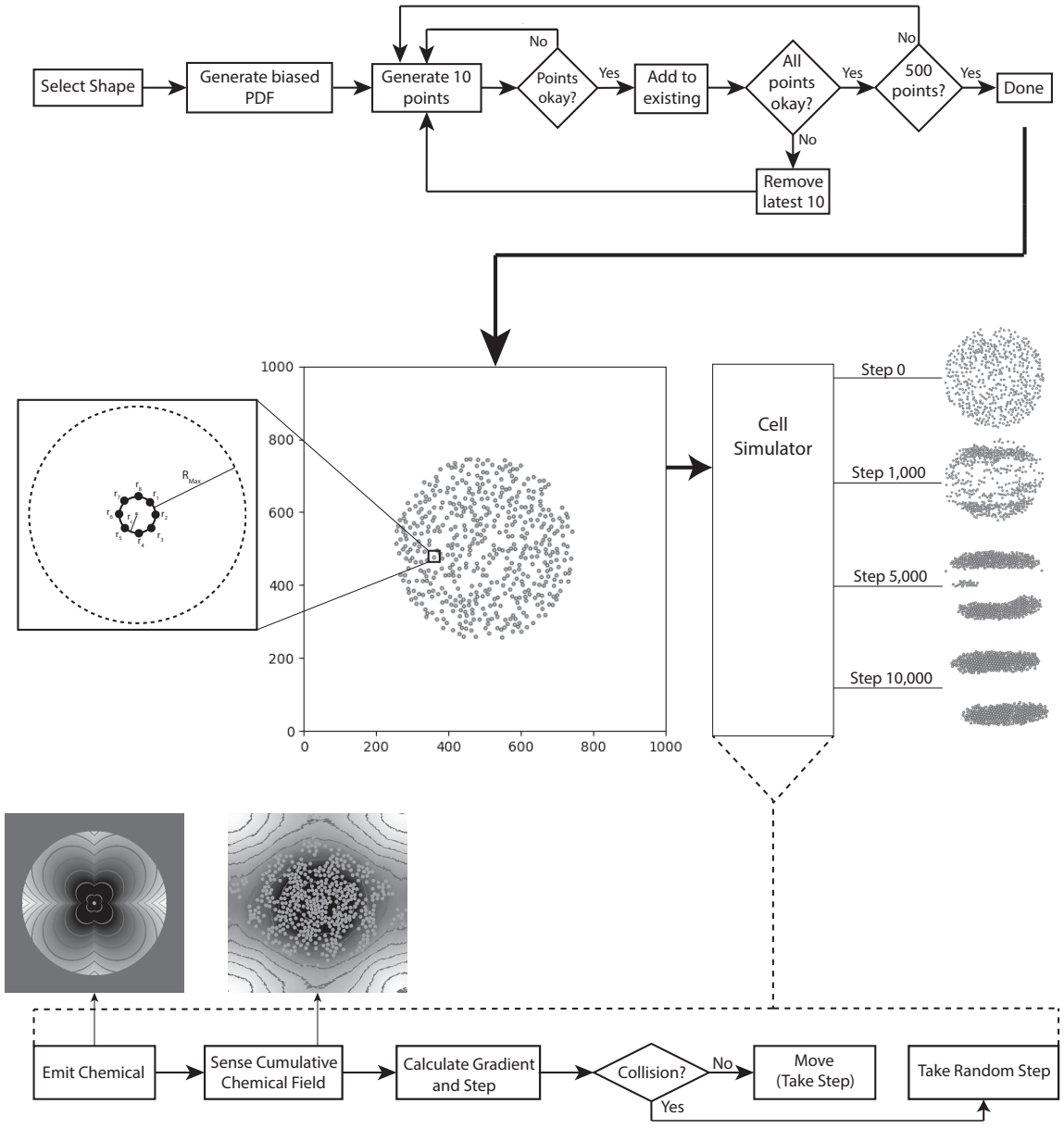

Figure 2.2: Schematic diagram of the directed self-organization process based on specifying biased initial conditions for Morphogenetic Primitives.

While MPs' fundamental interactions are based on a chemotaxis-inspired paradigm, we do not limit their behaviors/properties to be physically realistic or completely consistent with biology. Instead, developmental biology provides a motivating starting point for MPs. As a way to customize chemotaxis-inspired agents for

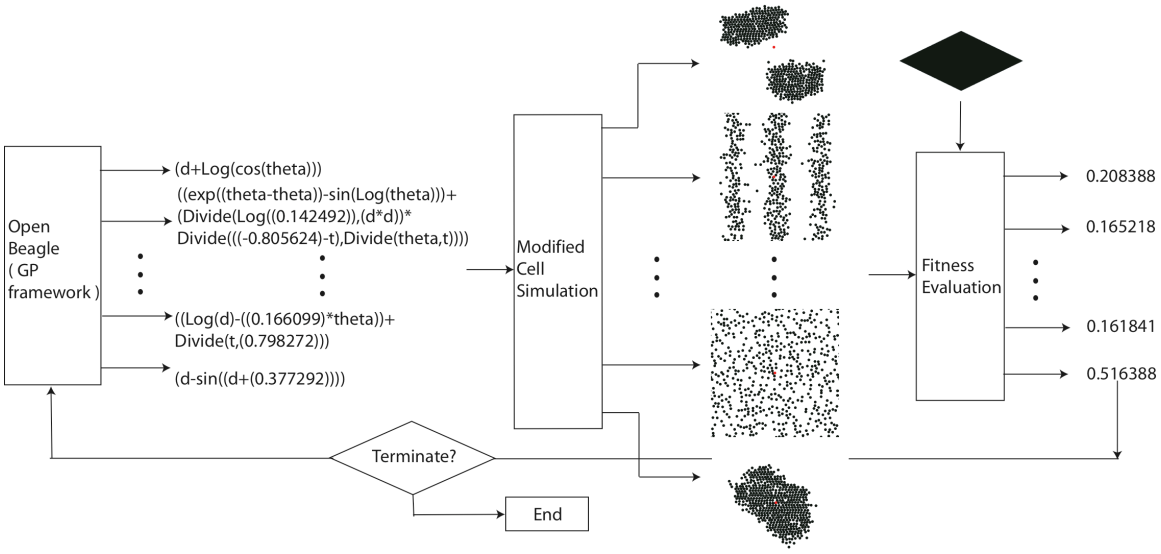

Figure 2.3: The genetic programming process that produces the local chemical field functions of the shape primitives. Initially published in [11].

shape formation, we alter the chemical concentration fields around individual cells. Instead of the chemical concentration dropping off only as an inverse function of distance d from the cell's surface (e.g. $1/d$), in our system we define the concentration field with an explicit function of d and θ, the angular location in the cell's local coordinate system.

Currently, there is no prescriptive way to specify a particular local field function that will direct MPs to form a specific macroscopic shape, we therefore employ genetic programming [59] to produce the mathematical expression that explicitly specifies the field function. In order to meet the substantial computational requirement imposed by our evolutionary computing approach, we have implemented a master-slave form of the distributed genetic programming process [11]. The fitness measure associated with each individual field function is based on the shape that emerges from the chemical-field-driven aggregation simulation, and determines which functions will be passed along to later generations. The genetic process stops once an individual (i.e., a mathematical expression) in the population produces the desired shape via a chemotaxis simulation, or after a certain number of generations have been produced and evaluated. Figure 2.3 illustrates this approach. See [11], [9] and [8] for more details on MPs and the software system that implements them.

With this algorithm, we have successfully evolved local MP chemical field functions for a number of simple shapes [9]. These results support the proposition that biological phenomena offer paradigms for designing cellular primitives for self-organizing shape formation. While the resulting explicit chemical fields are

not biologically/chemically plausible, they do provide an approach for controlling robot swarms that communicate wirelessly over short distances and share minimal information with each other. Thus the agents in the swarm do not require significant compute power to self-organize. Additionally, evolutionary computing techniques, specifically genetic programming, have been crucial for discovering the detailed local interactions that lead to the emergence of the swarm's macroscopic structure.

However, given the MPs' initial random configurations and the stochastic nature of the self-organization process, the outcomes of the simulations with a specific field function are not always the same. We have found that the shape formation simulations, which include random displacements of the MPs and noise in their movements, can generate bifurcating results. For some field functions, if we run numerous simulations each starting with a different random uniform distribution of MPs, two sets of final configurations will be formed. In most cases an equal number of each configuration are produced, but in a few cases the ratio of the numbers is not one. Since it would be useful to control the outcomes of the self-organization process, we have developed methods for directing the final configuration of a bifurcating simulation by starting the simulation with biased initial conditions [7].

2.3.2 Outcome Prediction

The first step towards developing methods that direct the outcome of a swarm simulation involved identifying spatial features that are correlated with and can differentiate the final, different swarm configurations. Our initial effort towards achieving this goal investigated methods for predicting the final configuration of a bifurcating simulation at an early stage of the aggregation process. Our reasoning was that if certain spatial features can be used to predict the outcome of an aggregation, then they represent unique attributes of the swarm that could be manipulated to direct the swarm. In order to predict the final outcome of a self-organizing shape formation simulation, we first extracted features that capture the spatial distribution of the MPs. Moments provide a quantitative way to describe a distribution. Since MPs are defined as small discs, we use the center of each disc to represent each MP's location. We therefore can simplify the collection of MP locations as a set of $2D$ points, and apply moment analysis to this set over the duration of the MP simulation.

We calculated the mean (first moment), variance (second central moment), skewness (third central moment) and kurtosis (fourth central moment) from the x and y coordinates of the MP centers. We analyzed the locations X_i of all points (MPs) as a whole, rather than tracking the location and movement of each in-

dividual point. The population size of the agents is denoted as n, $(n = 500)$, and the formulas of the four moments M_1 to M_4 are given in Equations 2.1 to 2.4,

$$M_1 = \frac{1}{n}\sum_{i=1}^{n} X_i, \tag{2.1}$$

$$M_2 = \frac{1}{n}\sum_{i=1}^{n} (X_i - M_1)^2, \tag{2.2}$$

$$M_3 = [\frac{1}{n}\sum_{i=1}^{n} (X_i - M_1)^3]/(M_2)^{3/2}, \tag{2.3}$$

$$M_4 = [\frac{1}{n}\sum_{i=1}^{n} (X_i - M_1)^4]/(M_2)^2. \tag{2.4}$$

These statistical moments provide quantitative information about the shape of histograms/distributions. When computed for the x and y coordinates of the MPs, these moments capture the asymmetry and shape of the spatial distribution of the whole population. We have not found it necessary to compute cross moments, with the first four moments providing sufficient information for our analysis. Since the x and y coordinates of the points change over time, so do the four moments of the distribution of the x and y values. The change of the moments as a function of simulation time also provides insight into the dynamic nature of a particular MP simulation.

At each simulation time t, the four moments $M_i(t)$ $(i = 1\ to\ 4)$ of the overall distribution are calculated and we then approximate the time derivative of the moments as the slope of a linear interpolating function of consecutive moment values. By calculating the moments and their time derivatives for both the x and y coordinates of the point set, at a given time t, we obtain a 16-dimensional vector to represent the distribution,

$$M_{x_1}(t), M_{y_1}(t), M_{x_2}(t), M_{y_2}(t), M_{x_3}(t), M_{y_3}(t), M_{x_4}(t), M_{y_4}(t), k_{x_1}(t), k_{y_1}(t),$$

$$k_{x_2}(t), k_{y_2}(t), k_{x_3}(t), k_{y_3}(t), k_{x_4}(t), k_{y_4}(t).$$

Given the sensitivity of non-linear dynamical systems to initial conditions [125], it makes it extremely difficult, if not impossible, to predict the outcome of our complex, self-organizing system from its initial, random spatial configuration. We therefore attempted to predict the final spatial configuration at an early stage of the aggregation, usually before it is visually evident what shape will emerge from the process. We considered prediction of the bifurcating outcomes as a classification problem and utilized support vector machines (SVMs) [24] to solve it. We have found that applying SVMs to the distribution feature vector at

a simulation time that is a small percentage of the total time needed for the final aggregated shape to form produced acceptable results. Given 200 MP simulations for a variety of bifurcating self-organizing shapes we found that we could predict the outcome of the aggregation at a time point 5% to 10% into the simulation with an accuracy of 81% to 91%. We view these results as satisfactory because they demonstrate that a strong correlation between a swarm's moments and its final formed shape does exist. More details about this study may be found in [7] and [12].

2.4 Directing Spatial Self-Organization

Since the outcome of an MP simulation can be predicted at an early stage of aggregation using the moments of the agents' positions, we then explored methods for controlling the swarm via manipulating the moments of the swarm's initial configuration. The general strategy is to create random initial configurations for the MP simulations, but with constrained, biased moments. We have observed that this strategy can consistently direct the swarm to aggregate into specific final configurations [45]. The first step of the strategy analyzes the bifurcating simulations to determine which of the moments diverge the most for the two different outcomes. This is the moment that will be biased in the swarm's initial, random configuration.

2.4.1 Moment Analysis

One of our aggregations, which produces what we call the quarter-moon shape, provides an example of the moment analysis. Of the 200 simulations starting with a uniform random, unbiased initial condition, 100 produce left-pointing structures and 100 produce right-pointing structures. Figure 2.5 shows a typical swarm aggregation for this shape. The four moments of both x and y coordinates are calculated over all 35,000 simulation steps. Additionally the mean and standard deviation of each statistical moment are calculated for the two categories, i.e. left-pointing and right-pointing, over the simulation time steps. Plotting the mean and the mean $\pm$ standard deviation of the moments over time immediately highlights the moments which are the most differentiating and may be used to identify specific shapes. For the quarter-moon example the third x moment (skewness) is the one with the greatest separation of values for the two possible outcomes, as seen in Figure 2.6. The solid and dashed lines are the mean of the skewness of the x coordinate for the two outcomes. The dotted and dot-dashed lines are mean $\pm$ standard deviation. The dashed curve is produced from structures that are right-pointing and follow the path in the top of Figure 2.5. The solid curve is produced from the left-pointing structures, with a typical aggregation presented at the bottom of

Figure 2.5.

By analyzing the time series in Figure 2.6, we see that the skewness of the x coordinates of the two classes starts at about the same value, approximately -0.05 to 0.05, at time step 0. The values should be near zero, since all simulations begin with uniform random configurations. The skewness of the two classes first separates by increasing or decreasing, followed by a zero crossing and then a reversed trend appears until they reach their final states at step 35,000. Observing the values for the solid and dashed curves over all simulation time steps, we can identify three regions in the plot: a region occupied by solid/dotted curves only, a region occupied by dashed/dot-dashed curves only and an overlapping region. To be specific, considering the values of x skewness in Figure 2.6 (by projecting the curves into the y axis), the range of $[-0.195, -0.150]$ is covered by solid/dotted curves only; $[0.150, 0.190]$ is covered by dashed/dot-dashed curves only and $[-0.150, 0.150]$ is covered by both outcomes.

When determining the appropriate threshold value for a constrained moment, we start with the mean value of the non-overlapping moment range for a particular shape, and then adjust if needed. For the region covered only by the quarter-moon's dashed/dot-dashed curves (for right-pointing shapes) the mean value of skewness is 0.170. While using this value for the moment constraint produced reasonable results (94% of the biased initial conditions produced the desired shape), we found, via multiple experimental runs, that the threshold value on the skewness had to be increased to produce a consistent result. In general this was the process employed for determining the moment constraint thresholds needed to generate the desired outcomes.

2.4.2 Generating Constrained Biased Distributions

Once the most significant distinguishing moment (the one with the greatest value difference in the final configuration) for a shape is identified, the information is utilized to direct the shape aggregation by imposing constraints on this moment in the initial conditions. We assume that the MPs' x and y coordinates are independent, and therefore create two probability density functions, each representing the x and y coordinate. One probability density function is created for the constrained coordinate and the other coordinate is considered to be uniformly random. Samples are drawn independently from the two distributions to produce a single (x, y) location.

This approach generates random distributions, i.e. 2D random initial configurations for the shape simulation, that meet a constraint on a particular moment in the x or y coordinate. We have found that constraining

one of the eight moments (mean, variance, skewness and kurtosis for x and y) is sufficient for producing satisfactory results. Constraining multiple moments does not significantly improve the outcomes, and would further complicate the process of generating initial conditions. Once one significant moment of a distribution and its threshold value have been identified, the remaining three moments for that spatial coordinate (x or y) may be set to values observed in uniform random distributions. These values are $mean = 500$ (the center of our computational arena), $variance = 15,000$, $skewness = 0$ (to make the distribution symmetric), and $kurtosis = 2$. Once the four moments for one of the coordinates have been specified, a probability density function (PDF) with those moments is defined. The values for the other coordinate are generated from a uniform random distribution.

For the constrained dimension we create a Gram-Charlier expansion of the normal distribution (chosen for its convergence properties) with specified moments [101]. This Gaussian-expanded probability density function is then discretized with 100,000 samples with values falling within the range of 0 to 1000 (the range of the computational arena). Slice sampling [77], a Markov chain sampling method chosen for its efficiency, is then utilized to draw 500 samples from this discretized distribution. Theoretically, the specified PDF can be sampled to produce a distribution that has the same moments as the PDF. Our experience has shown that the sample size needs to be quite large (on the order of 1 million) for this to be true. For our sample size of 500 (the number of MPs in an aggregation simulation) and heavily biased PDFs, the resulting moments of the finite sample set do not necessarily match the ones desired for a particular shape.

As the distribution generation process cannot guarantee that a sampled distribution will meet the required moment constraints, the four moments of the generated distributions for the constrained coordinate are computed to determine if the constraint is actually met. If the value of the significant moment is not above or below (depending on the constraint to be enforced) the specified threshold, the sampled distribution is rejected. If the constraint is met, the distribution is accepted as the initial conditions for a simulation computation.

Since our agents are not points, but in fact are discs with a fixed radius, we maintain a distance of $2R$ (where R is the radius of a disc) between sample points, to ensure that the MPs do not overlap. Therefore, if an (x, y) pair is generated that is less than $2R$ distance to a previously generated location, it is rejected and another (x, y) pair is calculated. This process continues until a sufficient number of MP locations are generated for the initial conditions.

We have found that it is more computationally efficient to generate numerous smaller sample sets and

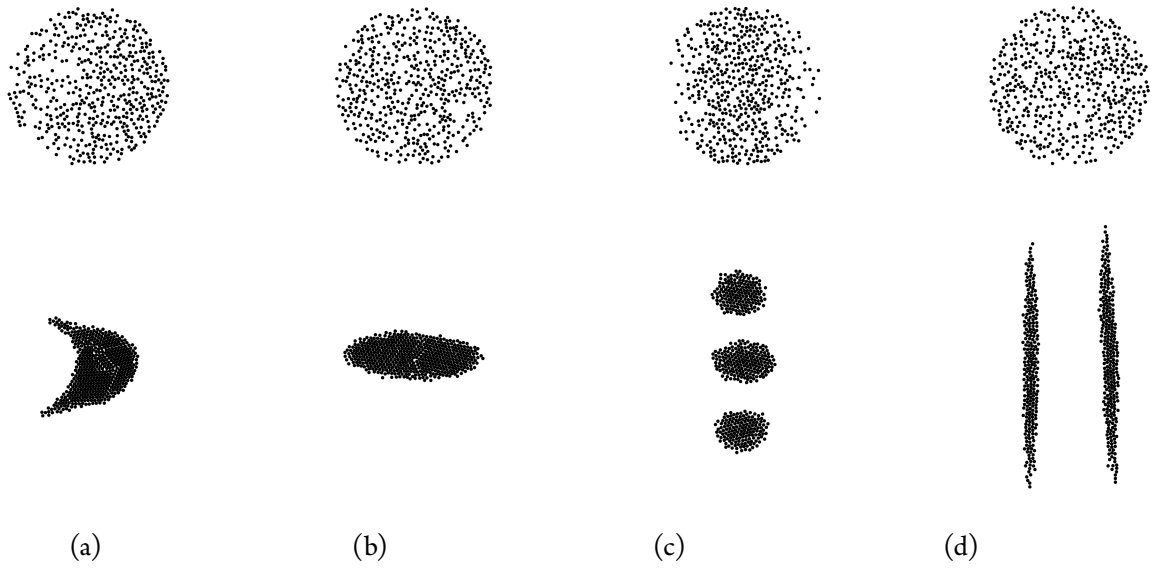

Figure 2.4: (top row) Biased initial conditions ((a) x skewness = -0.315, (b) y kurtosis = 2.150, (c) x variance = 9,596, (d) x kurtosis = 1.88) that robustly evolve (bottom row) into (a) a right-pointing quarter-moon, (b) a single ellipse, (c) three discs and (d) two line segments.

then merge them into a single point set, rather than attempt to compute a single, large point sample, when composing biased initial conditions for our aggregation simulations. We apply this approach by drawing 50 subsets, $T_{k=1,\ldots,50}$ of 10 samples, rather than 1 set of 500 samples. Each sample drawn is checked for overlap with existing samples, discarded if overlap exists, and otherwise added to the current T_k. Once each T_k has 10 samples, it is checked for conformance to the moment restrictions prior to insertion into the final set, S, of 500 samples. If a given T_k fails, a new T_k is drawn. An acceptable T_k is merged with S, which is then checked for conformance to the moment restrictions. If the updated S does not pass, it is reverted to its prior state, $S - T_k$, and a new T_k is drawn. This process continues until $|S| = 500$. The algorithm for generating biased initial condition, once the initial sampling does not meet the moment requirements, is diagrammed in the flowchart at the top of Figure 2.2. Via this approach, we are able to generate initial conditions for our computational experiments in a few seconds, as opposed to several hours, when attempting to generate all 500 points of S at once for certain "extreme" biased conditions, e.g. low kurtosis.

2.5 Results

We have applied our method for directing spatial self-organizations, which generates biased initial configurations, to a number of bifurcating shape aggregations. We refer to the resulting shapes as the quarter-moon, ellipse, discs, and two parallel line segments. These shapes (and their associated chemical fields) were uti-

Shape	Field Function
quarter-moon	$1.0/(\ln(\ln(\ln(exp(\sin(\ln(\cos(\theta) * (((d + 0.761214) * \ln(d)) + \ln(d - \theta)))) * (\theta - e^{\theta}))) - ((\theta - d) * (d/\theta))) - ((\theta - e^{\theta}) * \ln(\ln(exp(\sin(\ln(\cos(\theta) * (((d + 0.761214) * \ln(d)) + \ln(d - \theta)))) * (((\theta - e^{\theta}) - ((\theta - e^{\theta}) * (d/\theta))) + 0.432846))) - ((\theta - d) * (d/\theta))))))$
ellipse	$1.0/\cos(\theta) + \ln(d)$
discs	$1.0/(\cos(\sin(\theta) - (\cos(\theta) - (\ln(-0.367378)/(\theta + d)))) + \ln(d))$
lines	$1.0/(\ln(\ln(\ln((\cos(\cos(d)) + (d * (3d + \ln(\theta) + \ln(\ln(\theta))))) + (((\ln(\theta)/(0.285192)) * ((1.0/\theta) + 0.423969))/d)) + d) + d))$

Table 2.1: The field functions for the MPs utilized in this study. Note that exp(x) signifies e^x.

lized in an earlier study [12], and had shown not to produce a single final, aggregated result. In our initial MP work it was not uncommon for the output of the evolution process to generate chemical fields that led to the formation of different shapes from a single field. These four were chosen because they generated two different shapes in equal proportions (except for the parallel lines shape) from uniformly random initial conditions. The chemical field functions that direct MPs to form into these shapes are detailed in Table 2.1. Given biased initial conditions the aggregation simulations produce one final outcome in almost all of cases. Moreover, by thresholding the moment constraints on the biased initial conditions it is possible to control which shape is produced by a simulation. Figure 2.4 (top row) shows biased initial conditions for a number of shape aggregations created with this method. The bottom row illustrates the final outcome of each MP simulation that is produced from the associated biased starting configuration.

In order to identify the significant, distinguishing moments for each shape, aggregation simulations (usually several hundred) with unbiased initial conditions are first performed. The shapes of the final outcomes are visually inspected and placed into categories. For each final shape, the mean and standard deviation of the four statistical moments of the evolving system are computed over the entire shape aggregation process. For

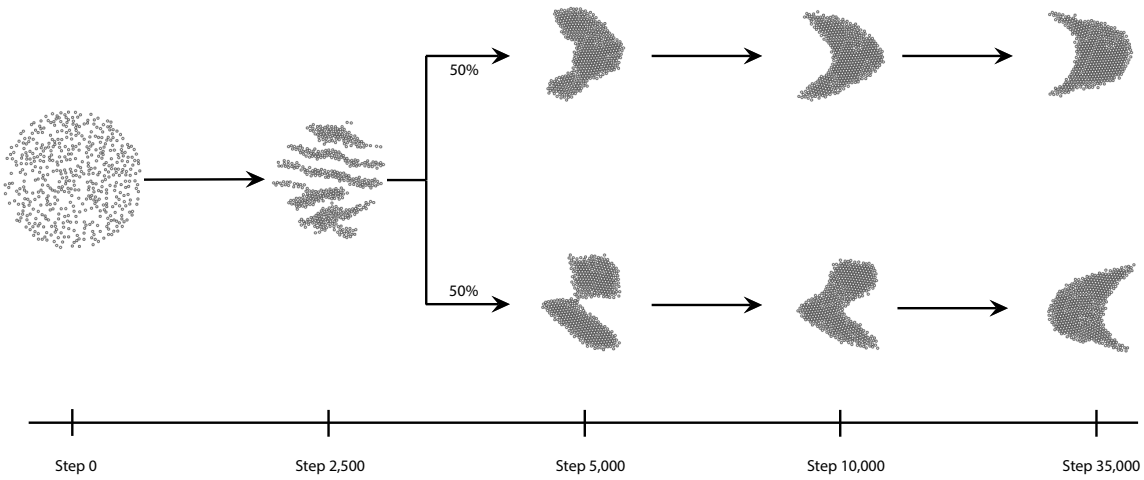

Figure 2.5: Shape aggregation of the quarter-moon MPs starting from random initial conditions.

simulations starting with uniform random, unbiased initial conditions, the final outcomes of the quarter-moon , ellipse, and discs shapes evenly split into two categories, with roughly 50% of the final outcomes belonging to each class. The outcomes of the parallel line shapes are unbalanced, with 84.1% belonging to the majority class (two lines) and 15.9% belonging to the minority class (one line). These results, as well as the details that follow in the remainder of the section, are summarized in Table 2.2.

A typical unbiased aggregation for the quarter-moon shape is shown in Figure 2.5. The simulation reaches a stable state by 35,000 simulation steps. We identified skewness in the x coordinate to be the significant macroscopic feature for this shape. See Figure 2.6 for the evolution of this feature over the course of the aggregation given uniform random initial conditions. Two sets of biased initial conditions (each with 100 examples) were generated with constrained skewness values in the x coordinate, with the thresholds set as greater than 0.315 (2 standard deviations from the mean) and less than -0.315. Since the distributions are generated stochastically, they do not have the exact targeted skewness value. So our acceptance test is based on a threshold. We performed simulations with the quarter-moon interaction function for these 200 biased initial conditions. Of the 100 initial conditions with a thresholded third x moment below -0.315, 100% of the final outcomes are right-pointing structures. Of the 100 initial conditions with a thresholded third x moment above 0.315, 100% are left-pointing structures. Figure 2.7 presents the evolution of this feature over the course of the aggregation given the biased initial conditions.

A typical unbiased aggregation for the ellipse shape is shown in Figure 2.8, with half of the unbiased initial conditions producing a single "perfect" ellipse, with the other half producing either two ellipses or a de-

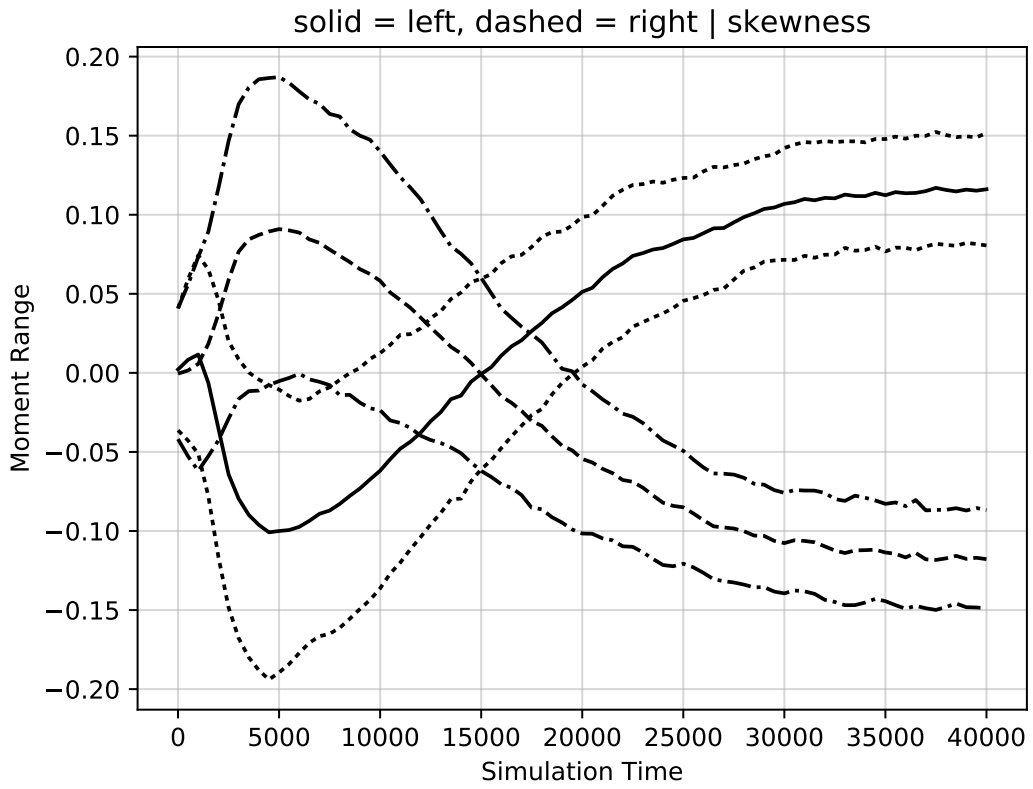

Figure 2.6: Skewness of the x coordinate of the unbiased quarter-moon shape aggregations over time.

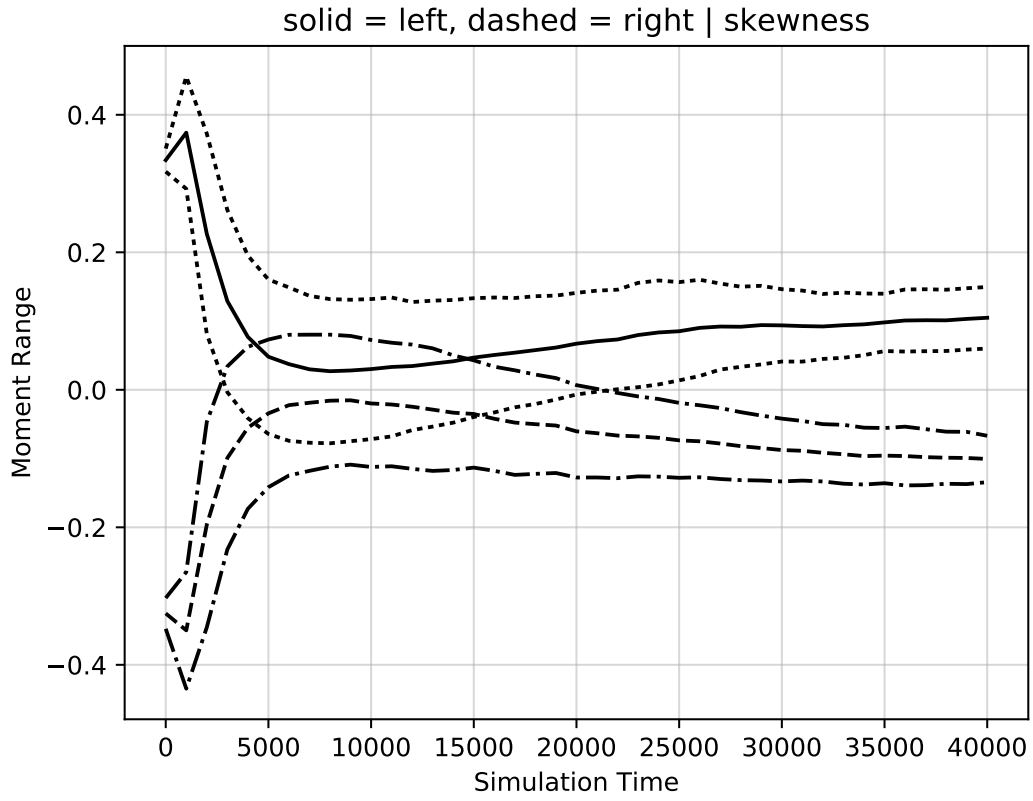

Figure 2.7: Skewness of the x coordinate of the biased quarter-moon shape aggregations over time.

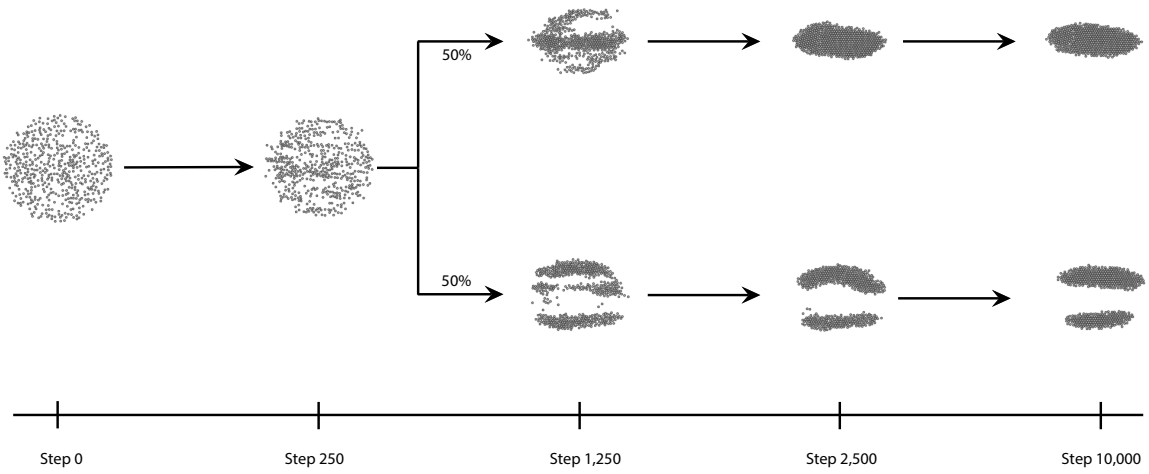

Figure 2.8: Shape aggregation of the ellipse MPs starting from random initial conditions.

formed "blob". The unbiased simulation is computed for 10,000 steps. If the simulations are run for 50,000 steps they all will produce a single ellipse. Kurtosis in the y direction was found to be the significant macroscopic feature for this shape. See Figure 2.9 for the evolution of this feature over the course of the aggregation given uniform random initial conditions. 100 simulations were performed with initial conditions that had their y coordinates' kurtosis thresholded to be above 2.18. Given these biased initial conditions all simulation (100%) produced a perfect single ellipse by step 7,500. 100 additional simulation were performed with initial conditions that had their y kurtosis set below 1.85. All 100 simulations produced two ellipses by step 7,500. Figure 2.10 presents the evolution of this feature over the course of the aggregation given the biased initial conditions. These results show that not only can biased initial conditions direct the outcomes of the simulations, but they can also significantly speed up the formation of the desired result, with the single ellipse being guaranteed to form by 50,000 steps in the unbiased case and by 7,500 steps given biased initial conditions.

The discs dataset, when run with 200 unbiased initial conditions, produces 100 four discs structures and 100 structures of five or more discs, with a typical shape aggregation shown in Figure 2.11. The simulation reaches a stable state by 15,000 steps. We identified variance in the x direction to be the significant macroscopic feature for this shape. See Figure 2.12. In our experiments as the variance of the initial conditions was lowered, we found that we could generate a new shape (one that did not appear with unbiased initial conditions), that contained only three discs. Thresholding the x variance of the initial conditions to be less than 10,270 would always (100%) produce a 3-disc result. A typical 3-disc shape is presented in Figure 2.4(c).

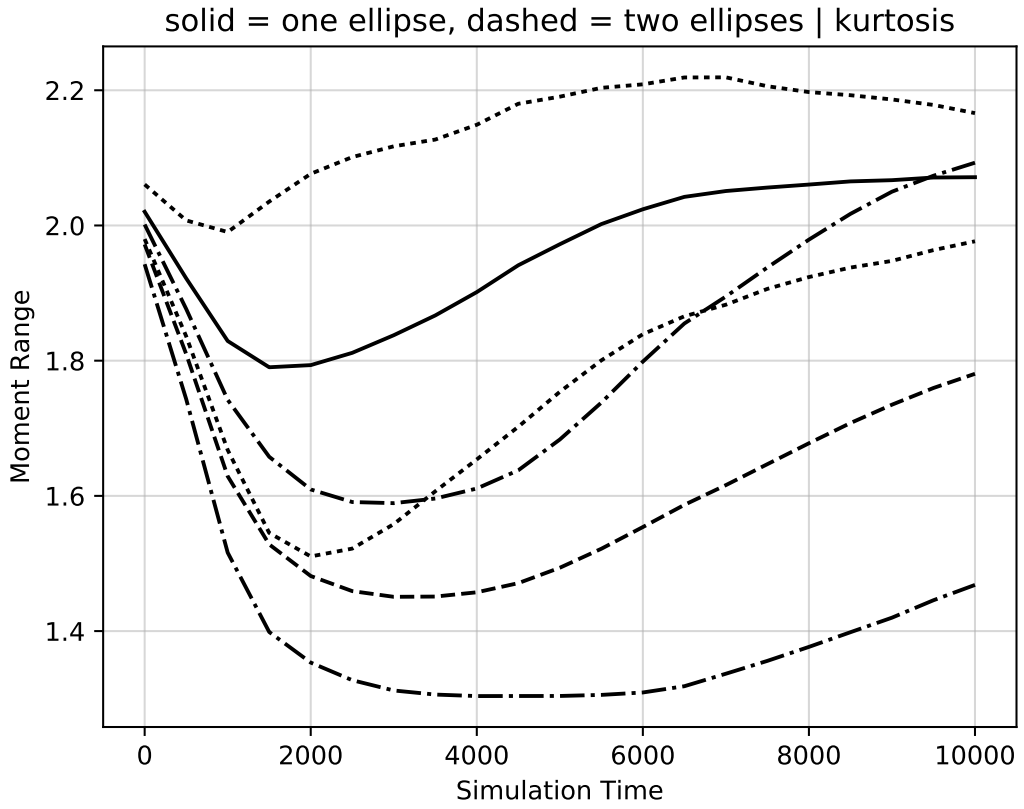

Figure 2.9: Kurtosis of the y coordinate of the unbiased ellipse shape aggregations over time.

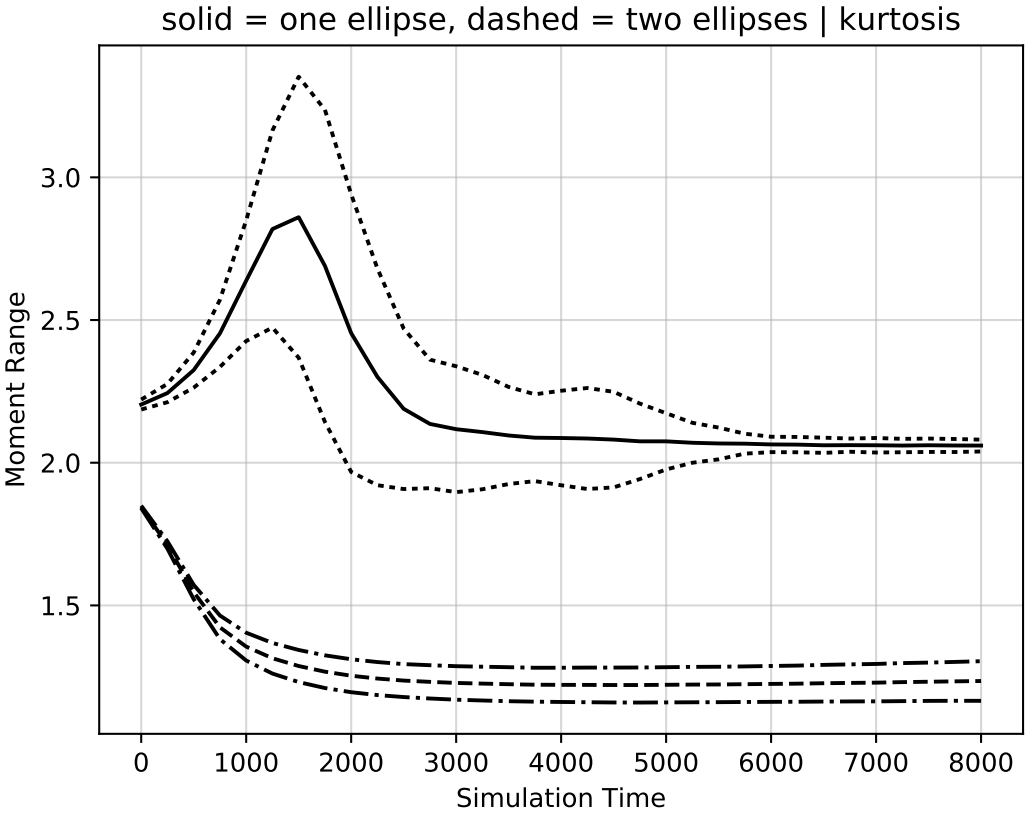

Figure 2.10: Kurtosis of the y coordinate of the biased ellipse shape aggregations over time.

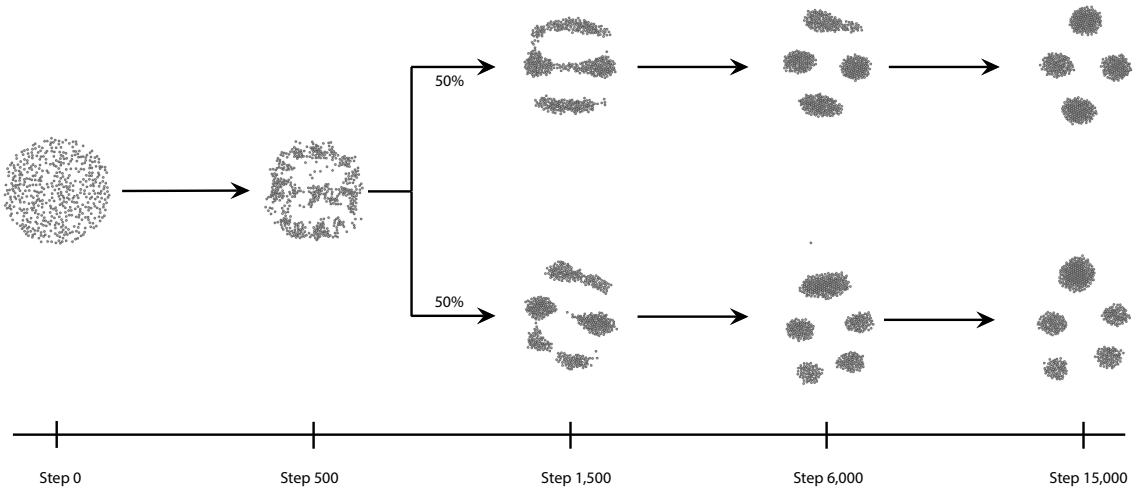

Figure 2.11: Shape aggregation of the discs MPs starting with random initial conditions.

We were unable to consistently generate a 4-disc result for most simulations by thresholding the variance. Thresholding the x kurtosis to be greater than 1.90 and less than 2.09 did lead to an increased number of 4-disc results (75%), which we deemed as less than consistent or robust. Figure 2.13 presents the evolution of x variance over the course of the 3-disc aggregation given the biased initial conditions. Note that no 3-disc results were produced when keeping the x variance above 15,500 in the biased initial conditions.

The parallel line dataset, when run with unbiased initial conditions, contains 526 instances of two vertical parallel line segments (84.1%) and 100 instances of one vertical line (15.9%), as seen in Figure 2.14. The simulation reaches a stable state by 50,000 steps. We identified kurtosis in the x coordinate to be the significant macroscopic feature for the shape. See Figure 2.15. 100 simulations were performed with initial conditions that had their x coordinates' kurtosis thresholded to be below 1.90. Given these biased initial conditions all simulation (100%) produced the two line structure. 100 simulations were then performed with initial conditions that had their x coordinates' kurtosis thresholded to be above 2.29. These biased conditions produced results that consistently (100%) created the minority class structure of a single line by 10,000 steps. Figure 2.16 presents the evolution of x kurtosis over the course of the aggregation given the biased initial conditions.

2.6 Discussion

Our experiments show that our agents (MPs) can be reliably directed to form into large-scale, macroscopic structures using local-only behaviors, based on chemical diffusion fields and biased initial conditions;

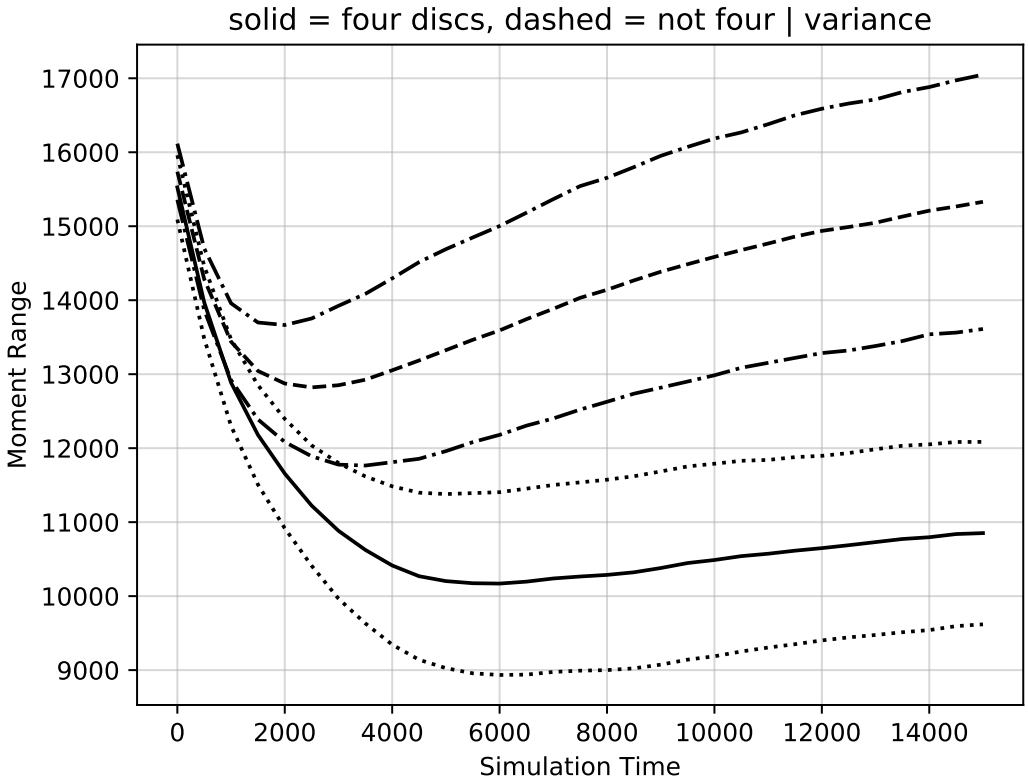

Figure 2.12: Variance of the x coordinate of the unbiased discs shape aggregations over time.

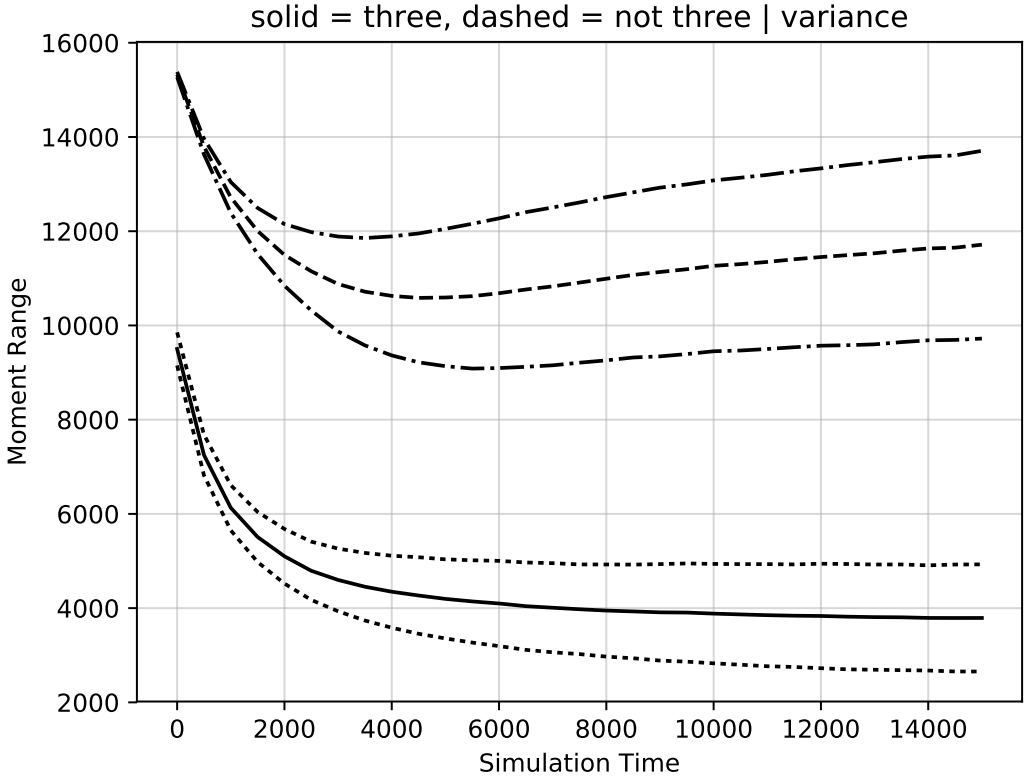

Figure 2.13: Variance of the x coordinate of the biased discs shape aggregations over time.

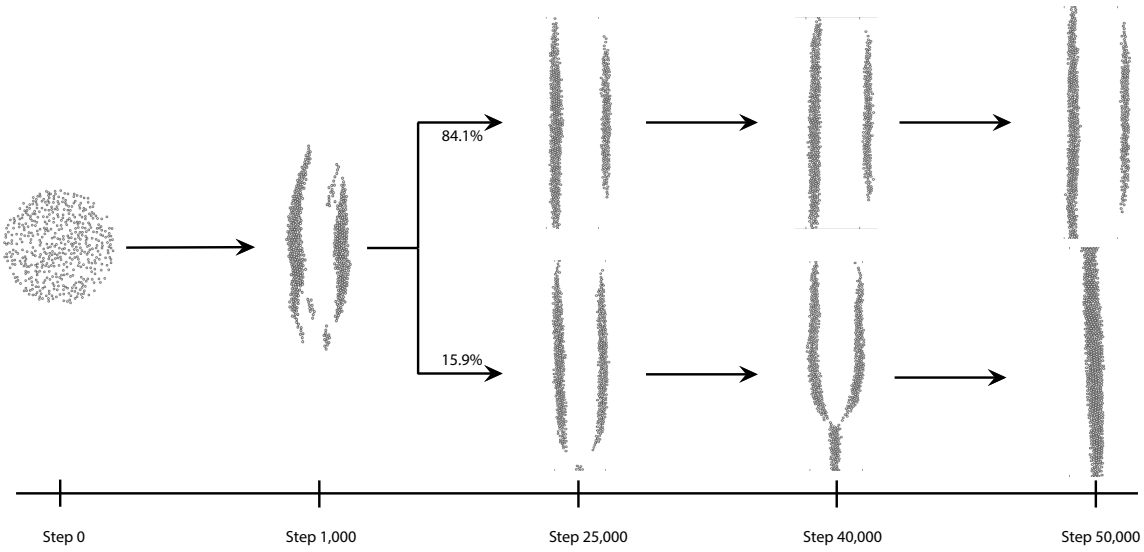

Figure 2.14: Shape aggregation of the line segment MPs starting from random initial conditions.

thus producing a stigmergic phenomenon [112]. This type of global outcome is of particular interest to those looking for a robust, adaptable, and independent self-organizing system. A review conducted by the European Space Agency has shown that for their harsh working environment (space, Mars, etc...) the robustness of a local-only system is a key consideration and would allow for relatively simple (and easy to transport) satellites/equipment to form into a larger and more complex system that would be impossible to transport as a monolithic structure [53]. Systems using global information, with centralized communication between primitives, or a command-and-control structure, can form more complex shapes more quickly than local-only approaches. However these systems can fail if this communication is interrupted or the command-and-control structure breaks down [86].

The reasons that certain biased initial conditions may be used to direct the outcome of an aggregation process are frequently visually obvious. For example, the biased initial conditions seen in Figure 2.4(a) are clearly skewed to the right side of the arena. So it is clear that the majority of the agents are already amassed around the center of the object to be formed. This can also be seen in Figure 2.4(c), where lowering the variance of the X component makes the initial distribution of the MPs cluster around the central axis which the three discs will form along. The visual evidence that biasing initial conditions effectively pre-starts the aggregation process towards a desired shape is not as evident in the examples that constrain kurtosis ((b) and (d)) from Figure 2.4. A higher Y kurtosis value in example (b) means that there should be a higher concentration of agents along the $Y = 500$ axis. In (d), a lower X kurtosis value means that there should be

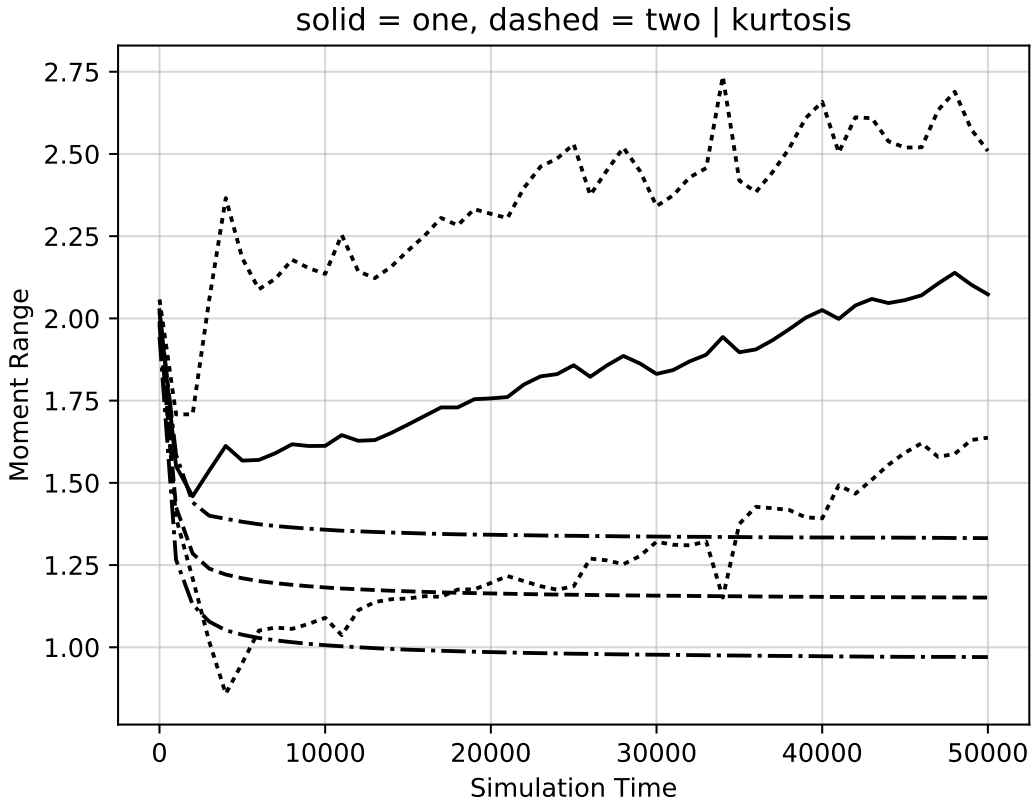

Figure 2.15: Kurtosis of the x coordinate of the unbiased line segment shape aggregations over time.

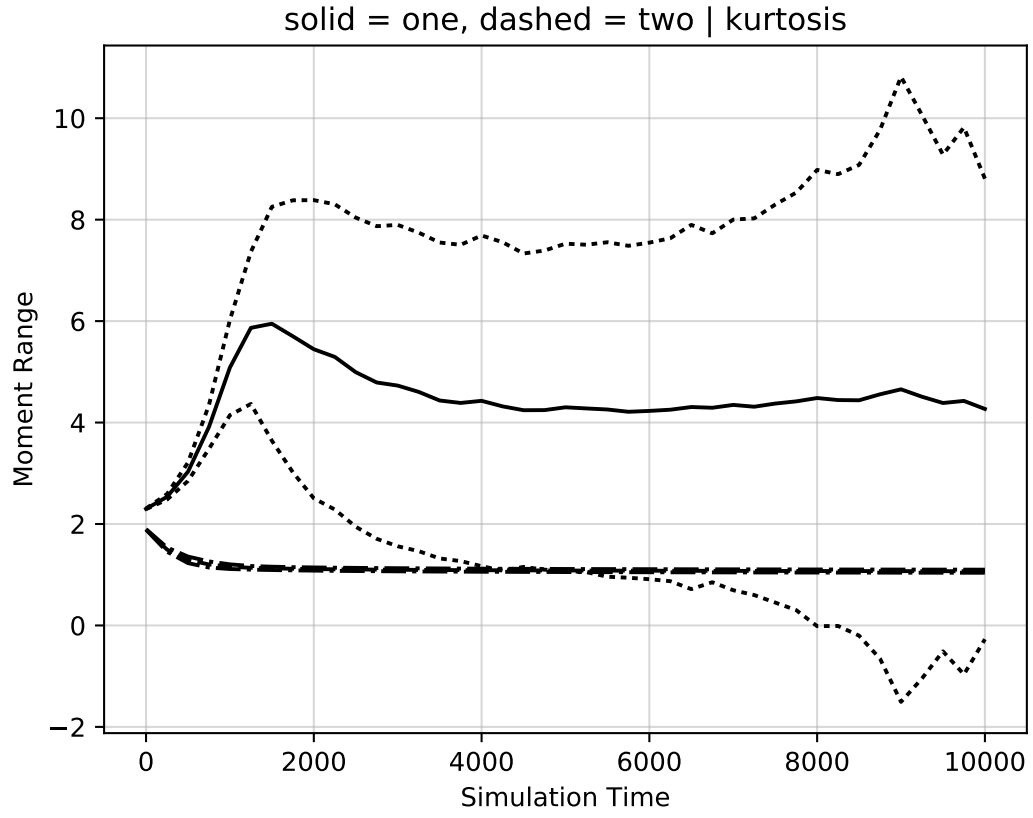

Figure 2.16: Kurtosis of the x coordinate of the biased line segment shape aggregations over time.

more agents distributed away from the center of the arena. But in both of these cases, while statistically this is true, it is not visually evident.

While a local-only system can be more robust than a system that relies on global information, there are some notable shortcomings associated with local-only systems. The main challenge in these types of systems is finding the local interactions that will lead to a desired emergent global behavior. While we use genetic programming to find these local interactions, it is not always possible using this generate-and-test approach to identify the correct local interaction. A more prescriptive approach to local interaction determination is clearly needed.

As previously described our system generates initial conditions that bias the primitives' locations based on a significant statistical moment that was manually determined after processing unbiased data. This takes time and requires a non-trivial amount of manual sorting and identification of images. Future work could benefit from a computer-vision based shape identification system, something similar to the system described in [131]. It should be noted that not all shapes are stable throughout the simulation. The vertical lines being a good example; up to approximately 5,000 steps the 1-line and 2-lines shapes look very similar and have similar statistical moments, however between 10,000 and 45,000 steps 15.9% of the simulations will converge into a single line. This shows that some shapes may appear to be stable at one point during a simulation, when actually they have not yet stabilized. Or in other words, our agents may produce more than one type of distinct shape during their evolution. Thus, our experiments show that we are able to consistently produce certain self-organized swarms at a specific point in time during their aggregation.

2.7 Future Work

For these types of self-organizing systems, it is clearly desired to have a local-only solution. Given that the methods described here require the computation of global system-level quantities (moments of the entire distribution) and the manipulation of the locations of the agents in order to meet some global constraint, we have not achieved this goal. Future work could explore if our genetic programming approach to local chemical field evolution, that leads to the formation of macroscopic shapes, can also be used to find chemical fields (i.e. local interactions) that direct a swarm that is uniformly randomly distributed into one that has specific statistical properties. This would lead to a two-step approach that is truly based on local-only interactions. In this case, the agents, which have been uniformly randomly placed in an environment, would follow chemical fields that move them into a biased initial condition. Then they would switch to a field that robustly directs

them to form in a specific macroscopic shape.

Additional future work with this system should involve robustness testing of the parameters of the initial conditions. Previous work in this area has defined a lower-bound for the number of primitives required to successfully create a single ellipse (400), but has not defined an upper-bound [10]. An upper and lower bound on the number of primitives (which of course is related to MP density) in the simulation is important as more complex shapes are created. Additionally, future work may involve further analysis of the aggregation processes that cannot be completely controlled. These investigations should reveal new features that differentiate swarms that can be controlled via the reported method and those that cannot. We imagine that manipulation of other features will further enhance our ability to direct our self-organizing system.

2.8 Conclusions

We have previously developed an agent-based self-organizing shape formation system. The agents perform identical behaviors based on sensing local information emitted into the environment by the agents. Genetic programming may be used to discover local interaction rules that lead the agents to self-organize into a number of user-specified shapes. However, since the agents are initially uniformly randomly placed in the environment and they stochastically follow prescribed rules, the aggregation simulations do not always produce the same final results. In order to develop methods that could be used to direct the agents to robustly form one specific configuration, we explored the relationships between an agent swarm's moments and its final configuration. After having shown that these moments could be used to predict the outcome of an MP aggregation in previous work, we demonstrate in this work that biasing the swarm's initial conditions based on these moments can be used to consistently direct the swarm to produce a desired macroscopic shape.

By analyzing the statistical moments of the agents' positions over the entire shape aggregation process, we have identified significant, distinguishing moment features, and utilize them as constraints on simulation initial conditions for a number of bifurcating shapes. Biased initial conditions may be generated that meet these moment constraints, which then affect the resulting shape outcomes. In almost all of our examples we can completely control the result of the self-organization process. In some other cases we can significantly increase the likelihood of producing a desired configuration. In a more general sense, our work also indicates that complex, non-linear dynamical self-organizing systems may be controlled by manipulating their initial conditions.

In the following chapters the agents, while remaining simple, become more complex than those presented

in this chapter. Subsequent agents maintain internal state and allow for and seek social interactions with other agents. Social interactions take the place of chemical concentrations, allowing for information dissemination without using global control or knowledge transfer. Each agent, when interacting, has a local–only interaction with another agent; each agent sharing the knowledge they have and their current beliefs about that knowledge. MPs in this chapter produced emergent behavior (shape formation) through self–organization while following chemical gradients emitted by each MP. Agents in subsequent chapters use an aggregation mechanism to arrive at an emergent prediction representing the beliefs of all agents.

Quarter-moon			
Shapes	Unbiased Percentage	Thresholded Moment	Biased Percentage
left-pointing	50%	$Skewness_x \geq 0.315$	100%
right-pointing	50%	$Skewness_x \leq -0.315$	100%
Ellipse			
Shapes	Unbiased Percentage	Thresholded Moment	Biased Percentage
single ellipse	50%	$Kurtosis_y \geq 2.18$	100%
non-single ellipse	50%		0%
two ellipses	–	$Kurtosis_y \leq 1.85$	100%
Multiple Discs			
Shapes	Unbiased Percentage	Thresholded Moment	Biased Percentage
three discs	0%	$Variance_x \leq 10,270$	100%
no three discs	100%	$Variance_x \geq 15,500$	100%
Shapes	Unbiased Percentage	Thresholded Moment	Biased Percentage
three discs	0%		4%
four discs	50%	$1.99 \leq Kurtosis_x \leq 2.09$	75%
five or more discs	50%		21%
Parallel Line Segments			
Shapes	Unbiased Percentage	Thresholded Moment	Biased Percentage
two lines	84.1%	$Kurtosis_x \leq 1.90$	100%
one line	15.9%	$Kurtosis_x \geq 2.29$	100%

Table 2.2: Table summarizing results generated with unbiased and biased initial conditions.

Chapter 3. Wisdom of Crowd Bots: Hollywood Movie Classification with Social Agents

In this chapter I present the multi–agent, social environment where WoC–Bots are trained, interact socially, and aggregate an overall prediction. This method tracks agent opinion throughout each of their interactions, as well as the initial features agents are trained on, allowing for additional analysis to determine when and how opinions may be changing throughout the crowd during the interaction process. The agents, WoC–Bots, are designed as simple computer–based agents, implementing theories from Wisdom of Crowds and use a competent aggregation mechanism to elicit an overall conclusion from a crowd of individual agents. This method is tested using a Hollywood movie dataset, predicting the success of a movie based on the movie's budget and the reported revenue of the movie. Each prediction is completed after agents interact socially, using a trust–based aggregation mechanism to determine an aggregate prediction.

3.1 Background

We, humans, want to predict the future; disease outbreak and risk factors, business success, economics, and many more applications can benefit from better forecasting. Researchers have developed many tools to help us make predictions, with artificial neural networks (ANNs) being a popular choice [30]. ANNs can be used for classification, allowing us to take, for example, a series of features about an upcoming movie and determine, with fairly high accuracy, if the movie will be successful [90]. ANNs, however, are inherently inflexible [134]. Recent efforts are improving their flexibility by adding to their basic design, as seen in transfer learning [79], however this increases complexity and compute / data requirements while further obfuscating the internal workings of an ANN, making it more difficult to answer the "why" about some outputted classification [129].

Prediction markets (PMs) are an alternative to traditional classification algorithms. They are designed to determine the probability of a future event taking place [18]. Well–designed PMs encourage agents, human or computer–based, to contribute information to the market through trading shares and incentivizing correct, truthful information sharing, then aggregate the information from individual agents into a collective knowledge [85]. PMs work because the aggregate knowledge of the group will generally be more precise and complete than the knowledge that any individual within the group holds [17]. However, participants

are expected to be well–informed, which works well with human participants, but is infeasible for computer–based agents [128]. Computer agent–based PMs are difficult and programmer–intensive to create. Othman said on computer–based agents, "Agent–based modeling of the real world is necessarily dubious. Attempting to model the rich tapestry of human behavior within economic structures – both the outstandingly bad and the terrifically complex – is a futile task."[80] Even if it were possible to model the complexity of human knowledge and decision making within some narrow topic, it is extremely difficult to generalize across topics. Alternatively, prediction markets based on simple agents have had some limited success. However, work done by Othman and Sandholm [81] has shown that simply changing the order in which the simple agents participate in the market can drastically impact the outcome of the market, indicating that "markets may fail to do any meaningful belief aggregation."

An alternative to PMs is Wisdom of the Crowd (WoC). WoC takes the approach that the opinion of a large, diverse group will be more accurate than any individual opinion within the group given a sufficiently competent aggregation mechanism [34]. WoC does not expect or require expert knowledge, Scott Page said "The squared error of the collective prediction equals the average squared error minus the predictive diversity" [82]. This means the more diverse the crowd, the smaller the predictive error.

WoC–Bots are a computer–agent–based approach to making binary predictions using a WoC approach where simple agents (WoC–Bots), without expert knowledge, are trained with different, small subsets of features that describe the data. This initially gives a group of agents with a diverse and independent set of knowledge. The agents interact with one another socially, sharing knowledge, determining the trust they have in other agents and the confidence they have in their own opinion, forming a knowledge–diverse crowd. During the interaction period the agents can change their opinion given enough evidence found during interactions. The architecture described in this chapter is the basis for experimental results presented in this and subsequent chapters. This approach is designed to be modular and extensible, allowing for varied algorithms that govern agent initialization, movement, interaction, and voting [44].

3.2 Data & Experiment Description

WoC–Bots were tested by making binary predictions about the success of Hollywood movies using a trust–based aggregation mechanism to determine an overall prediction from all agents and were compared with the classification performance of a simple multi–layer perceptron classifier. Success for a Hollywood movie was defined as the reported revenue being greater than $2\times$ the reported budget for the movie. It is

Table 3.1: Features available for classification

Features	Description
budget	given to all agents, reported budget for movie
tmdb_popularity	dynamic variable from TMDb API attempting to represent interest in movie
revenue	used for sanity checks, reported revenue
runtime	unreliable metric for success without including genre information
tmdb_vote_average	average score from TMDb, can be combined with ML average
tmdb_vote_count	total votes for a movie from TMDb, can be combined with ML count
ml_vote_average	average score from ML, can be combined with TMDb average
ml_vote_count	total votes for a movie from ML, can be combined with TMDb count
ml_tmdb_genres	combined genre information from TMDb & ML; first 2 listed genres used
vote_average	combined tmdb_vote_average and ml_vote_average
vote_count	combined tmdb_vote_count and ml_vote_count

difficult to determine exactly what revenue is considered a success, and it differs on a movie–by–movie basis. However, advertising and promotion budgets are generally less than the production budget, which indicates studios should start to see positive cash flow if a movie makes $2\times$ the production budget [26]. Additionally, defining success in this manner split the data roughly equally between success (47.5%) and failure (52.5%). Data was gathered from The Movie Database (TMDb) [1] and MovieLens (ML) [2]. The MovieLens dataset provides information for more than 27,000 movies, while the TMDb dataset includes 5,000 movies. Only movies found in both datasets, with complete information for all features used for classification, were considered. We found 4,722 movies which exist in both datasets, however 1,023 movies appear to contain incorrect information, e.g. negative values for movie budget or movie revenue; these movies were removed from the testing and training subsets. Two datasets were considered to help reduce sparse areas for less popular or older movies. 3,699 movies were considered for testing and training, split 80% / 20% respectively. Table 3.1 describes the features considered for classification. The data was transformed in the following ways:

- Movies were matched between the two datasets based on ID, using "movieId" and "tmdbId" values provided in the ML dataset.
- The ML dataset used a 0-5 rating system while the TMDb dataset used a 0-10 rating system, the ML ratings were multiplied by 2.
- Only overlapping genres from each dataset were considered; e.g. if, for Toy Story, the ML dataset lists

[1] https://www.kaggle.com/tmdb/tmdb-movie-metadata/
[2] https://www.kaggle.com/grouplens/movielens-20m-dataset/

it as "action, animation, family" and the TMDb dataset lists Toy Story as "family, adventure, animation", the movie was considered to fall into only the "animation" and "family" genres.

3.3 Method Design

3.3.1 Agent Design

Agents are designed to be modular, presenting an interface that allows for different algorithms which govern all aspects of their behavior. Agents are responsible for coordinating and managing the following functions:

- Classifier: Simple MLP classifier
- Classifier configuration: Shape, depth, activation and optimization algorithms
- Initialization algorithm: How the agents are initialized within the interaction space
- Movement algorithm: How the agents move within the interaction space
- Interaction algorithm: How (and if) an agent interacts with other agents
- Scoring algorithm: How an agent reaches the conclusion it does; a combination of the internal classifier and information learned while interacting with other agents

3.3.2 Classifier

This classification method uses simple agents (WoC–Bots) without expert knowledge. Each agent contains a very small, very simple MLP classifier. All agents are configured with similar classifiers, each containing a small number of input nodes (2-10), 1-4 hidden layers – depending on input size – rounded to the nearest integer value of $input\_size * 0.3$, and an output layer, with two output nodes for binary classifications. The goal is for each MLP to be as shallow as possible while still extracting features from the input, and $input\_size * 0.3$ has proven to be sufficient to extract features without creating overly deep classifiers. This work uses DeepLearning4J (DL4J) as a neural network and data parsing library [3][110]. All MLP networks use the DL4J implementation of the Adam updater [57] (learning rate), softmax activation function [43], and traditional stochastic gradient descent for optimization [4].

[3] https://deeplearning4j.org/

In each of the experiments conducted, each agent's classifier received 1-4 highly correlated features in common with all agents. Feature correlation was determined using principal component analysis (PCA) [1], run externally to the approach being described here. Other available features were distributed across multiple agents randomly with duplicate features and multiple agents with the same feature set neither limited nor encouraged. Following theories from WoC, this leads to a group of agents with diverse knowledge. Agents that show poor MLP classification performance during the initial training step are not included in the remainder of the classification steps. Poor performance was defined as less than 50% accuracy on the evaluation set.

3.4 Interaction Period

Following the MLP training period for each agent, the agents participate in an interaction period in a simulated 2D arena. The purpose of the interaction period is to encourage information dispersal amongst agents using local–only interactions, e.g. two agents interacting with each other.

3.4.1 Agent Initialization & Movement

All agents participate and interact within a centralized 'interaction arena' (`Arena`). All agents must be initialized to an empty space within the bounds of the interaction arena. All movements must be within the bounds of the arena and at most two agents can share the same space within the arena. Interaction between three or more agents at the same location is not supported, with no current plans to implement support. The `Arena` can take any 2D shape comprised of rectangular faces, allowing for arrangements from a simple 4×4 square to something more complex, with multiple rooms, floors, and restricted movement between each, e.g. a simulated building. In this work, the arena is a square with the number of discrete spaces available to the agents equal to 2 times the number of participating agents.

Each agent maintains a history of their movements within the arena, their interactions, and how each interaction affected their internal belief. Historical information for each agent is stored in memory during each iteration before being dumped to individual SQLite tables for long–term storage.

Agents are initialized with an `InitializationAlgorithm` and a `MovementAlgorithm` which implement simple interfaces, `init()` and `move()`, respectively. `init()` has a single goal, initialize the agent in the arena in an empty space. Initialization can be random, or account for complexities like location of other agents, placing agents with similar features close together, localizing similar information, or spreading them

out to facilitate transmission of information between dissimilar agents. Localizing similar information may allow a group of similar agents to come to an optimal conclusion, whereas spreading similar agents out may allow the best performing agents to maximize their influence on other agents [40]. In the experiments conducted throughout this dissertation, the agents are initialized randomly inside the arena and agents are not allowed to share spaces with other agents at initialization. The method handles missing data for any given sample by simply disallowing participation of agents that have missing data for the current sample. This method allows us to consider all available information for each sample without requiring data completeness for all samples.

`move()` also has a simple goal, move the agent within the arena. `move()` can be as simple or complex as necessary; randomly selecting a space within the arena and 'teleporting' the agent to the new space, or it can require the agent to move towards some target location or another agent. Agents interact when two agents move onto the same space. The `MovementAlgorithm` used for this work moves agents randomly in a "Manhattan–like" fashion, allowing each agent to move one step north, south, east, or west, within the bounds of the arena, at each iteration. The direction is randomly selected, and agents cannot interact with each other two times in a row, or more than twice within five separate movements. Agents who cannot move because of these restrictions are 'teleported' to a randomly selected empty space within the arena. Additionally, agents are occasionally 'teleported' to a random space within the arena to further facilitate information dispersal. The total number of iterations ($interaction_{iters}$) the interaction period is allowed to run for is determined by the number of participating agents (num_agents)

$$interaction_{iters} = num\_agents * 0.10 \tag{3.1}$$

and was determined to balance information sharing with run time through experimentation. Experimentation with the Hollywood dataset used in this chapter determined increasing the number of iterations to $num\_agents * 0.25$ did not increase the predictive performance but did increase runtime. The interaction period is operated in discrete iterations; each agent is moved during each iteration and all interactions between agents must complete before the next iteration can begin. Centralized control for movement location and validation is being used to ease implementation, however it is not a requirement. Agents are capable of validating location and movement on their own, or if required, having some number n other agents confirm all positioning as valid in a decentralized manner.

Table 3.2: Internal scoring variables

Variable	Description
`current_prediction`	Binary, class 0 or class 1.
`trust_score`	Updated during interaction based on agreement & performance.
`features`	A list of features used by the agent's classifier.
`prior_performance`	Long-term history of agent performance, varied between 0.7 and 1.3 where 1.0 is average performance.
`certainty`	Initialized to MLP classifier performance, updated during interaction period. Represents how strongly the agent believes in the current prediction.
`eval_accuracy`	Initial classification accuracy.
`eval_precision`	Initial classification precision.
`eval_recall`	Initial classification recall.
`confidence`	Biased value based on accuracy, precision, and recall.

3.4.2 Interaction & Scoring

The `InteractionAlgorithm` and `ScoringAlgorithm` are also designed to be modular, with the `InteractionAlgorithm` responsible for deciding with whom an agent should interact, truthfulness, and trust updating. The `InteractionAlgorithm` is required to implement three functions, `shouldInteract()` which determines if the agent will interact with another agent, `truth()` which determines if the agent should be truthful with another agent, and `updateTrust()` which attempts to update the other agent's trust score. `updateTrust()` is allowed to be a NO-OP function when it is not desirable to update other agents' trust scores. This work currently assumes all interactions are acceptable, does not limit repeat interactions (except as previously described), and requires all interactions to be truthful.

The `ScoringAlgorithm` determines how interactions update an agent's internal belief state. Agents are initialized with specific internal values, referenced in Table 3.2, that are (in part) updated during each interaction. Table 3.2 presents the attributes which describe the internal state of an agent during the interaction period. Many of these values are available to other agents during interactions within the social interaction arena. These values allow each agent to determine how certain another agent is, what that agent's initial classification values were, and how much influence it will allow the agent to have over its current belief. Additionally, the `confidence` attribute can be biased to favor accuracy, precision, or recall metrics depending on which metric is the most important for the given classification problem. For example, a classification task such as the breast cancer dataset presented in Chapter 4 may benefit from biasing towards recall to better

avoid false negative predictions

$$confidence = accuracy * 0.25 + precision * 0.25 + recall * 0.50. \tag{3.2}$$

Similar to the previous algorithm interfaces, the `ScoringAlgorithm` used by each agent allows the scoring to be implemented in a way most appropriate to the given problem. The algorithm is required to implement a single function, `updatePrediction` which updates the current prediction based on information from the most recent interaction. The `ScoringAlgorithm` used for the experiments in this dissertation works as follows: during the interaction period, two agents, agent a and agent b, interact when sharing a space within the arena. The following equations govern the updates to the internal state of agents during the interaction and determine if and when an agent's prediction will change, and what confidence the agent has in its current prediction. The equations assume agent a is receiving information from agent b, however during an interaction the inverse will also occur.

When two agents meet, agent a first determines how willing it is to accept information from agent b, a function of a's current `certainty` where $a_{certainty}$ represents a's current `certainty` and $a_{acceptance}$ represents a's willingness to accept information from agent b.

$$a_{acceptance} = 1.0 - a_{certainty} \tag{3.3}$$

Agent a determines how much influence to allow agent b, $b_{influence}$, a function of agent b's `confidence`, `trust_score`, and `certainty`, and agent a's acceptance, $a_{acceptance}$.

$$b_{influence} = b_{confidence} * a_{acceptance} * b_{trustCertainty}, \tag{3.4}$$

where $b_{trustCertainty}$ represents `trust_score` $*$ `certainty`,

$$b_{trustCertainty} = b_{trustScore} * b_{certainty}. \tag{3.5}$$

Agent b's influence is further modified based on agent b's `prior_performance`, represented by $b_{priorPerf}$, such that the corrected influence can be represented by $b_{correctedInfluence}$,

$$b_{correctedInfluence} = b_{priorPerf} * b_{influence}. \tag{3.6}$$

If agent a and agent b have different predictions $b_{correctInfluence}$ will be multiplied by -1 to act against agent a's current predictive belief. Agent a's updated certainty, $a_{certainty}$, can be calculated by Eq. 3.7, where a's certainty increases if both agents a and b have the same prediction and decreases if the predictions are different,

$$a_{certainty} = a_{certainty} + b_{correctedInfluence}. \tag{3.7}$$

If agent a's certainty value falls below 0.50, agent a will change its prediction. If the prediction is changed, agent a's certainty is updated by Eq. 3.8.

$$a_{certainty} = 1.0 - a_{certainty}. \tag{3.8}$$

Following the interaction period, an overall prediction is made by an opinion aggregation algorithm. The following chapters discuss two approaches that have been implemented and report on the results of each method.

3.5 Opinion Aggregation

Effective opinion aggregation is an open question with many different possible approaches [31]. The first method implemented uses a voting system, where each agent receives a maximum of 100 possible votes for their predicted binary outcome. Three aggregation methods for vote aggregation were considered; the first and simplest method gives equal weight to each agent regardless of performance, the Unweighted Mean Model (UWM) [49]. The second method gives each agent votes based on prior accuracy, where an 80% accuracy rate would result in 80 votes, similar to the Weighted Voter Model (WVM) presented in [117]. Agents are initially allowed 50 votes each until an accuracy for prior performance can be determined. The third method is a custom design that is similar to the WVM, however it also takes into account the trust score that other agents are allowed to modify during the interaction period, giving more granular control over how much influence an agent has on the aggregate opinion. Total votes for agent a is represented by $a_{totalVotes}$, where $a_{priorAccuracy}$ represents a's prior accuracy and a_{trust} represents a's trust score,

$$a_{totalVotes} = \left(\left(a_{priorAccuracy} + a_{trust}\right)/2\right) * 100. \tag{3.9}$$

During an interaction the agent, a, is allowed to modify another agent's, b, trust score (b_{trust}) based on

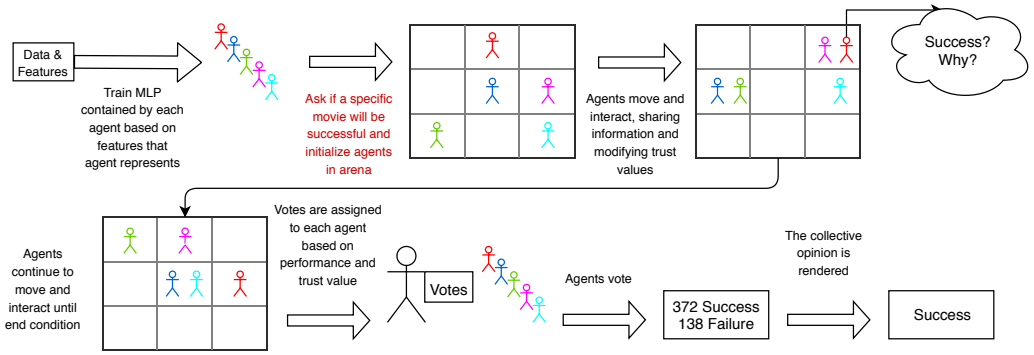

Figure 3.1: Agent training, initialization, movement, interaction, and voting.

how agent b has performed in the past, if agent a and b are in agreement ($doAgree$), and if prior information that a has received from b was correct. Agent a will check its interaction history, look for any interactions with b to determine what percent of interactions gave advice that was correct ($b_{percCorrect}$). If there are no prior interactions the trust score will not be modified. The trust score can be modified a maximum of 5% during each interaction

$$b_{trust} = b_{trust} + 0.05\left(b_{percCorrect} * b_{priorPerf}\right) * doAgree. \tag{3.10}$$

A high-level overview of the training, interaction, and voting process can be seen in Figure 3.1. The internal MLP for each agent is trained using available training data, the agents are presented with a binary question, they are initialized in an arena where they move and interact for some number of iterations. Agents are then assigned some number of votes based on the algorithm; described in section 3.5 which they use to vote at the end of the interaction period.

3.6 Results

This chapter compares the results from the social, agent–based approach to the results produced by multiple configurations of a traditional, monolithic MLP developed in DL4J. Table 3.3 shows the configuration and accuracy under different training conditions for each of the 10 agents, showing how the MLP for each agent performs by itself after initial training. Once trained, the agents can participate in any decision making configuration without retraining their classifiers.

10 MLP networks were tested, with a variety of feature sets, and with one containing the final `revenue` to confirm both the agents and the MLP networks were learning. Figure 3.2 shows the accuracy for five

Table 3.3: Agent Classifier Accuracy for 5 and 50 epochs

Features	5 Epochs	50 Epochs
budget, revenue	98%	100%
budget, vote_average, vote_count	77.2%	77.6%
budget, tmdb_popularity, vote_average, vote_count	75.4%	75.7%
budget, vote_count	75.7%	75.5%
budget, tmdb_popularity, tmdb_vote_average, tmdb_vote_count	72.8%	74.9%
budget, tmdb_vote_count, ml_vote_count	73%	73.4%
budget, ml_vote_average, ml_vote_count	62.2%	64.1%
budget, ml_vote_count	60.3%	61.9%
budget, tmdb_vote_average	60.9%	61.4%
budget, runtime	53.9%	56.4%
Average (budget, revenue agent removed)	67.93%	68.99%

classifier configurations when trained for 5 and 50 epochs. The figure also shows the change in classifier accuracy when various features, shown on the Y–axis, have been removed as inputs to the network.

Comparing training times with the social agents, training 10 MLP classifiers in parallel takes 1 minute and 1 second (3 minutes and 32 seconds if trained sequentially) vs. 22 seconds to train 10 agents for 50 epochs. It should be noted that inference is slower using WoC–Bots; it takes an average of 260 milliseconds to test 740 examples using an MLP classifier incorporating all the features listed in Table 3.1, while it takes the WoC–Bots, encompassing the same feature set, an average of 13.4 seconds to test the same 740 examples. But, once trained, the agents can be reconfigured to compute new prediction results for different feature sets, without requiring retraining, unlike a monolithic MLP.

Accuracy results from five configurations of the social, agent–based prediction method can be found in Figure 3.3. Results are shown for three aggregation mechanisms after 50 epochs of training: (1) unweighted equal voting, (2) votes assigned based on initial classifier performance, and (3) votes assigned based on classifier performance and agent trust (described in section 3.5), with method (3) consistently out–performing methods (1) and (2). Similar to Figure 3.2, Figure 3.3's labels show which features were included in the interaction. Feature distribution across agents was optimized for accuracy; that is, the most highly correlated features were distributes to each agent. Five agents participated in each interaction. with the budget, vote_average and budget, vote_count simulations being comprised of five copies of the same agent.

WoC–Bots out–performed the MLP classifier in all cases except where final revenue was included as a feature, indicating the aggregation method does not give enough weight to an agent with exceptionally good

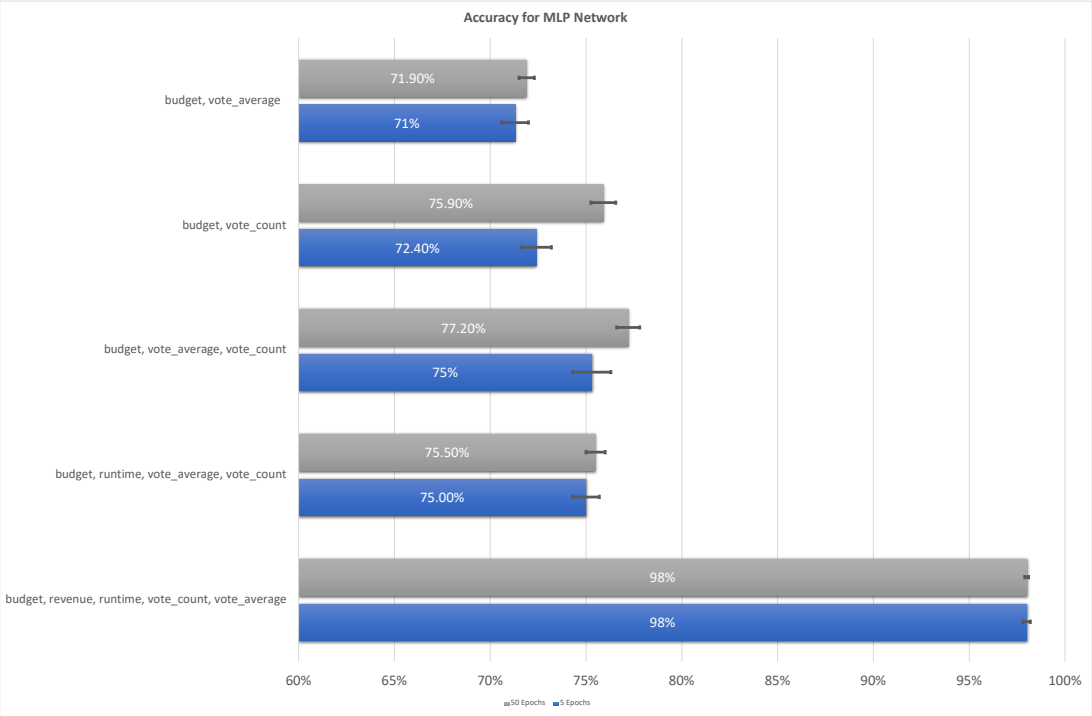

Figure 3.2: Accuracy for single hidden layer MLP classifier when trained for 5 and 50 epochs, averaged over 10 runs of the algorithms. The x-axis represents prediction accuracy, and the y-axis shows which features were included in the classification. Error bars represent standard deviation.

performance. We tested removing a highly correlated [4] feature, `vote_count`, which caused a performance decline in both the MLP and social agents, with the MLP network accuracy declining 4% compared to a decline of 1.9% in WoC-Bots, indicating the agents are more resistant to feature drop–out. Removing a noisy feature was also tested, `runtime`, which showed a 1.7% performance increase in the MLP network and only a 0.3% increase for WoC–Bots, indicating poorly performing agents have little impact on other agents during the interaction period and receive few votes during opinion aggregation. Statistical analysis confirms that `runtime` is not highly correlated in both the TMDb dataset and an ensemble dataset combining the Movielens and TMDb data, as used in these article [6, 70].

Figure 3.4 shows the performance of MLP networks and WoC–Bots as features are systematically added. The results presented in this figure are produced via the classifier performance & trust aggregation mechanism. The agents are configured to allow for maximum agent participation without duplicating agents in any simulation testing more than 2 features. Four copies of an agent, representing `budget` and `vote_count`, participated in the first simulation. Four agents participated in the `budget`, `vote_count`, `popularity`

[4] `http://ibomalkoc.com/movies-dataset/`

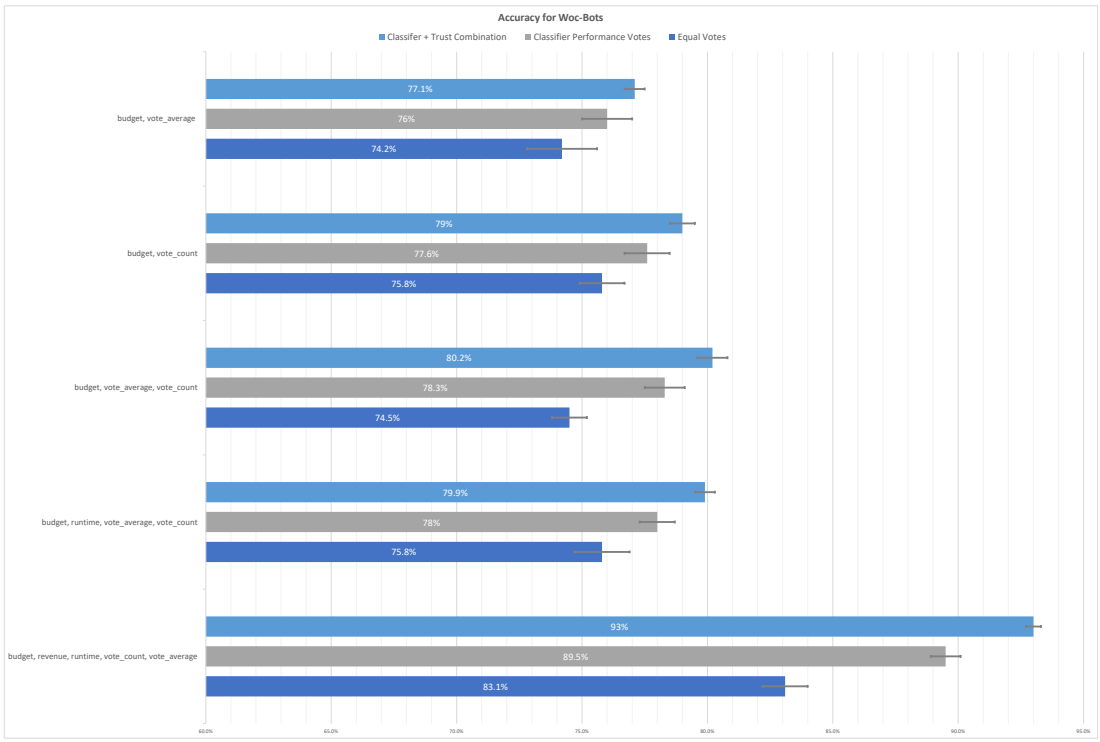

Figure 3.3: Accuracy for agent interaction using three different aggregation mechanisms, averaged over 10 runs of the algorithms. The x-axis represents prediction accuracy, and the y-axis shows which features were included in the classification. Error bars represent standard deviation.

simulation, eleven agents participated in each of the following 4–feature simulation. 26 agents participated in the `budget`, `vote_count`, `vote_average`, `runtime`, `popularity` simulation with one agent receiving 5 features, five agents receiving 4 features, ten agents receiving 3 features, and ten agents receiving 2 features.

In five out of six simulations, WoC–Bots out–performed the MLP network, and significantly out–performed the MLP network when the most important feature, `budget`, was removed. WoC-Bots performed best when all features were available, and when the maximum number of unique agents participated in the simulation. The MLP network performed best when the three most highly correlated features were the only features being considered. This performance difference indicates WoC–Bots are able to gather additional information from features that are less correlated with `revenue` without a net negative impact to their accuracy from the additional feature noise.

Figure 3.5 shows the accuracy for training epochs 1–50 for an MLP network and WoC–Bots. The network was configured with 5 features, `budget`, `vote_count`, `vote_average`, `runtime`, and `popularity`. Five agents participated in the simulation with four agents receiving 2 features and one agent receiving 5 features. Each 2–featured agent received `budget` as a feature and one other feature from the list of features.

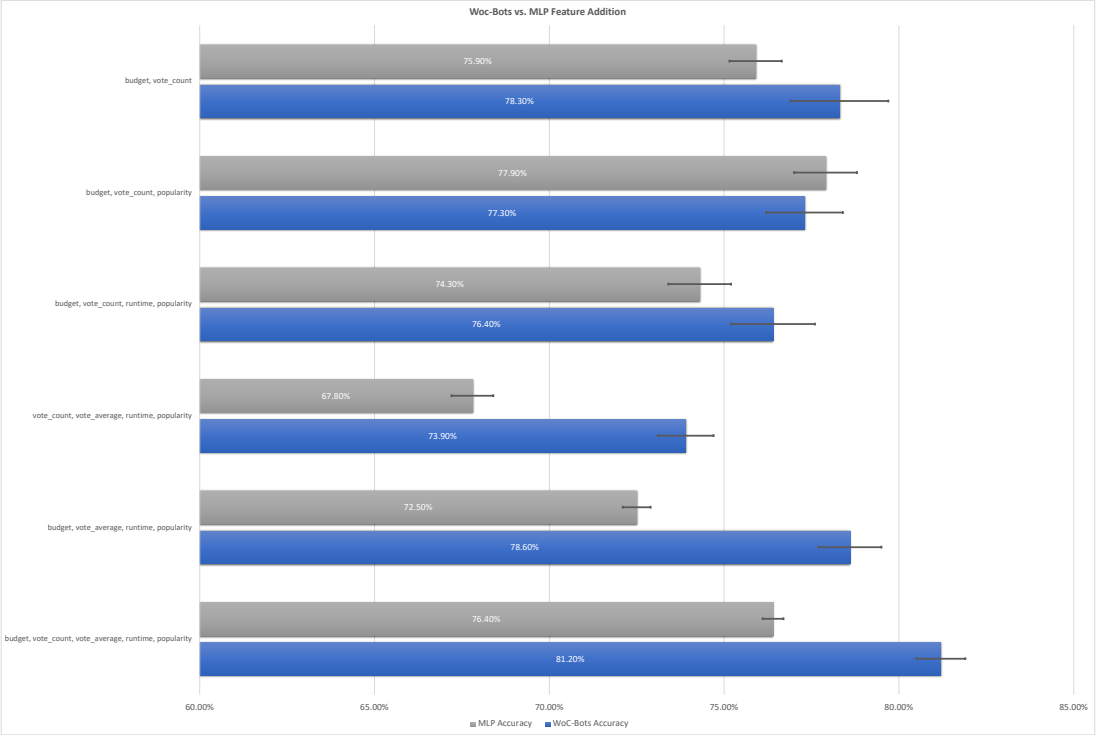

Figure 3.4: Accuracy of MLP vs WoC-Bots w/ Max agent configuration while adding features, averaged over 10 runs of the algorithms. Error bars represent standard deviation.

No agent was duplicated. We choose this agent configuration to best match the features used by the MLP network in order to make as fair and direct comparison with the MLP network as possible, despite WoC–Bots performing better when more agents participate in the simulation, as seen in Figure 3.4. WoC–Bots are out–performed in this configuration, slightly, by the MLP network when trained for 40+ epochs, however they are able to more quickly integrate information compared to the MLP network, reaching an optimum (76.3%) at 20 epochs vs. the MLP optimum (76.8%) at 40 epochs.

3.7 Conclusion

The WoC–Bots approach can include new features by creating a new agent to represent those features, allowing the agent to be included in the next "interaction" without the need to retrain a network or employ complex, dynamically expandable networks found in [129] when incorporating new data. Additionally, our design allows us to quickly test the impact of removing features or testing various combinations of features without the time–consuming retraining step required when changing features in an MLP network. This allows, for example, testing the impact of removing a feature like `runtime` quickly.

WoC performance depends on two attributes, a diverse and independent crowd and an aggregation mech-

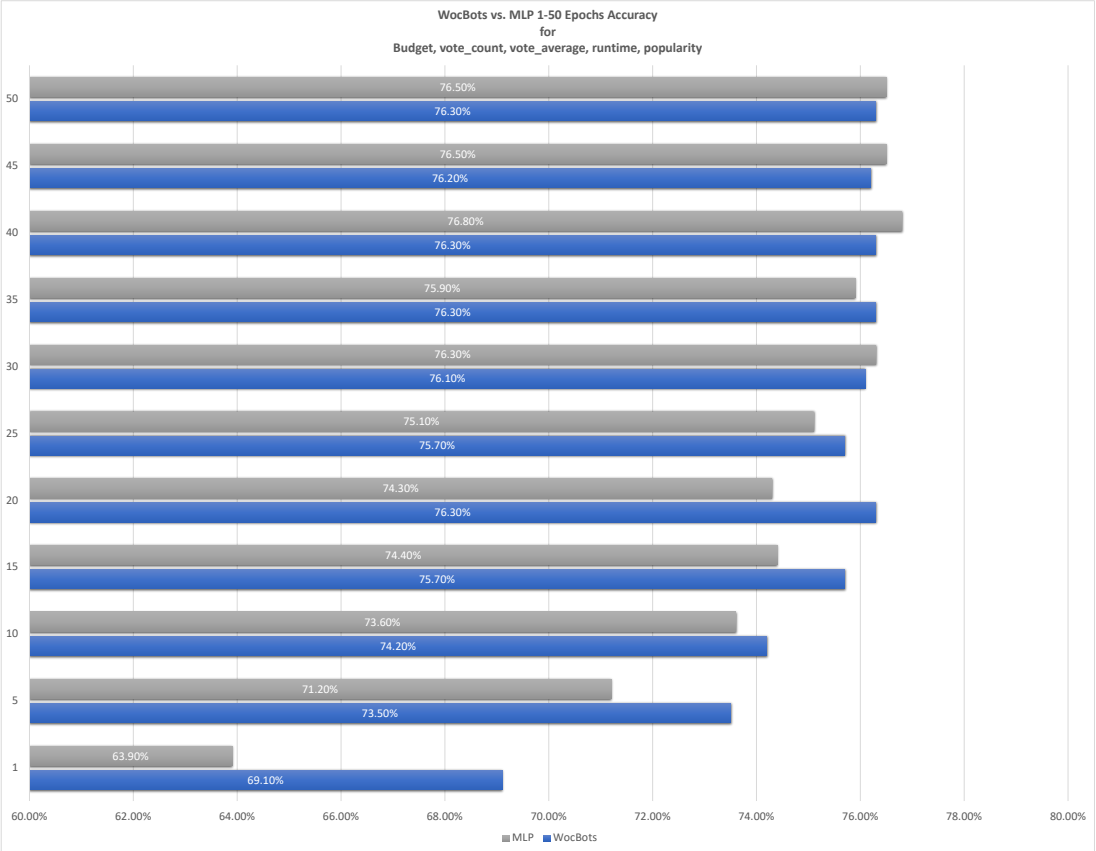

Figure 3.5: Accuracy of MLP and WoC-Bots for 5 features over 1 - 50 epochs

anism that assigns appropriate weights to individuals within the crowd to reach the correct collective decision [109]. We also know from Othman [80] that is it not reasonable to develop a computer agent that accurately represents the intricate and diverse knowledge that a human has. Therefore, we need to find the right balance between independently thinking agents within a crowd and information sharing to better represent the diverse knowledge of human agents.

WoC–Bots demonstrate a robust, flexible alternative to traditional ANN methods for making predictions about specific future events using theories from prediction markets and wisdom of crowds. The results presented in this chapter show we can attain similar predictive performance to that of a multi–layer perceptron classifier when classifying Hollywood movies, while requiring less training time and offering more flexibility and prediction options. Further, this method is robust, demonstrating only a 1.9% loss in accuracy when removing the `vote_count` feature vs. a 4% loss in accuracy when the same feature was removed from the MLP network.

Future chapters investigate an improvement to the aggregation mechanism. Theories from swarm intel-

ligence [137] are employed to better aggregate the information that each agent possesses in a manner that better extracts information from the **correct** agents while limiting the impact that **incorrect** agents have on the collective prediction. Unanimous A.I.[5] has a unique, swarm–based aggregation method that is capable of arriving at a collective prediction. Unanimous A.I. maintains a "human-in-the-loop" approach [94], where their 'swarm' is comprised of humans, answering binary and non–binary questions by working together to move a virtual puck to the collective answer [95]. We prefer a computer–agent–based approach that allows for new agents to be created as needed to answer questions as they come up. Chapter 4 focuses on implementing a swarm–based algorithm to produce an "emergent prediction" from a group of relatively simple, modular agents.

[5] https://unanimous.ai/

Chapter 4. Swarm-based Opinion Aggregation

This chapter presents results for a swarm–derived aggregation mechanism, specifically a honeybee–derived optimization algorithm. The basis for the initial training and social interaction steps is the same as Chapter 3, with each agent containing an MLP network and interacting socially in an interaction arena. However, instead of using a trust–based voting mechanism, a swarm–based mechanism is used for prediction aggregation. This mechanism allows for confidence scores to be derived for each classification as well as improves the overall variance in classification when compared with the previously used vote aggregation mechanisms.

4.1 Background

Prediction aggregation in multi–agent / social systems is an open problem with various different proposed solutions [31]. Swarming behavior in nature is something many people have witnessed, schools of fish avoiding a predator, birds flocking together, and insects working collectively to achieve some goal are common sights. In all cases, relatively simple, from an intelligence standpoint, individuals perform some action which ultimately produces complex behavior for the school, flock, or colony. This emergent complex behavior can be seen in more complex individuals as well, with human opinion successfully aggregated in [95] and [94] using 'swarm intelligence'. Swarm intelligence in computer science can be used to solve large and complex optimization problems [50]. Based on the emergent behavior seen in simple animals and the success seen when aggregating more complex human opinions, a swarm–based aggregation mechanism, specifically modeled on honeybee foraging behavior, was implemented to aggregate computer–agent predictions. Bee–colony optimizations was selected based on its efficient computation time, compared with other swarm–based mechanism [64].

Bee colonies are able to forage across a large area in multiple directions, often finding optimal sources for nectar and pollen – a task that looks like many optimization problems in computer science. A subset of bees (the scouts) move throughout the colony's foraging area, searching for high yield food sources [111]. Upon returning to the colony, the scouts perform a "waggle dance", advertising the location of the food source to other bees within the colony, which can then forage at the advertised location [20]. The length of time a bee is dancing is generally proportional to the quality of the food source, with additional bees recruited to higher quality food sources due to longer waggle dances [122]. This allows for high yielding

food sources to be advertised and foraged as long as they remain profitable, with a drop off in interest as better sources are found or as the source becomes depleted [16]. This chapter describes a computer–agent–based implementation of this method, simulating bee behavior to aggregate individual agent predictions to give a system–wide emergent prediction for binary classification problems.

In this chapter, the swarm aggregation method is applied by retrospectively examining tumor characteristics acquired in the routine care of patients at Drexel University College of Medicine to make a binary classification, node–positive or node–negative, that predicts lymph node metastasis status, a determination that could obviate surgical dissection of the lymph nodes. Multiple methods exist for predicting lymph node metastasis in breast cancer patients, the Memorial Sloan-Kettering Cancer Center (MSKCC) nomogram is an early attempt at solving this problem, that uses standard patient characteristics, such as patient age, primary tumor size, presence of lymphatic invasion, histologic grade, among other patient characteristics [105]. The MSKCC nomogram has, however, shown inconsistent results across datasets and may not be viable as a tool for decision making in some patient populations [119, 118]. Deep learning methods have been also been applied to this problem, specifically using convolutional neural nets as seen in [63], SVMs as seen in [132], and logistic regression in [14]. Similarly to the MSKCC nomogram and logistic regression study from [14], WoC–Bots use available patient characteristics to predict if lymph node metastasis has occurred in breast cancer positive patients. The swarm aggregation mechanism is used to place predictions in confidence intervals, which leads to an improvement in classification accuracy for a subset of high–confidence subjects.

4.2 Methods & Materials

4.2.1 Patient Features

Lymph node metastasis is predicted in 483 de–identified breast cancer patients. This research was approved by the Drexel University IRB office, with the data being distributed to the investigators using an honest broker. All features that were available for at least 50% of patients were considered, unlike previous work which considered only highly correlated features and required feature completeness [120, 58]. Table 4.1 shows the clinical features available for classification. Not all features were available for each patient.

In addition to the basic score values for ER, PR, P53, Ki67, and Her2, the agents were given access to the raw count values (when available) for 1+, 2+, 3+ (cell staining intensity), total cells measured, membrane intensity (where applicable), and positive and negative intensities (where applicable). These features were

Table 4.1: Characteristics of patient populations

Feature	$n = 483$	%
Age		
≤ 45	103	21.3
> 45	380	78.7
pT Max size (mm)		
≥ 200	5	1.0
100-199	15	3.1
50-99	51	10.6
25-49	125	25.9
0-24	271	56.1
unknown	16	3.3
Angio Lymphatic Invasion		
Absent	127	26.3
Present	200	41.4
Unknown	156	32.3
pT Stage		
Unknown	36	7.5
pT1	210	43.5
pT2	173	35.9
pT3/pT4	64	13.3
Histologic Grade		
Unknown	33	6.8
1	53	11.0
2	164	34.0
3	233	48.2
Tubule Formation		
Unknown	30	6.2
1 ($> 75\%$)	13	2.7
2 ($10 - 75\%$)	98	20.3
3 ($< 10\%$)	342	70.8
Nuclear Grade		
Unknown	29	6.0
1	20	4.1
2	151	31.3
3	283	58.6
Lobular Extension		
Unknown	202	41.8
Absent	147	30.4
Present	134	27.7
⋮	⋮	⋮

Feature	$n = 483$	%
⋮	⋮	⋮
⋮	⋮	⋮
⋮	⋮	⋮
Pagetoid Spread		
Unknown	213	44.1
Absent	177	36.6
Present	93	19.3
Perineureal Invasion		
Unknown	267	55.3
Absent	186	38.5
Present	30	6.2
Calcifications		
Unknown	115	23.8
Absent	126	26.1
Present	176	36.4
Present w/ DCIS	66	13.7
ER Status		
Unknown	51	10.6
Negative	155	32.1
Positive ($> 10\%$)	277	57.3
PR Status		
Unknown	54	11.2
Negative	201	41.6
Positive ($> 10\%$)	228	47.2
P53 Status		
Unknown	81	16.8
Negative	255	52.8
Positive ($> 5\%$)	147	30.4
Ki67 Status		
Unknown	56	11.6
Negative	114	23.6
Positive ($> 14\%$)	313	64.8
Her2 Score		
Unknown	83	17.2
0	119	24.6
1	169	35.0
2	54	11.2
3	58	12.0

collected using quantitative image analysis applied to whole–slide images of representatives slides acquired at the time of diagnosis. Slide scanning was done on either an Aperio XT or a Hamamatsu S210 slide scanner with quantification performed using Aperio Imagescope software and conformed to College of American Pathologists (CAP) recommendations.

4.2.2 Swarm Aggregation

The computer–based version of the honeybee foraging algorithm operates similarly to the bee foraging behavior previously described. The swarm aggregation step is designed for all agents to support a single binary opinion, either 1 or 0. All of the agents which participated in the social interaction step must also participate in the swarming step. In nature, between 20% and 30% of bees act as scouts [55], so 20% of agents are randomly selected to be "presenters", similar to the "scout" bee in nature. The remaining agents are assigned as "watchers", assigned to the presenting agents. Initial assignment of watchers is accomplished using 'fitness proportionate selection' [68]. Using fitness proportionate selection, each of the presenting agents has an assigned probability, a_{prob}, between 0 and 1, where the sum of all probabilities of all presenting agents is 1 after normalization. The probability for each agent is calculated by Eq. 4.1, an equally weighted combination of the presenting agent's `prior_performance`, $a_{priorPerf}$, `confidence`, $a_{confidence}$, and `trust_score`, a_{trust}

$$a_{prob} = (a_{priorPerf} + a_{confidence} + a_{trust})/3. \tag{4.1}$$

The fitness proportionate selection algorithm uses the computed probabilities for each of the presenting agents to assign watchers to be supporters of each of the presenting agents using a_{prob}, such that presenting agents with a higher a_{prob} will more frequently be assigned watchers than those with a lower a_{prob} value. Watchers can only be assigned to a single presenting agent at any given time and 'support' that presenting agent simply by being assigned to it. However, each watcher can request re–assignment up to two times if its prediction is different from that of the presenting agent it was assigned, and if its $a_{priorPerf}$ value is higher than the presenting agent's, allowing a high performing agent the opportunity to move to an agent it thinks better represents its prediction. However, this process uses the same fitness proportionate selection algorithm, which offers no guarantee that a move will be seen as beneficial to the watching agent. The limit of two moves accomplishes three things, (1) reducing compute time, (2) preventing a potential infinite loop of constant movement, and (3) introducing further diversity of opinion where agents may be assigned to support an agent they disagree with – conceptually similar to introducing randomness in genetic algorithms [72]. Once

assignment and agent movement is complete, the presenting agent represents itself and the watchers it was assigned; for example, if a presenting agent is assigned 10 watchers, that presenting agent's opinion is now worth 11 'votes' during the final aggregation process, 1 vote for each watcher and its own vote.

4.3 Confidence Intervals

The presenters and watchers now represent a swarm of agents which goes through iterative steps to arrive at a collective binary prediction with an associated confidence value. The confidence value is determined by repeating the process of assigning watchers to presenters and taking a vote of all presenting agents. While this process is running, the decision threshold is iteratively lowered if some higher threshold has not been met. The initial threshold is 100% agreement between all presenting agents, directly following the initial process of assigning watchers to presenters and taking a vote of all presenting agents. If this threshold is met, the prediction is considered "Very High Confidence", the prediction and confidence is returned and the swarming period ends. When there is no immediate agreement among all participating agents, each watcher performs an interaction with all of the other watchers assigned to the presenter they are assigned to, and an interaction with the presenter. For example, if there are 10 watchers assigned to a presenting agent, each of the watchers would perform 10 interactions, one with each of the other nine watchers, and one with the presenting agent. This interaction step follows the same protocols outlined in Chapter 3 and facilitates additional information dispersal between agents. Following the interaction step, a new group of presenting agents is randomly selected and the agents go through the same steps as previously outlined. This process can run for an additional 100 iterations with the threshold for agreement lowered to 90%; if the 90% threshold is met at any point the prediction is considered "High Confidence" and the swarming period ends.

When neither of the previous thresholds are met, the decision threshold is further reduced to 75% for an additional 50 iterations of selection, voting, and interaction. If the 75% threshold is met, the prediction is considered "Medium Confidence". However, if the threshold has still not been met, a weighted vote is taken from all presenters and watchers, and the prediction is "Low Confidence". The weighting is based on the agent's `certainty` score, with agents more certain in their opinion given more weight than those who are less certain. The confidence thresholds were initially selected using the breast cancer dataset described in section 4.2.1, based on experimentation, with the goal of 100% accuracy for the "Very High Confidence" interval. 100% predictive accuracy for the "Very High Confidence" category required agreement from all presenting agents immediately following the fitness proportionate selection step. For this dataset, when there

was not immediate agreement from all presenting agents, 100% accuracy was a very unlikely outcome for the "Very High Confidence" category. The number of iterations for each threshold was selected to balance classification performance with runtime. A future goal is to automate the selection of threshold values and number of iterations for each confidence interval, for each dataset and classification constraints. For example, some usage scenario may prefer more samples captured in the "Very High Confidence" interval while allowing for a lower predictive accuracy for that category, i.e. some accuracy lower than 100%.

The confidence categories can be summarized as:

- Very High Confidence: All presenting agents agree on a prediction class immediately following fitness proportionate selection,
- High Confidence: 90% of presenting agents agree on a prediction class within 100 iterations of the swarming process,
- Medium Confidence: 75% of presenting agents agree on a prediction class after 100–150 additional iterations of the swarming process,
- Low Confidence: Weighted vote of presenters and watchers if above thresholds are not met.

4.4 Results

4.4.1 Swarm Aggregation

Fig. 4.1 shows 5–fold validation results for predicting lymph node metastasis status from the clinical features listed in Table 4.1, with the output from the WVM on the left and the bee–inspired swarm aggregation method on the right. The folds used to test the WVM and the swarm aggregation method were the same, i.e. they were not re–randomized between tests. The averaged 5–fold results for accuracy, sensitivity, and specificity can be seen in Fig. 4.2. The swarming method improved average accuracy by 6.2%, the sensitivity by 10.9%, and the specificity by 2.0% compared with the WVM aggregation method. The worst case values for accuracy, sensitivity, and specificity were improved using the swarm method, with a worst-case accuracy in the WVM of 67% in fold 1 improved to 79.4% using the swarm. The bee-inspired swarm method was strictly an improvement for both the accuracy and sensitivity statistics. However, the WVM out-performed the swarm method for specificity in 2 of the 5 folds (folds 2 and 4), despite a 2% average improvement.

The WVM showed significant variance across each fold, which was reduced using the swarm aggregation method. Table 4.2 shows the variance across the same five folds shown in Fig. 4.1 for both the WVM and

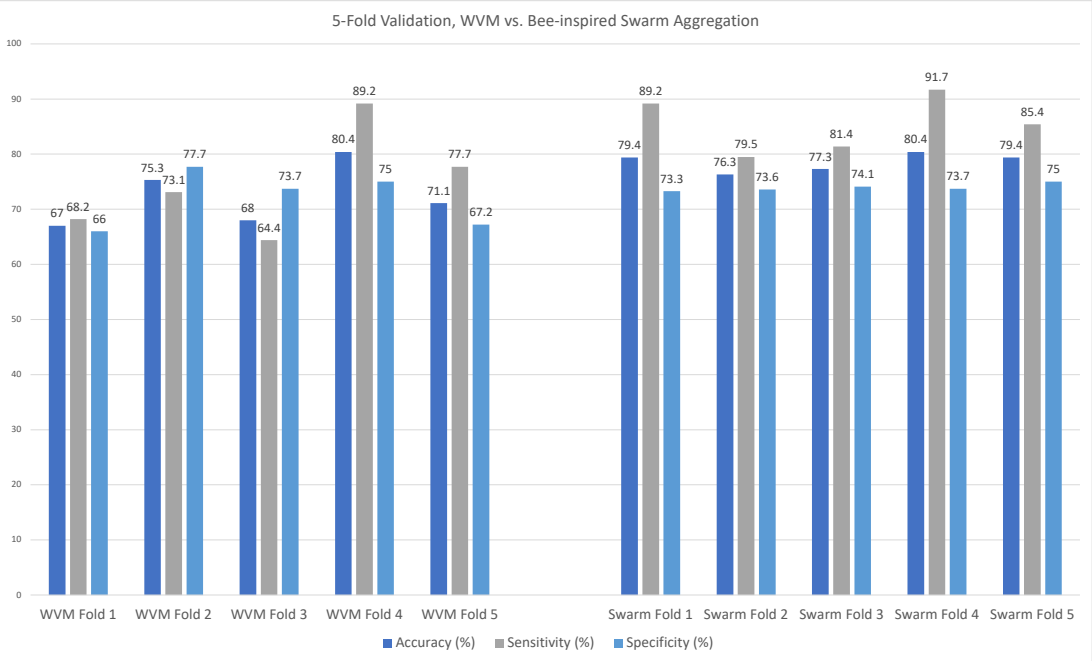

Figure 4.1: 5-fold validation results for predicting lymph node metastasis status from the clinical features listed in Table 4.1. Weighted Voter Model (left) vs. Swarm Aggregation (right)

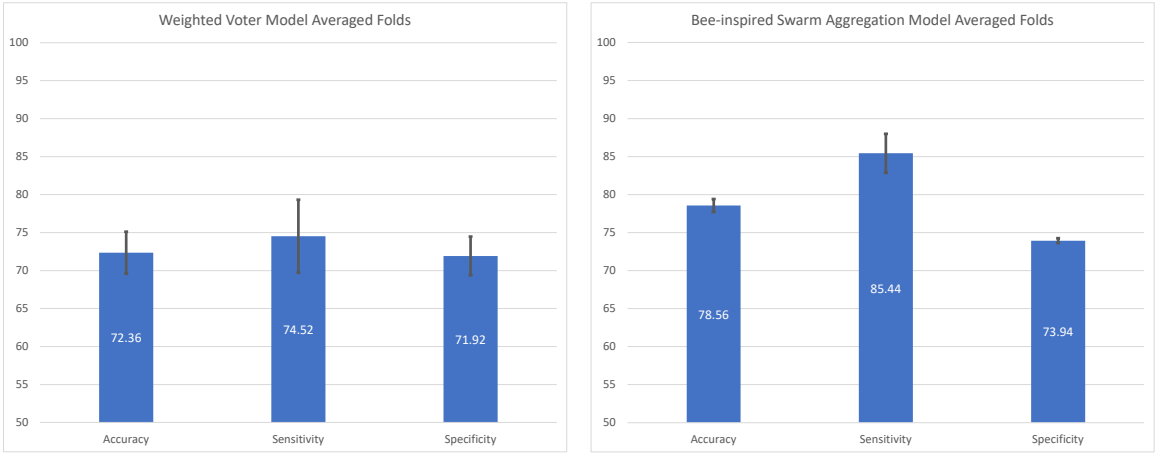

Figure 4.2: Averaged validation results for predicting lymph node metastasis status from the clinical features listed in Table 4.1. Weighted Voter Model (left) vs. Swarm Aggregation (right). Error bars represent standard deviation.

Table 4.2: Swarm Aggregation vs. Weighted Voter Model Variance (σ^2)

Method	Accuracy (σ^2)	Sensitivity (σ^2)	Specificity (σ^2)
Swarm	2.30	20.99	0.35
WVM	24.52	74.0	20.68

Table 4.3: Confidence Interval Distribution and Accuracy

Interval	$n = 483$	% of n	Accuracy (%)
Very High Confidence	3	0.62	100
High Confidence	154	31.9	93.1
Medium Confidence	224	46.4	82.3
Low Confidence	102	21.1	64.7
Very High + High + Medium	381	78.9	86.8

swarm aggregations methods, with the swarm method showing significantly reduce variance for accuracy, sensitivity, and specificity measurements.

4.4.2 Confidence Intervals

Four confidence intervals described in Section 4.2.2 were computed for the clinical patient features dataset. The results shown in Table 4.3 represent the fold–averaged results from Fig. 4.2. The "Very High Confidence" category was only able to capture 0.62% of patients and was not considered very useful for this dataset despite meeting the goal of 100% accuracy for patients in this category. The "High Confidence" category captured 31.9% of patients, with an accuracy of 93.1% and the "Medium Confidence" category captured 46.4% of patients with an accuracy of 82.3%. 21.1% of patients fell into the "Low Confidence" category with an accuracy of 64.7%. Eliminating all "Low Confidence" predictions (21.1% or 102 patients) from the output yields an accuracy of 86.8%, accounting for 381 of the original 483 patients (78.9%). This demonstrates the ability to stratify the subjects into confidence categories that improve the overall accuracy of the prediction for all but the lowest confidence predictions.

Any multi–agent–based system will have variances in outcomes based on the interactions between agents and between agents and the environment and as the number of agents change [69]. Table 4.4 shows the variance in the number of agents assigned to each confidence interval, as well as the variance in accuracy for each interval from five randomized simulations. Unsurprisingly, the variance in accuracy increases as the confidence decreases. Additionally, there is little variance in the number of patients assigned to each confidence category between simulations, indicating the categories are stable and not dependent on data ordering or agent initialization and interaction order as occasionally seen in the WVM and UWM aggregation mecha-

Table 4.4: Confidence Interval Variance (σ^2) Across 5 Simulations

Interval	$n = 483$ (σ^2)	Accuracy (σ^2)
Very High Confidence	0.222	0
High Confidence	8.667	0.082
Medium Confidence	2.889	0.376
Low Confidence	4.222	4.602

nisms. There is more variance in the "High Confidence" category than in the other categories, which should be investigated further with additional data.

4.5 Discussion

In this chapter I show the application of a swarm–intelligence–derived aggregation mechanism in a multi–agent, WoC–Bot architecture which improves classification accuracy, specificity, and sensitivity while reducing the variances associated with large, multi–agent systems by using a random ordering and interactions, as compared with the WVM aggregation method. Additionally, the honeybee–based aggregation mechanism allows for prediction stratification across confidence intervals, improving accuracy further for a subset of samples.

The confidence intervals show clear predictive performance differences, where predictions in higher confidence categories are more likely to be correct than those in lower confidence categories. However, the 'Very High Confidence' category captured less than 1% of samples, rendering it inconsequential for this dataset. Future work in this area to automate the confidence category delineation should be investigated, based on the dataset being analyzed, and dynamically updated as performance is tracked throughout the inference process. This requires dynamic updating of the iterative swarming process such that agreement threshold and number of iterations are changed automatically. The "Very High Confidence" category's predictive performance should not be degraded during this process. The swarm will track predictions made after initial roulette–wheel selection (therefore not be considered "Very High Confidence" under the current rules), but before the end of the "High Confidence" prediction point (currently 100 iterations) to determine if there is a new threshold for agreement that can be used, threshold y, after some number of iterations, x, where the prediction can still be considered "Very High Confidence". Agent–specific information, the predictive opinion of very high performing agents, may be useful during this process.

Chapter 5. Incremental Feature Addition

This chapter presents work and results on incremental feature addition for a classification process by incorporating new features which were unknown at the time of initial training. The WoC–Bots approach is able to quickly integrate these new features, as they become available, without the expensive retraining required in traditional networks. The results presented are based on the Hollywood data from Chapter 3, augmented with additional data about cast, production company, and producers and the patient data from Chapter 4, augmented with additional cellular morphology features. All additional data is incorporated into the classification task without retraining existing agents and leads to an improvement in the overall accuracy of the classification for both datasets.

5.1 Background

Another benefit of the multi–agent approach to classification versus a traditional monolithic neural network is the ease at which new input features can be added to the computation. While ANNs are the current 'state–of–the–art' approach to learning and classification problems, they require an expensive retraining operation when incorporating additional features that change the number of inputs to the network [48]. Some convolutional neural networks and recurrent neural networks allow for variable length input, which works well for image, text–based, and some time–series data, but comes with additional complexity and does not generalize well to typical binary classification problems[76, 56]. In general, most ANNs cannot expand horizontally, i.e. to include previously unknown feature(s), at the input layer, without retraining the network [135]. Transfer learning allows a model developed for one task to be used for a different, but related, task and can allow for an additional intermediate input layer to be included to process an increase or decrease in the number of input features [115]. However, transfer learning also adds complexity to a network and, importantly, does not guarantee that an existing model will perform well on the new problem [79]. The features that were used to train the original model need to be general enough to be predictive for both the original dataset and the new dataset [130].

While each WoC–Bot has a fixed number of inputs at creation, a restriction of each agent's internal MLP network, the interactive and swarm steps of this approach do not have restrictions on the number of participating agents. As noted in Chapter 3, agents with missing data simply do not participate. A natural extension

to the variable number of participating agents is the ability to include newly generated agents representing new data, a feature that does not require retraining any of the existing agents.

5.2 Incremental Feature Addition

5.2.1 Additional Features: Hollywood Success Prediction

The original features for the Hollywood success classification problem can be found in Table 3.1 and the additional features from the MovieLens dataset being considered are:

- Cast, top 5 listed
- Crew, top 5 listed
- Production company
- Director

where the cast is typically listed in order of importance (though sometimes order of appearance) and the crew can consist of writers, designers, photographers, audio engineers, and occasionally marketing personnel. Only the primary production company was considered, though typically larger budget movies have multiple production companies, and only the lead director was considered.

5.2.2 Additional Features: Breast Cancer Node Positive / Negative Prediction

The original features for predicting node–positive or node–negative status in breast cancer patients can be found in Table 4.1. The additional features considered for testing incremental feature addition are the following:

- Cellular mean area
- Cellular mean circularity
- Cellular mean eccentricity
- Cellular mean intensity
- Standard area
- Standard circularity

- Standard eccentricity
- Standard intensity.

The additional features were derived from whole–slide images of a representative hematoxylin and eosin (H&E) slide of the primary tumor and ID–matched with the existing patient data presented in Chapter 4.

5.2.3 Design: Feature Addition

When new features become available new agents are generated using a combination of the new features and the existing features. For example, a newly generated agent working on the breast cancer dataset may receive "Standard area", "Cellular mean eccentricity", "Age", "pT Max size" and "pT Stage" as features, combining two of the newly available features with three of the existing features. Combining the new and existing features in the newly generated agents solves two problems; (1) new agents need to know about the features that the existing agents are trained on, and (2), the existing agents need a way to determine `trust_score` and `prior_performance` metrics for the newly generated agents. Both the social interaction arena and swarm components of the method rely on social variables built around an agent's prior performance, trust, and existing knowledge of features. Chapter 3 describes how these values are used and modified when two agents interact with one another in detail, but without some estimated values for new agents, neither the existing agents nor the newly generated agents are able to effectively judge the value of another agent's prediction when interacting with a newly generated agent. In order to solve this problem, newly generated agents derive estimated `prior_performance` and `trust_scores` values from the existing features they receive. The estimates come from averaging these values from existing agents that have similar features. The averaging step does have some shortcomings; agents with similar, but not identical, features may have a very different predictions based on the features that differ between them. This can lead to inaccurate estimated values for `prior_performance` and `trust_scores` which do not accurately reflect the true values for the newly generated agent's `prior_performance` and `trust_scores`. However, each new agent has a shorter interaction and prior performance history compared with existing agents, which allows for larger initial modifications to these metrics during the interaction step, correcting inaccurate estimates quickly during the interaction phase.

5.3 Results

The following sections show the predictive performance improvements produced when including additional features for the Hollywood movie and breast cancer datasets. The goal is to show this method can incorporate new features as they become available without requiring a costly retraining of the full population of agents, and without significant loss of predictive performance compared with a full retraining. This process requires generating new agents, which increases the total number of participating agents. In both datasets, results are shown that control for additional predictive performance when including more agents in the simulation.

5.3.1 Additional Features Results: Hollywood Success or Failure

Fig 5.1 presents the 5–fold average accuracy when predicting the success of Hollywood movies using 600 agents in the leftmost column. In order to control for the possibility of a large accuracy improvement produced by simply increasing the number of agents, results are included for a test using the original Hollywood data but with an increased number of agents, from 600 to 1,000. Each of the additional agents received features in the same method as the existing agents. The second–from–left column shows the accuracy results from this test, with the accuracy improving from 81.14% to 81.38%, an improvement of 0.24%, indicating that increasing the number of agents may improve the accuracy minimally. The second–from–right column shows the accuracy after including the additional cast, crew, production, and director information, an increase to 86.77% using the same number of agents, 1,000. The rightmost column shows the accuracy when fully retraining all 1,000 agents with all of the features. There is a slight increase in accuracy when fully retraining the agents compared with incrementally adding agents after including the additional features, increasing 1.04% from 86.77% to 87.81%. However, the substantial accuracy increase came from including the new features, improving predictive accuracy from 81.14% to 86.77%, an improvement of 5.63%. Using 4x Nvidia GTX 1070 GPUs, an AMD Threadripper 1950x CPU, and an M.2 SSD for storage it takes 12 minutes to generate and train 400 new agents for the Hollywood dataset, and 43 minutes to fully retrain 1,000 agents.

5.3.2 Additional Features Results: Cellular Morphology

Figure 5.2 shows the original 5–fold average accuracy of the WoC–Bots method for predicting lymph node metastasis status in breast cancer patients. Testing with this dataset again controlled for the possibility

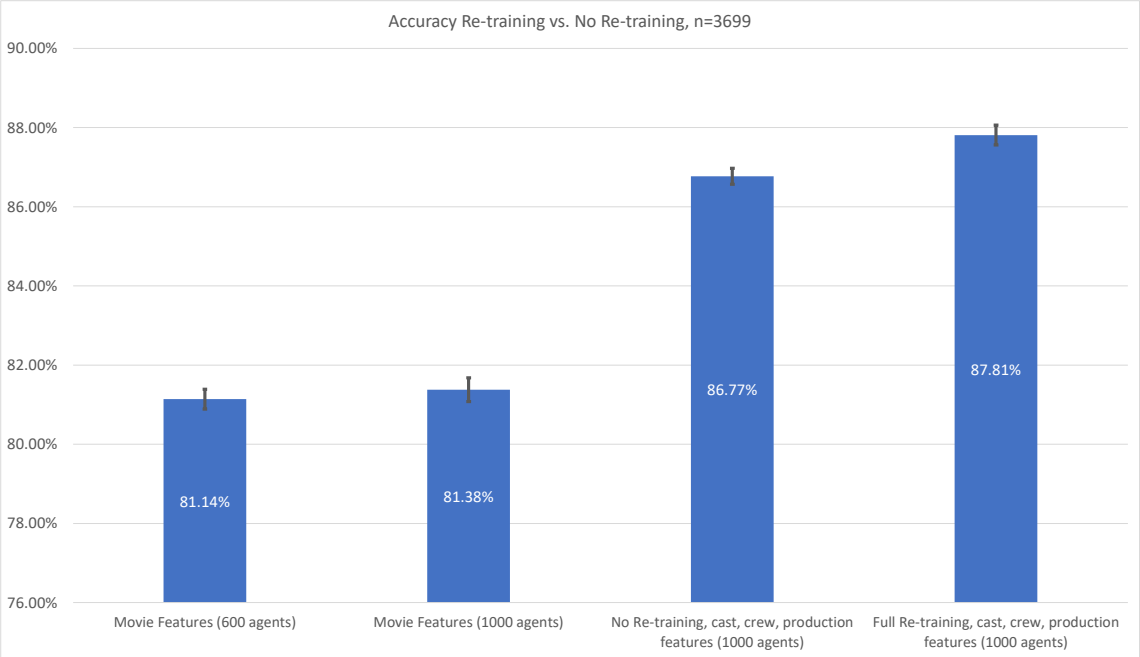

Figure 5.1: Cast, Crew, Production Features - Accuracy, averaged over 10 runs. Error bars represent standard deviation.

of more agents increasing the accuracy alone, with the second–from–left column showing the accuracy when 1,250 agents participated, increased from 750, using just the original clinical features. The accuracy improved by 0.13%, to 79.47% from 79.34%, again indicating that increasing the number of agents can improve the predictive accuracy minimally.

The second–from–right column shows the prediction accuracy after including the cellular morphology features with the original clinical features. The same number of agents (1,250) as the previous test were used and additional agents were again generated as outlined in Section 5.2.3. In this case the accuracy increased 3.49% from the original test (left column), and 3.36% as compared with the original clinical features and same number of agents (1,250), indicating the accuracy increase largely comes from incorporating the new cellular morphology data rather than from simply generating additional agents. The rightmost column shows the accuracy results after fully retraining the agents with the original clinical features and additional cellular morphology data, with 1,250 agents trained on all of the available data. This test shows a small increase in accuracy, 0.12%, compared with the prior method, but requires an expensive retraining of all participating agents. The time and computation cost of retraining depends on the number of agents and available hardware. Using the same hardware as described above, it takes 2 hours, 25 minutes to generate and train 1,250 new agents compared with 35 minutes to generate and train 500 new agents, a speedup of 4.1×.

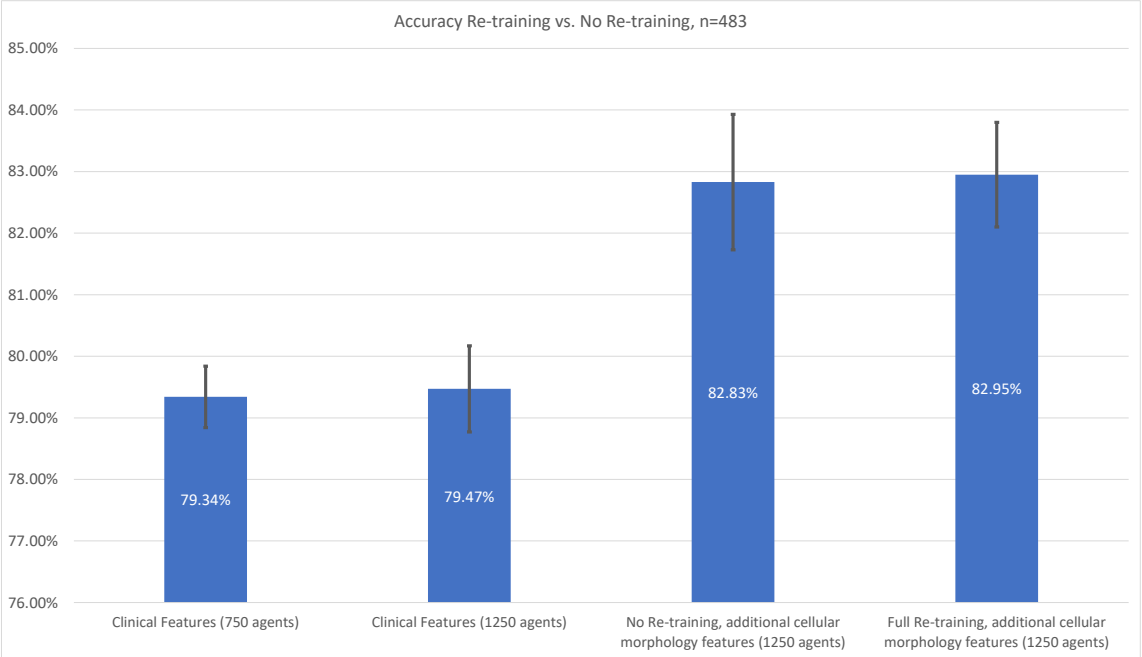

Figure 5.2: Additional Cellular Morphology Features - Accuracy, averaged over 10 runs. Error bars represent standard deviation.

5.4 Discussion

This chapter presents an extension to the design presented in Chapter 3, where agents with missing data do not participate in the interaction and swarming phases, leading to a variable number of participating agents based on the current sample being classified. Now, as new data with new features becomes available, this approach can incrementally incorporate the data without retraining existing agents by generating new agents and injecting them into the interaction and swarming steps of the classification process. While this requires additional participating agents, the results show the performance improvement comes largely from the additional data rather than by simply increasing the number of participating agents.

It is currently unclear why there was a larger increase when fully retraining agents for the Hollywood dataset as compared with the breast cancer dataset. One possibility is the additional features were more correlated with the output than in the breast cancer dataset, but this has not been confirmed using PCA. Given the 3.6× speedup in training time, the runtime advantage when incrementally adding agents compared with a full retraining of all agents is substantial. Future work could consider testing on additional datasets to determine if there are specific characteristics which allow some features to work better with incremental training compared with fully retraining all agents.

Chapter 6. Distributability

This chapter describes how WoC–Bots can be distributed across multiple compute nodes and presents the runtime improvements that results from parallel processing. This approach has three main phases, (1) the initial MLP training phase, (2) the social interaction phase, and (3) the swarm aggregation phase. All three phases have been designed to operate independently, with the work on the swarm aggregation phase leading to a new mode of operation, called a "meta–swarm", which allows agents to operate with an initial belief coming from an external source instead of an MLP network as described in Chapter 3. How the meta–swarm can be used for distributing the swarm phase in described in this chapter, and new uses for the meta–swarm are described in Chapter 7, which can lead to predictive performance improvements.

6.1 Distributed Design

The multi–agent nature of the WoC–Bots approach, which has no global control structure, lends itself well to being distributed across multiple computational nodes using clusters of independent machines. There are three main phases of this work to consider when discussing distributability, the per–agent training phase, where each small MLP network is trained, the social interaction phase, where agents share information in a virtual interaction environment, and the swarming phase, where agents both interact and vote before a classification is made.

6.1.1 Phase 1: Training

The training phase relies on a small amount of data which can be made available on a node–by–node basis or transferred across the network without significant delay. During this phase, each agent is independent and distribution is straight forward. Agents can be placed on each node randomly, or using an initialization algorithm similar to the initialization method discussed in Chapter 3 to optimally pick a node for each agent based on computational load or network limitations. The MLP network for each agent is trained locally on whichever node the agent is initialized on. Given the small size and depth of the MLP networks used, the training phase should not require powerful hardware. The results presented in this chapter initially distribute agents to each compute node randomly before training their MLP network.

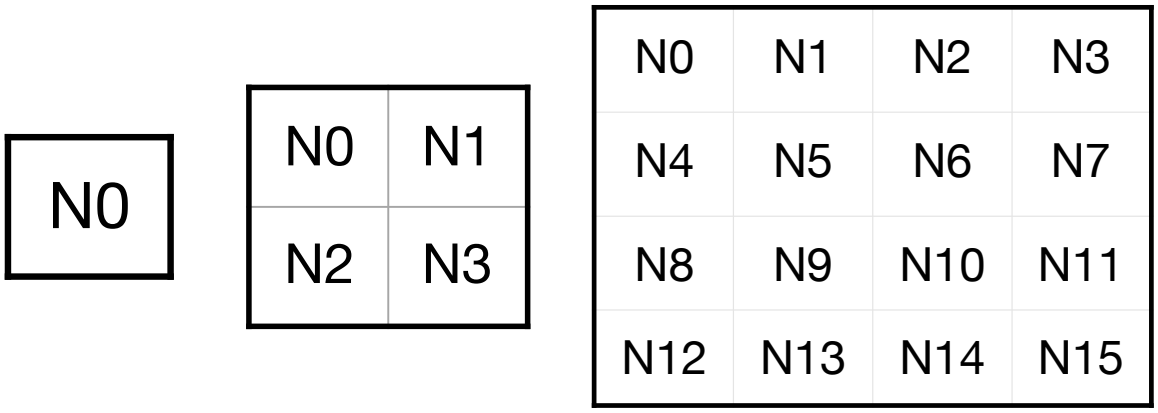

Figure 6.1: Representation of the full interaction arena for a single node, 4 nodes, and 16 nodes

6.1.2 Phase 2: Social Interaction

The social interaction phase initializes all participating agents in the interaction arena. The flexible design of the arena allows each compute node to represent some subdivision of the full arena, e.g. an arena with 4 nodes and 120 possible spaces can be subdivided so each node is responsible for a block of 30 spaces. An agent moving between each subdivision is moved between nodes. Similar to movement biasing on a single node, the movement can be modulated to encourage agents to remain on the node on which they were initialized by imposing a penalty for cross–node movement (requiring a network transfer), or it can allow for any movement without penalty, or even encourage agents to move between nodes as frequently as possible to better facilitate information dispersal.

Figure 6.1 shows three possible arena configurations, where the leftmost image represents a single node, the middle image represents four nodes, and the rightmost image represents sixteen nodes. The dark outline represents the bounds of the arena and the lighter internal lines present in the middle and rightmost images represent boundaries between the nodes. Agents can move between nodes freely during the interaction period, which requires network transfers, either explicitly from one node to another node, or implicitly through shared storage. All agent interaction information, and agent internal state are stored in SQLite databases which are written to disk, rather than stored in memory, at the end of each iteration. This work takes advantage of a disk–based database (SQLite) and a networked Ceph storage cluster [1] [2] that are connected to each node to implicitly share data on each node without explicit network transfers. As each agent crosses the boundary between nodes there is a flush of in–memory internal state to a SQLite table and a per–agent

[1] https://www.cs.drexel.edu/Account/public-cci-documentation.html
[2] https://ceph.io/en/

flag is set indicating on which node the agent is now present. At the start of each interaction step, each node checks a table to determine which agents are active on that node, and to ensure no agents are active on multiple nodes at the same time. On systems without shared storage between nodes it is possible to explicitly transfer agents using Java serialization [3], or any other preferred data transfer method.

6.1.3 Phase 3: Swarm Aggregation

The third phase of this method is the swarming phase. This phase can require frequent movement of agents between nodes due to the frequent re–selection process for presenters and watchers while the swarm is iterating. The selection process randomly selects presenters and then distributes agents to available nodes. This can reduce the benefit of distributing this phase across multiple compute nodes, due to the additional run time cost from network transfers.

The swarming phase can work in two distinct modes of operation. The first option requires a simple command and control node, which is responsible for selecting the presenters and watchers, as described in Chapter 4, and uniformly distributes the agents to available nodes. Each additional iteration of the swarming process does, however, require global knowledge and control to redistribute agents to each node. This method produces results identical to swarming on a single node, but is the slower of the two options due to frequent network transfers while redistributing the agents to new nodes.

The second option allows localized swarming on each node with aggregation at the end of the swarming process. Following the interaction phase, the agents can either stay on the node they are already present on, or agents can be randomly distributed to nodes. There is no additional movement of agents between nodes throughout the swarming process. In order to accomplish this, a method of representing each node's prediction was developed. Initial methods attempted to assign a weight to each node based on multiple factors, such as confidence category for the node's prediction, the average prior performance of agents on the node, the average prior performance of the top 15% of agents on the node, and the average confidence of all agents on the node and the top 15% of agents on the node. Of these methods, assigning a node weight based on average performance of the top 15% of agents produced the best results, however the accuracy of this method did not match the the previous option or single–node swarming.

Meta–Swarm Another method to represent each node was implemented and tested based on the idea of a "meta–swarm". The meta–swarm replaces the MLP within each agent with a prediction from an external (to

[3] https://docs.oracle.com/javase/8/docs/technotes/guides/serialization/index.html

the agent) source. Each source could be another classification method, a direct prediction, or in the case of distributed operation, the prediction from each node within the distributed compute cluster. This is conceptually similar to stacking in ensemble learning, where multiple weak learners predictions are used as input to a 'meta–learner' and a regression method is used to combine the predictions from the weak learners [126]. However, in the work presented in this dissertation, each prediction from WoC–Bots, another classification method, or a distributed node are encapsulated into agents, replacing the MLP step as originally described in Chapter 3. While the MLP training step is skipped, the agents encapsulating this prediction still operate as previously described, interacting socially and working together in a swarm to arrive at an aggregate prediction.

As with the above options, each node receives a random distribution of agents and the swarming process happens locally on each node. When the swarming process is complete, each node's prediction is encapsulated into an agent and these agents are moved to a single node to interact and swarm before a final prediction is made. While this does require a single node capable of running the swarming process, the maximum number of agents is equal to the number of nodes and each agent is computationally simple. This should not present computational limitations for any node that is also capable of participating in the swarming process. The meta–swarm method has shown similar performance to the first, network–intensive option, on two of the three datasets presented in this chapter.

6.2 Configuration

6.2.1 Airline Passenger Satisfaction Data

Distributed operations were tested with three different datasets; (1) the Hollywood dataset described in Chapter 3, (2) the breast cancer dataset described in Chapter 4, and (3) an airline passenger satisfaction dataset with a large number of samples, 120,000. Both the Hollywood and breast cancer datasets are small, with 3,699 and 483 samples, respectively. The airline satisfaction dataset allowed verification that agents are not over–fitting on specific input and provided a much larger test for distributed operation.

The airline satisfaction dataset has 22 input features, seen in Table 6.1. Passengers reported being either satisfied, neutral, or not satisfied. We considered neutral passengers to be dissatisfied for binary classification purposes. This more evenly split the dataset (48% satisfied and 52% not satisfied) compared with considering neutral passengers as satisfied. Not applicable features were represented with a value of 0 (zero). For example, "In-Flight WiFi satisfaction" was ranked on a 1–5 scale, with a value of 5 being very satisfied, and

Table 6.1: Airline features available for classification

Features	Description
Gender	Passenger gender: male, female, other
Customer type	Loyal or disloyal
Age	Customer age
Type of travel	Personal or business
Seat class	First, business, eco+, eco
Flight distance	Distance of journey
In-flight WiFi satisfaction	0 (N/A) 1-5
Flight time convenience	Satisfied with departure / arrival time
Ease of online booking	0 (N/A) 1-5
Gate location satisfaction	1-5
Food / drink satisfaction	1-5
Online boarding satisfaction	0 (N/A) 1-5
Seat comfort	1-5
In-flight entertainment	0 (N/A) 1-5
On-board service satisfaction	0 (N/A) 1-5
Leg room satisfaction	0 (N/A) 1-5
Baggage handling satisfaction	0 (N/A) 1-5
Check-in service satisfaction	1-5
Cleanliness	1-5
Departure delay	in minutes
Arrival delay	in minutes

a value of 1 being very unsatisfied; a value of 0 (zero) for "In-Flight WiFi satisfaction" indicates this feature was not present on the airplane, and not an applicable feature for classification for that customer.

6.2.2 Distributed Design

The following timing results are presented using virtual machines (VM) with the following hardware:

- CPU: AMD Opteron 6376 (Released Nov. 2012); 4 vCPU cores per VM
- RAM: 12 GB DDR3 per VM
- Ceph–base network storage

All virtual machines were configured with identical virtualized hardware, though with no control over how the physical hardware was divided by the virtualization software. For example, when running with four virtualized nodes, those four nodes could be powered by a single hardware CPU, or with each node on a dedicated hardware CPU, or some other possible combination with no user–facing way to determine the division of resources. 12 GB of RAM was enough to allow all agents to maintain themselves in memory without swapping memory to disk during both the training and interaction phases for the previously discussed datasets. Agents were distributed uniformly across each available node. An important note, the single–node

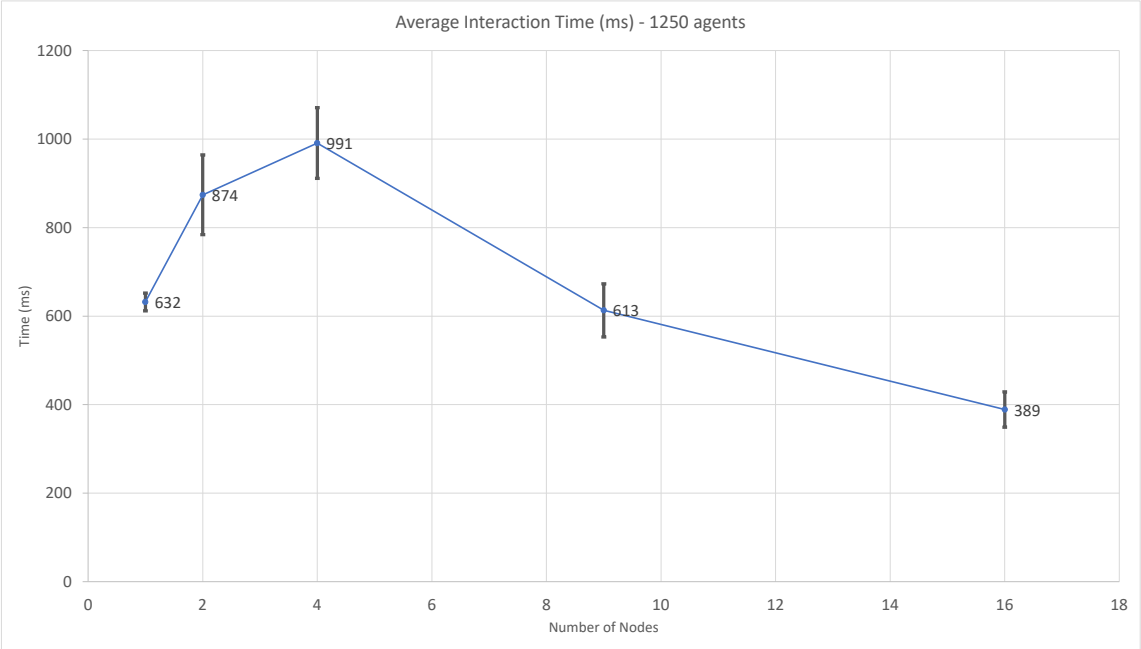

Figure 6.2: Interaction time (ms) for 1,250 agents on 1, 4, 9, and 16 nodes, using data from Chapter 4 with cellular morphology features, averaged over 25 runs. Error bars represent standard deviation.

interaction runtime is not representative of modern hardware, with the interaction step taking an average of 19ms for 1,250 agents and 96ms for 5,000 agents using a AMD Threadripper 1950x CPU and 64GB of RAM when testing the breast cancer dataset.

6.3 Distributed Processing

WoC–Bots, and the meta–swarm (depending on the number of inputs) both can have a large number of participating agents with many decisions influenced by a random number generator (which way to move, which node to be distributed to, etc.). Additionally, large multi–agent systems will have variances in their outcomes based on agent–to–agent interactions and agents interacting within their environment [69]. The results presented in this section are generated by the same seed to the main random number generator used by the system to guarantee that each run produces the same ordering of interactions and to hold predictive results consistent across multiple runs.

6.3.1 Runtime Performance

Figure 6.2 shows interaction run times (phases 1 & 2 only) for 1,250 agents on multiple node configurations. Runtime initially increases when using both 2 and 4 nodes, with testing showing this is largely due to network transfer penalties associated with frequent writes to the networked storage, as agents traverse

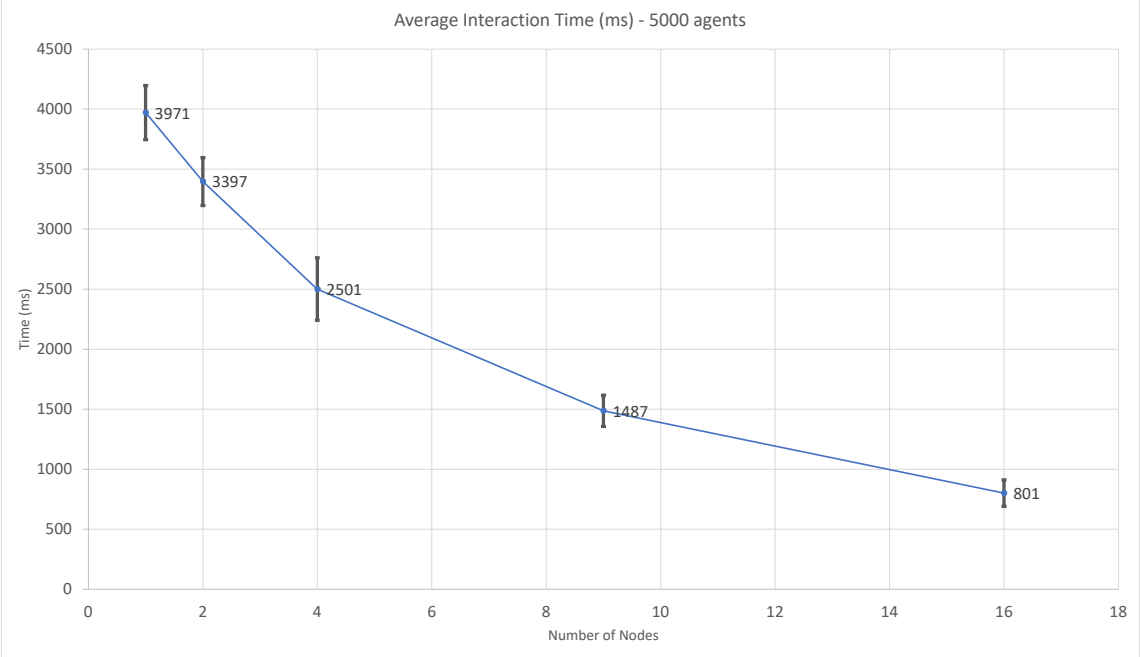

Figure 6.3: Interaction time (ms) for 5,000 agents on 1, 4, 9, and 16 nodes, using data from Chapter 4 with cellular morphology features, averaged over 25 runs. Error bars represent standard deviation.

nodes. However, 9 nodes out–perform a single node with continued runtime improvements when moving to 16 nodes. Figure 6.3 shows the same node configuration but with 5,000 agents participating. There is an increase in overall runtime, with runtime analysis indicating that CPU–bound tasks are the primary cause. The network transfer time takes a similar amount of absolute time in both the 1,250 and 5,000 agent configurations, indicating there is some initial time penalty. However, we do see a decrease in runtime moving from 1 node to 2 nodes, with further decreases in runtime as more nodes are included. Without additional hardware it is difficult to determine how many additional nodes will continue to improve runtime performance.

6.3.2 Swarm Performance - Timing and Accuracy

Figure 6.4 shows the average time all three phases take when allowing agents to move freely between available nodes. Using this method, the swarming phase operates exactly as it would when a single node is used, however it is distributed across multiple nodes, with agents frequently moving between nodes. Runtime initially decreases with 2 nodes, dropping from 9,128ms to 6,788ms, and continues to drop to 5,421ms with 4 nodes and 4,918ms with 9 nodes. However, runtime increases with 16 nodes, to 5,301ms. The swarming phase is not compute–intensive, but when free movement is allowed between nodes it is very network transfer–intensive, which is demonstrated with the increase in runtime when moving from 9 nodes to 16

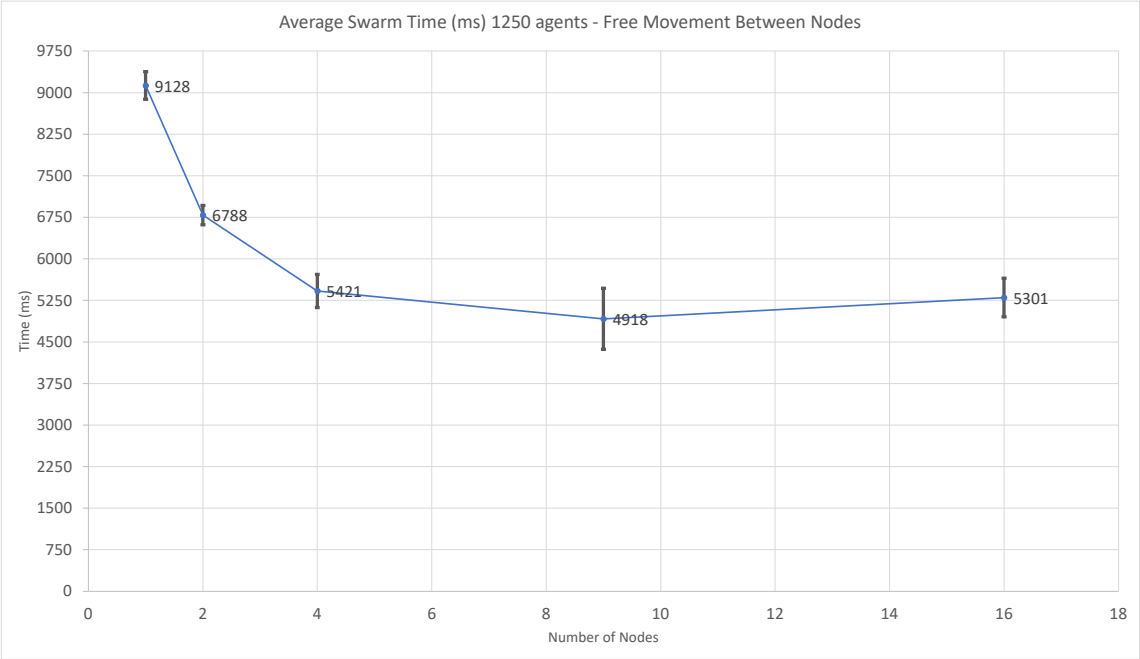

Figure 6.4: Swarm timing (ms) for 1,250 agents on 1, 2, 4, 9, and 16 nodes, using data from Chapter 4 with cellular morphology features, with agents moving freely between nodes, averaged over 25 runs. Error bars represent standard deviation.

nodes.

Figure 6.5 shows the average runtime for the swarm phase when disallowing agent movement between nodes. Agents must remain on the node on which they are initialized, all swarming processes happen locally on each node, with each node represented in a meta–swarm phase once localized swarming is complete. Using this method, the runtime continues to decrease with additional nodes, indicating the increase in runtime using the previous method was from network transfers between nodes.

Figure 6.6 shows the prediction accuracy using both of the above methods. In all cases the predictive performance decreases when swarming is node–local. The accuracy drops minimally in two of the three examples; dropping from 82.95% to 82.51%, a decrease of 0.44% in the breast cancer dataset and from 95.30% to 94.27%, a decrease of 1.03%, in the airline passenger satisfaction dataset. However, the accuracy drops substantially when predicting Hollywood movie success, from 81.0% to 74.97%, a decrease of 6.03%. It is currently unclear why this dataset has a larger decrease in predictive performance using the node–local swarming method, a phenomenon that can be studied in future work using additional datasets that may help isolate the performance issues when operating in the meta–swarm mode for the swarm aggregation phase.

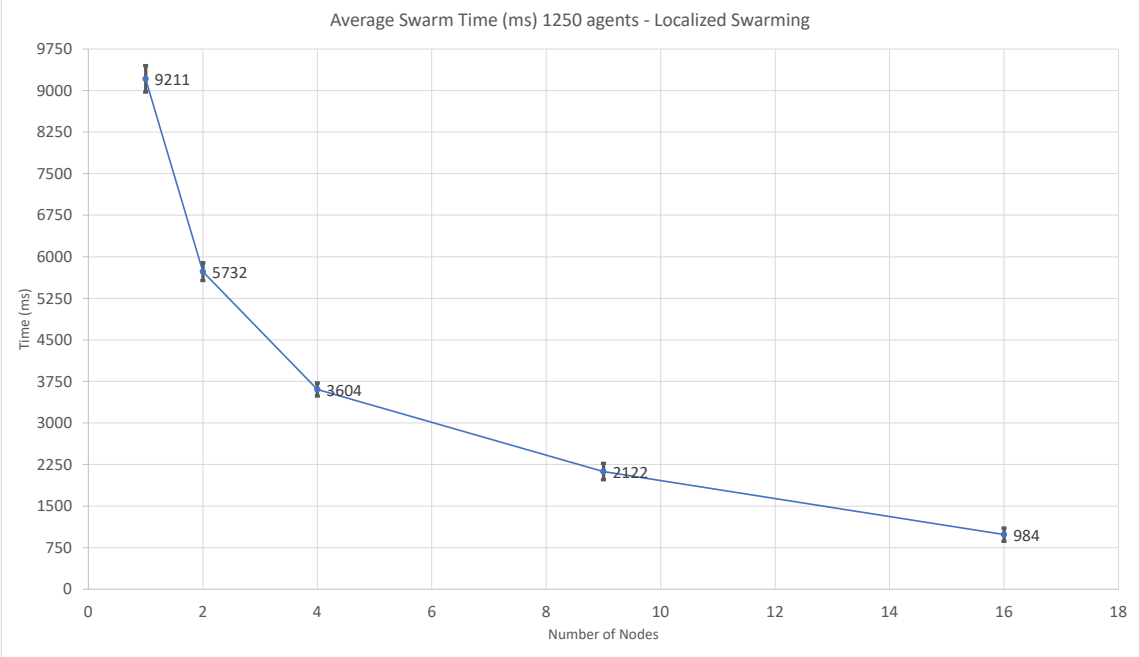

Figure 6.5: Swarm timing (ms) for 1,250 agents on 1, 2, 4, 9, and 16 nodes, using data from Chapter 4 with cellular morphology features, with node–localized swarming, averaged over 25 runs. Error bars represent standard deviation.

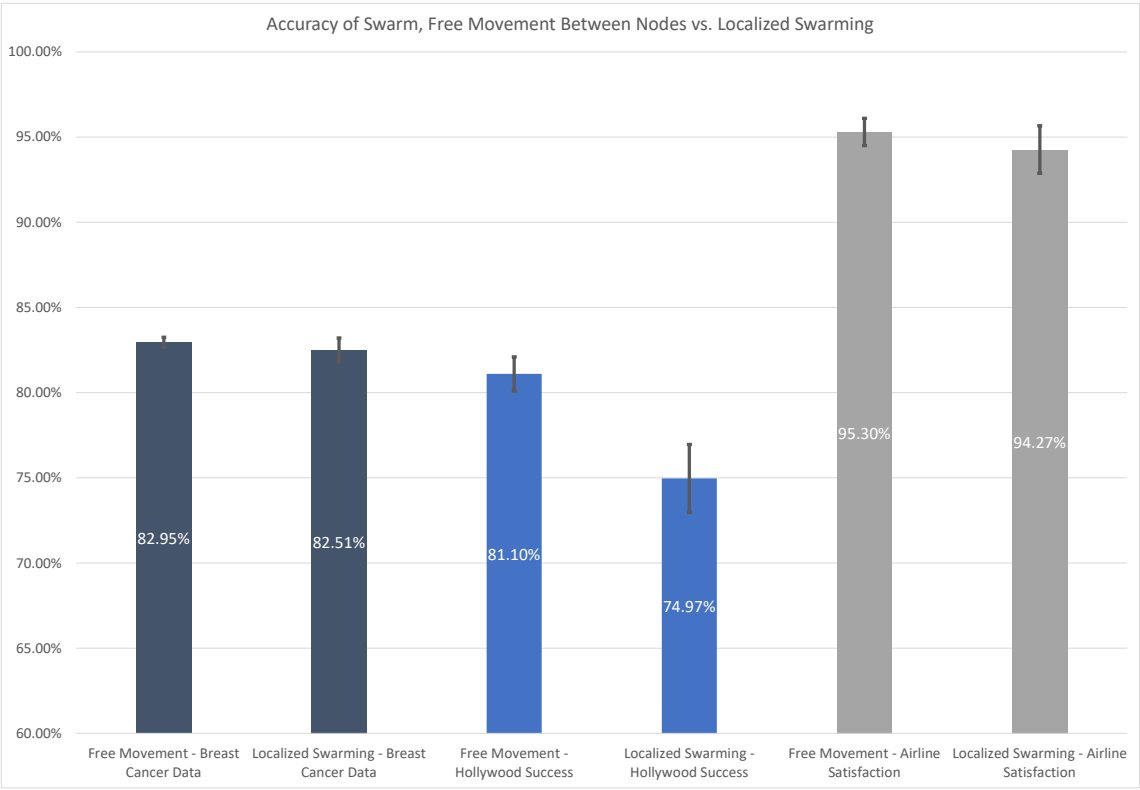

Figure 6.6: Comparison of prediction accuracy for breast cancer, Hollywood success, and airline passenger satisfaction when allowing free movement between nodes vs. node–local swarming, averaged over 25 runs. Error bars represent standard deviation.

6.3.3 Conclusion

This chapter shows how the three phases of this classification approach can be distributed across multiple compute nodes, showing how WoC–Bots can operate without a powerful machine to train a deep neural network. I demonstrated improved runtime performance when adding compute nodes for both the interaction and swarm steps. Importantly, the meta–swarm method developed to represent the prediction of each of the nodes has greatly reduced the runtime of the distributed swarm phase by allowing each agent to stay on a single node throughout the classification process. This method has shown similar accuracy to a single–node swarm in two of the three datasets tested.

Chapter 7. Classification Method Comparison & Application

7.1 Introduction

This chapter presents detailed comparisons between the WoC–Bots and meta-swarm methods described throughout this dissertation with existing "state–of–the–art" deep learning and ensemble learning classification methods. All experimental comparisons have been completed using the Hollywood data, first described in Chapter 3, the breast cancer patient data described in Chapter 4, and the airline passenger satisfaction dataset described in Chapter 6. All of the methods are tested by making binary predictions, and WoC–Bots are configured as described in Chapter 3, using the honeybee swarm aggregation mechanism described in Chapter 4.

The following classification methods have been selected for comparison based on their "state–of–the–art" status and common use in binary classification tasks [134] and can generally be divided into three categories: (1) deep learning, (2) ensemble learning, and (3) statistical modeling.

For category (1), a deep neural network was developed in DeepLearning4J, with a new network developed for each of the three tested datasets. Each network was developed with a suitable number of input nodes to represent the available features, and between three and seven hidden layers; the specific number of layers which produced optimal predictive performance for each dataset was used, and all hidden layers are fully connected.

For category (2), three ensemble methods were considered, random forest, AdaBoost, and XGBoost. In ensemble learning, multiple weak learners are combined to achieve better predictive performance or better efficiency compared with a single strong learning, like a deep neural network [96]. Random forest first builds numerous simple decision trees during training and, during inference, identifies the class which is predicted by the most individual decision trees [19]. AdaBoost is based in boosting in ensemble learning where each of the individual learners are trained sequentially and each subsequent learner should better represent the correct prediction. When a previous learner makes an incorrect prediction, the input data is weighted differently for a subsequent learning, giving a higher likelihood of a correct prediction from the subsequent learner [133]. In AdaBoost, each of the learners are assigned a weight based on their error rate during training, giving more "votes" to learners with less error, and the predictive class with the greatest total weight from all

learners will be the selected class for prediction [100]. XGBoost, like AdaBoost, is based in boosting but adds a regularization step to improve performance when over–fitting on input data is likely [23].

For category (3), logistic regression [30] was selected to compare performance with a pure statistical probability model which, while more efficient, would not be expected to have the same predictive performance of the other methods under consideration.

WoC–Bots have also been tested on sports betting data, making bets from from January 2021 through November 2022 on college and professional football (NCAA FBS, NFL) and basketball (NCAA Div. 1, NBA), professional hockey (NHL), baseball (MLB), and soccer (multiple leagues). The approach was tested primarily on "spread" and "moneyline" bets. In spread bets, the favorite needs to win by some pre–defined margin, or the losing team needs to lose by less than some pre–defined margin. In "moneyline" bets, the team that wins the event also wins the bet. Some testing was done on predicting the total score of a game, but this was not successful in early attempts and no further investigation was conducted. In total, roughly 10,500 bets were tested and the value of each bet was typically around 0.5 - 1% of available 'units'. For example, if there were 1000 units available, each bet would typically represent 5 to 10 'units'.

An additional use of the multi–agent design and aggregation mechanism, a meta–swarm, was developed to incorporate predictions from other sources. As described in Chapter 6, the MLP belief for each agent can be replaced with a belief from a distributed compute node, other classification methods, or predictions from external sources. Included in the comparison and sports betting results are results from the meta–swarm; in the comparisons with other classification methods, the prediction from each method was encapsulated as a belief in an agent, and the agents were allowed to interact and swarm before making a final prediction. In the sports betting testing, WoC–Bots were used to make predictions about sporting events, and direct predictions from other sources were also used, being encapsulated into agents before allowing all of the agents to interact and swarm to arrive at a final prediction about some event.

7.2 Datasets

The datasets used in this chapter have been described previously in detail. Details for the Hollywood data can be found in Chapter 3, with specific features seen in Table 3.1. Details for the breast cancer data are available in Chapter 4, specifically using the clinical features from Table 4.1. Neither the additional cast, crew, production data from the Hollywood dataset nor the additional clinical patient features presented in Chapter 5 were used for the comparisons in this chapter due to the amount of missing data for those features;

the methods used for comparison throughout this chapter do not easily work with missing features. The airline passenger satisfaction data used is the same as described in Chapter 6.

7.2.1 Sports Betting Data & Background Information

The sports betting data was used to demonstrate the flexibility of the agent–based design, making an initial classification prediction using data as described in this chapter, then also including external, independent prediction sources for spread and moneyline bets. WoC–Bots initially make a prediction about some sporting event based on the available data, then additional agents are generated using the external prediction as a replacement for the MLP network belief. Agents with the external prediction, as well as a representative agent for the WoC–Bots prediction, begin a meta–swarm phase, sharing and aggregating all of the available information before a final prediction is made. The sports betting results initially operated in a WoC–Bots–only state before switching over to using the meta–swarm for all subsequent predictions. The switch happened for two reasons; one, the meta–swarm was not developed when WoC–Bots were first tested with sports betting, and two, the meta–swarm produced a higher return on investment as compared with WoC-Bots alone.

This approach was tested on sports betting data from January 2021 through November 2022 on college football, National Collegiate Athletic Association (NCAA) Football Bowl Subdivision (FBS), professional football, National Football League (NFL), college basketball (NCAA Division 1), professional basketball, National Basketball Association (NBA), professional hockey, National Hockey League (NHL), professional baseball, Major League Baseball (MLB), and professional soccer, multiple European, South American, and Asian-Pacific leagues as well as Major League Soccer (MLS). As noted, the method was primarily tested on spread and moneyline bets. In spread betting, each team is given a point spread; for example, in basketball, the favorite may be given the spread "-6.5" and the underdog would then be given "+6.5". In spread betting, you are betting on the favorite winning by more than 6.5 points, or the underdog losing by less than 6.5 points. Selecting the favorite at a "-6.5" spread line for a bet means the bet wins when the favorite wins by 7 or more points, but your bet loses even if the favorite wins, but wins by less than 6.5 points. Similarly, when selecting the underdog to lose by less than 6.5 points, a spread line of "+6.5", means your bet wins if the underdog wins the match, or loses by less than 6.5 points. Moneyline bets are more straightforward, with the team who wins the match also winning the bet. If you bet on team *a*, and team *a* wins, the bet also wins. If team *a* loses then the moneyline bet loses. Soccer has more than two outcomes in typical betting; win, loss, or draw. In this case, using moneyline bets, if you select team *a* to win and team *a* loses or draws, the

bet is lost. Moneyline betting is a more common bet type for lower scoring sports like baseball, hockey, and soccer. Moneyline bets are also common in basketball and football when teams are evenly matched and the spread would be very small, less than 2.5–3.5 points.

Each bet will have some assigned "odds" from a sportsbook. Odds from different sportsbooks are typically similar, though deviations of 5% are not uncommon. Throughout this work, the best available odds were used from United States–based sportsbooks. Spread bets and total points bets are typically designed so the odds are as close to even as possible. This encourages bets on both sides of the line; that is, trying to balance the number of bets picking the favorite and the underdog. The odds for these types of bets are typically between "-105" and "-115", meaning to receive 100 units back (on top of the original bet amount) from a winning bet you would need to wager 105 units or 115 units, respectively. Truly even odds would indicate a line of "+/- 100", but the sportsbooks earn between 5 - 10% of each wager, so a line of "-110" is considered "even odds".

Moneyline bets will stray further from the even line, while still trying to balance the number of bets on each side of the line. Encouraging moneyline bets on an underdog, which is less likely to win, requires incentivizing wagers on that team. Sportsbooks do this by offering more money back than wagered should the bet on the underdog win. Favorites will have negative odds, "-150", for example, while underdogs will have positive odds, "+150", for example. While a negative line shows you how many units must be wagered to receive 100 units in return (in addition to the original bet amount) for a winning bet, positive odds show how many units are returned when 100 units are wagered. For example, a line of "+300" would return 300 units if 100 units were wagered and the bet won, plus the original 100 units, for a total of 400 units returned. Betting on underdogs, with positive odds, means you can win a lower percentage of bets wagered while still increasing the amount of units earned.

The odds assigned to each event can be used to determine what the sportsbook thinks about the likelihood of some outcome, the implied probability. Even odds indicate a 50/50 chance of an outcome, while +200 odds indicate an implied probability of 33.3% the underdog wins the match-up. The formulas for calculating implied probability, $probability_{implied}$, from the odds can be seen in Eq. 7.1 for negative, or favorite, odds, and Eq. 7.2 for positive, or underdog, odds.

$$probability_{implied} = |odds_{favorite}|/(|odds_{favorite}| + 100) * 100 \tag{7.1}$$

$$probability_{implied} = 100/(odds_{underdog} + 100) * 100 \tag{7.2}$$

For example, using Eq. 7.1 for a favorite with odds of -300:

$$300/(300 + 100) * 100 = 75\%,$$

gives an implied probability that the favorite team wins 75% of the time. When selecting bets, the goal is to maximize the value of the wager. Any time a betting system (WoC–Bots or otherwise) believes the probability of a team winning the bet is greater than the implied probability, there is value in placing a wager.

Another common type of betting is total betting, that is predicting whether the total points scored in an event will be higher or lower than some number. For example, a college basketball game may have a "total line" of 145.5 points. If the combined score of both teams is 146 or higher, then the "over" bet wins, if the combined score is less than 145.5 then the "under" bet wins. Some testing was done on predicting the total score of match-ups, but this was not successful for any of the major sports. Professional soccer moneyline bets were tested with three possible outcomes: win, loss, and draw. However, draw was not considered as a possible outcome due to limitations in non–binary predictions for this method. In matches where the odds suggested a high likelihood of draw as an outcome, those matches were not wagered on. For example, team *a*: +130, team *b*: +150, draw: +205 indicate neither team has a large expected advantage and the odds for a draw are low compared to more typical draw odds (+260–+320). The moneyline bets for all other tested sports were binary: win or lose. In total, roughly 10,500 bets were tested and the unit size of each bet was typically around 0.5 - 1% of total 'units' available for betting. For example, if there were 1000 units available, the size of each bet would be 5 to 10 'units'. Some spread and total bets can have a line with a whole number of points, for example "+6" for a spread bet, or "145" for a total bet. In cases where the favorite wins by 6 (and underdog loses by 6), or the total score is exactly 145, the bet is considered a push. You neither win nor lose, it is as if the bet did not happen and any units wagered on the event are returned.

Sports betting data was sourced from numerous independent data sources. The data used includes historical betting odds data for the NHL, NBA, MLB, NFL, and NCAA basketball and football. Historical odds were also available for the five major European soccer leagues – English Premier League, French Ligue 1, Spanish LaLiga, German Bundesliga, and Italian Serie A. Freely available historical data for the United States MLS soccer league was found to be inaccurate when compared with the official MLS website and was not used. Official MLS data has no publicly available API and was not used. Odds data for 2nd and 3rd

tier European soccer leagues were also found to be inaccurate for certain years and accurate for others after comparing multiple freely available data sources. Only years with data consistent from multiple sources were included in the training set.

Play–by–play data was available for the NBA, NFL, and MLB. However, this play–by–play data was only used to build season–long statistical data for players and teams when that data was otherwise unavailable through free sources. Some initial testing has shown the granularity of play–by–play data increases the noise, which then decreases predictive performance. Roster data, injury information, and game–by–game statistics were found on ESPN.com [1], Yahoo! Sports [2], and The Action Network [3].

ESPN.com was used to determine upcoming matches. The Action Network publishes odds from major US–based sportsbook and was used to determine the best betting odds for upcoming games. For each game, agents were given information about the competing teams. Statistical information for each team was pulled from ESPN or Yahoo! Sports, looking at the past 3, 5, and 10 games, as well as season–long trends and associated statistics. Points scored versus points allowed for NBA teams over the same 3, 5, and 10 games is an example of statistical information available as input. Roster information for each team was gathered from ESPN, and statistics for the listed players were populated from both ESPN and Yahoo! Sports. Players' season statistics, as well as previous 3, 5, and 10 games statistics were included. In some sports, match–up specific statistics were also available, this included the NBA, NHL, and MLB. MLB statistics also included pitcher versus expected batting lineups from Yahoo! Sports. The current betting lines were then pulled from The Action Network and the agents were asked, "will team a win by 5 points" if the spread betting line is "-5". Due to the limitations of making binary predictions, the agents will also be asked the same question on either side of the line until the prediction changes. That is, for the same game, agents would also be asked "will team a win by 4 points", "will team a win by 6 points", "will team a win by 3 points, "will team a win by 7 points", and so on until the prediction changed. Both the associated confidence from the swarm step and the distance from the published line when the opinion changed influenced the overall confidence in the prediction, with greater distance from the published line giving more confidence in the prediction. For example, if the agents predict team a does not win by 6 points, there is less confidence in the published -5 betting line than when the agents predict team a will win by 9 points. Spread bets were placed when there was medium to high confidence from the swarm, and when it was predicted a team would cover the spread by

[1] https://www.espn.com/
[2] https://sports.yahoo.com/
[3] https://www.actionnetwork.com/

at least one point. Moneyline bets were placed whenever there was medium to high confidence of winning.

Predictions from other sources were also gathered and included in the meta–swarm, replacing the MLP step as described in Chapter 6, Section 6.1.3. Predictions came from ESPN.com (BPI rankings), ESPN.com (fan rankings), Yahoo! Sports (fan rankings), The Action Network ('public' betting, 'expert' picks, 'sharp' action), SportsChatPlace [4], CollegeFootballNews [5], Winners and Whiners [6], FiveThirtyEight [7], and Sportskeeda [8]. Some sources have a numerical value associated with the prediction which was treated as a confidence metric. For example, Yahoo! Sports shows the percentage of votes on either side of the spread or moneyline bets which is used to represent public sentiment. ESPN.com BPI rankings have a match-up predictor which aggregates internal metrics developed by ESPN and presents a percent chance of winning (moneyline only) for either team, which can be used to represent confidence in either team winning the match-up. Other websites often, but not always, include a similar metric to represent confidence; when available, the confidence information was included with each external prediction.

7.3 Results

7.3.1 Classification Method Comparison

The following sections show comparisons on three datasets with the WoC–Bots method, the meta–swarm method and five "state–of–the–art" classification methods: AdaBoost, XGBoost, random forest, Deep Neural Network, and logistic regression.

Breast Cancer Metastasis Figure 7.1 shows the comparison for accuracy, recall, and precision of WoC–Bots with the other classification methods for a breast cancer dataset, predicting node–positive or node–negative disease in breast cancer patients. The results presented here were generated using the clinical features identified in Table 4.1. The leftmost set of columns show results for the WoC–Bots method, the rightmost set of columns show results for the meta–swarm method. The other sets of columns represent other classification methods, XGBoost, AdaBoost, random forest, DNN, and logistic regression. The meta–swarm encapsulates each prediction from the six methods as an agent to be included in the social and swarming phases of the meta–swarm. Random forest produced the best accuracy at 86.9% and the best precision results at 84.0%. The DNN implementation produced the best recall at 92.5%. WoC–Bots out–performed

[4] https://scpbetting.com/
[5] https://collegefootballnews.com/
[6] https://winnersandwhiners.com/
[7] https://fivethirtyeight.com/sports/
[8] https://sportskeeda.com/

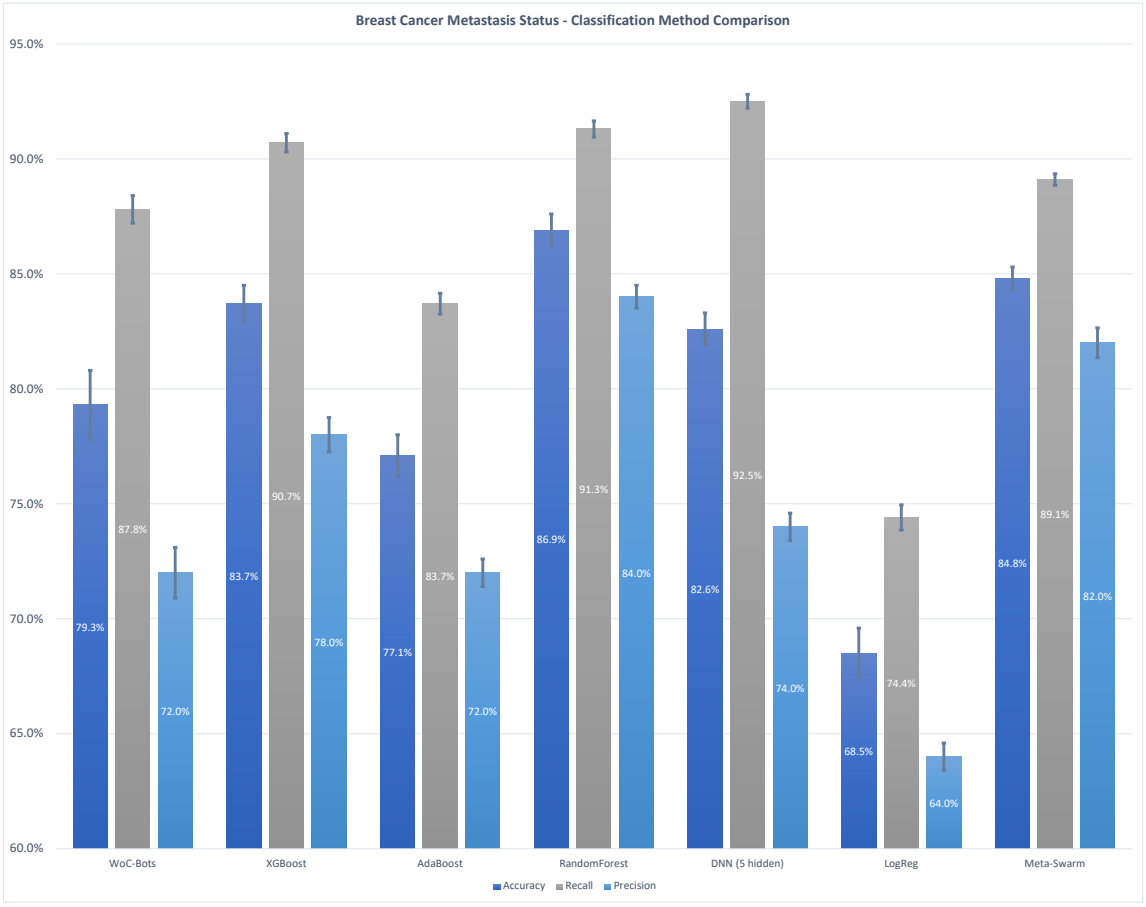

Figure 7.1: Comparison of five classification methods with two versions of the swarm (first and last columns) for the breast cancer dataset, results averaged over 10 runs of the algorithms. Error bars represent standard deviation.

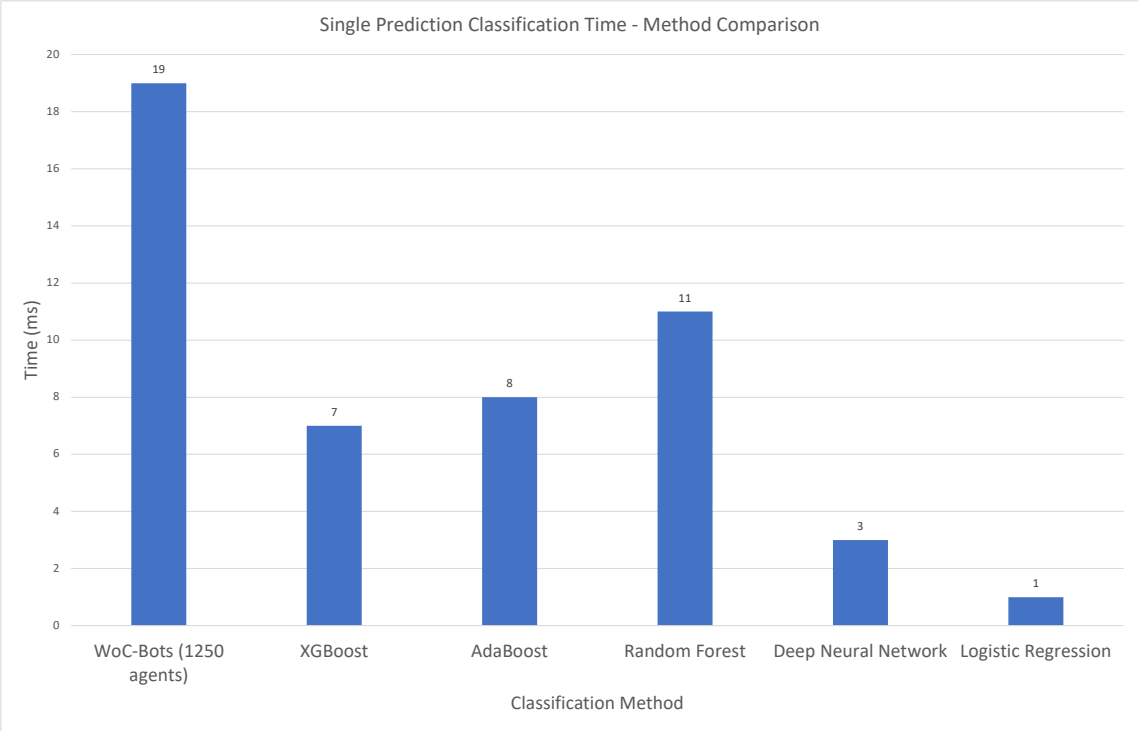

Figure 7.2: Runtime Comparison for a Single Prediction for All Methods.

AdaBoost and logistic regression in accuracy and recall, but had the second lowest precision results at 72.0% (tied with AdaBoost). The meta–swarm implementation, while not producing the best results for any of the three metrics, improved consistency between the three metrics, produced the second highest accuracy at 84.8%, and the second highest precision at 82.0%.

Figure 7.2 presents computation times, in milliseconds, needed to perform a single classification for this dataset using WoC–Bots and other methods. The timing results for all methods were run on the same hardware – AMD Threadripper 1950x CPU, 64GB of RAM, and 2x Nvidia 2070 GPUs. WoC–Bots and the deep neural network were implemented in Java. The other methods were implemented in Python. As expected, methods which require additional iterative steps take longer to complete, with the WoC–Bots method having the most steps, and taking the most time. The timing results here were consistent across multiple runs of the algorithms using the millisecond time scale.

Hollywood Movie Success Figure 7.3 presents the same information as Figure 7.1 but using the Hollywood movie success dataset. A movie was considered successful if the revenue was greater than 2× the initial budget for the movie. The results presented here use the features listed in Table 3.1 only, not including the addition cast, crew, production, and director information. WoC–Bots performed very similarly to a

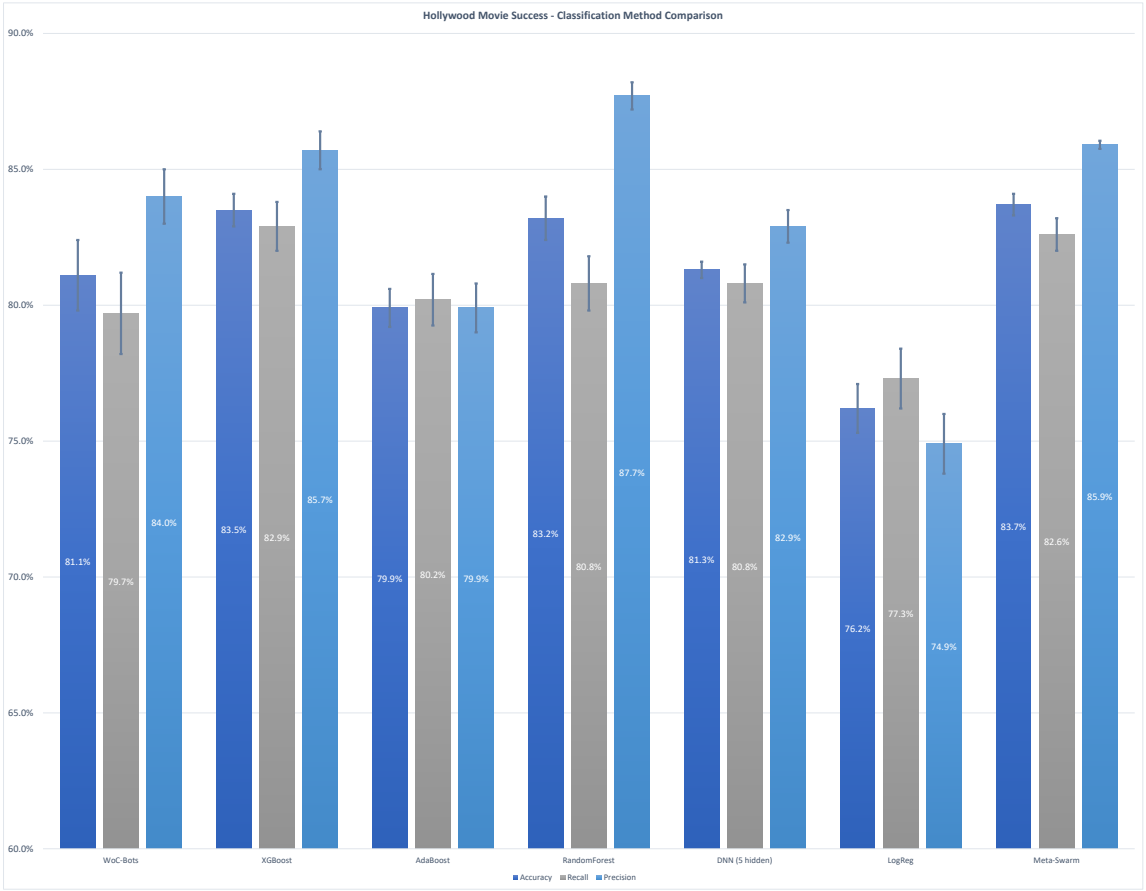

Figure 7.3: Comparison of five classification methods with two versions of the swarm (first and last columns) for the Hollywood movies dataset, results averaged over 10 runs of the algorithms. Error bars represent standard deviation.

Table 7.1: Confidence Interval Distribution and Accuracy - Meta-Swarm for Hollywood Success

Interval	$n = 3699$	% of n	Accuracy (%)
Very High Confidence	51	1.4	96.1
High Confidence	958	25.9	91.6
Medium Confidence	1261	34.1	84.5
Low Confidence	1428	38.6	77.3
Very High + High + Medium	2271	61.4	87.8

DNN implementation, with 81.1% accuracy compared to 81.3% for the DNN. WoC–Bots out–performed the DNN with a precision of 84.0% compared with 82.9%, but were out–performed on recall with 79.7% for WoC–Bots and 80.8% for the DNN. The XGBoost and random forest methods again out–performed the other classification methods on the accuracy and precision metrics.

The meta–swarm was again tested, out–performing all other methods with an accuracy of 83.7%. Additionally, only the XGBoost method had a better recall value at 82.9% compared with the meta–swarm's recall of 82.6%. Confidence intervals were tested with this dataset, but applied to the output of the meta–swarm instead of directly to WoC-Bots. Table 7.1 shows the results of this analysis, with 61.4% of samples captured in the "Very High", "High", and "Medium" confidence intervals with a combined accuracy of 87.8%. Additionally, the "Very High" interval was more useful for this dataset as compared with the breast cancer dataset (Table 4.3), capturing 1.4% of samples (instead of 0.62%), with an accuracy of 96.1%. The "High" interval captured over 25% of samples with an accuracy of 91.6%.

Airline Passenger Satisfaction Figure 7.4 presents the same classifications methods as Figures 7.1 and 7.3. This dataset, however, was far more predictive than the previous datasets, with accuracies over 95% for all methods except logistic regression. Additionally, all methods were consistent with a slighter higher precision than accuracy, and a slightly lower recall than accuracy. WoC–Bots performed similarly to XGBoost, AdaBoost, and random forest, better than logistic regression, and slightly worse than the deep neural network. With most methods already performing well, the meta–swarm did not improve over any of them, though it did reduce variability between accuracy, precision, and recall. Table 7.2 shows the confidence intervals for the meta–swarm for this dataset. Over 15% of samples fell in the "Very High" interval, over 70% in the "High" interval with the remaining 15% split almost evenly between "Medium" and "Low" confidence. The accuracy values for each interval were very similar, with the "Very High Confidence" interval being out–performed by the "High" and "Medium" intervals. The confidence intervals were not found to be useful for this dataset; the airline data is highly predictive which made it difficult to stratify samples into confidence

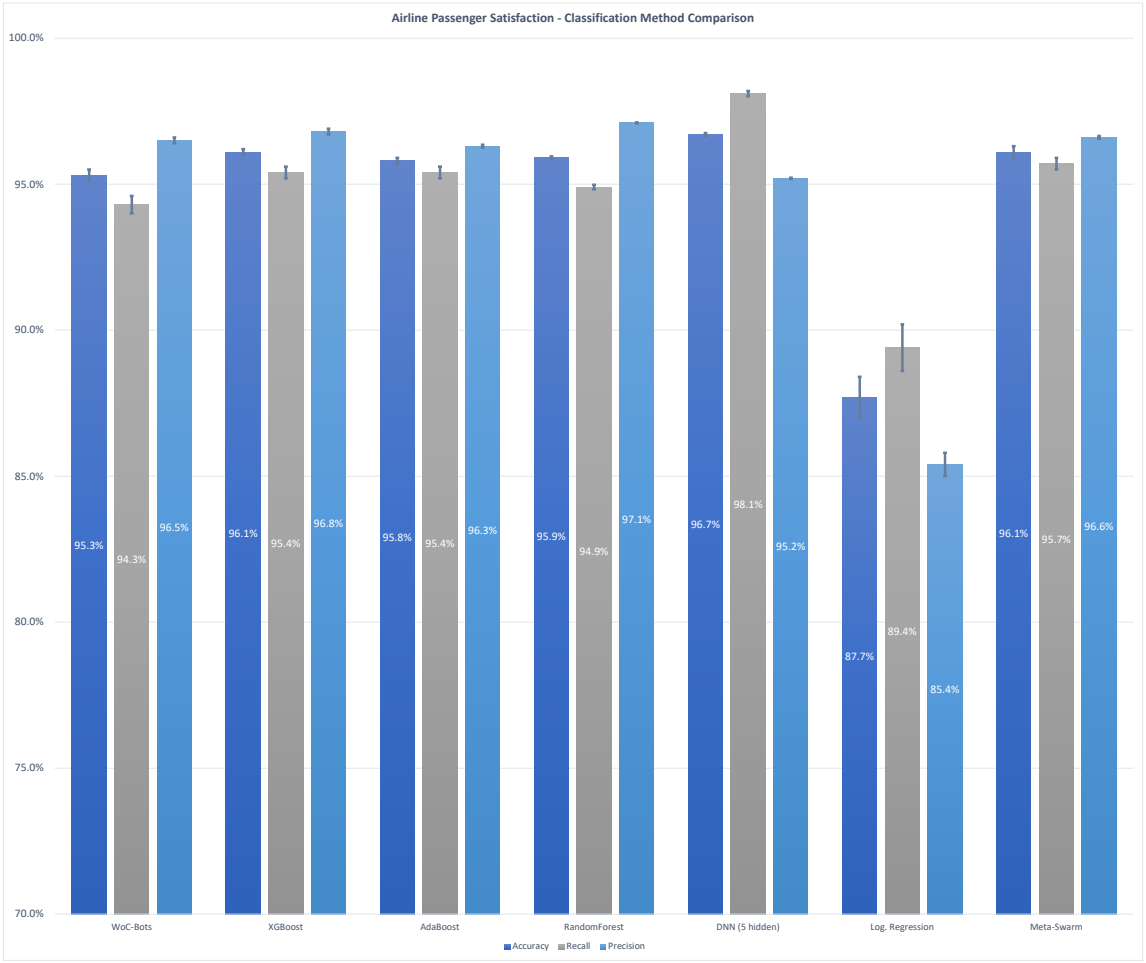

Figure 7.4: Comparison of five classification methods with two versions of the swarm (first and last columns) for the airline passenger satisfaction dataset, results averaged over 10 runs of the algorithms. Error bars represent standard deviation.

Table 7.2: Confidence Interval Distribution and Accuracy - Meta-Swarm for Airline Passenger Satisfaction

Interval	$n = 129882$	% of n	Accuracy (%)
Very High Confidence	19797	15.24	95.7
High Confidence	92125	70.93	96.2
Medium Confidence	9109	7.01	95.9
Low Confidence	8851	6.81	95.7

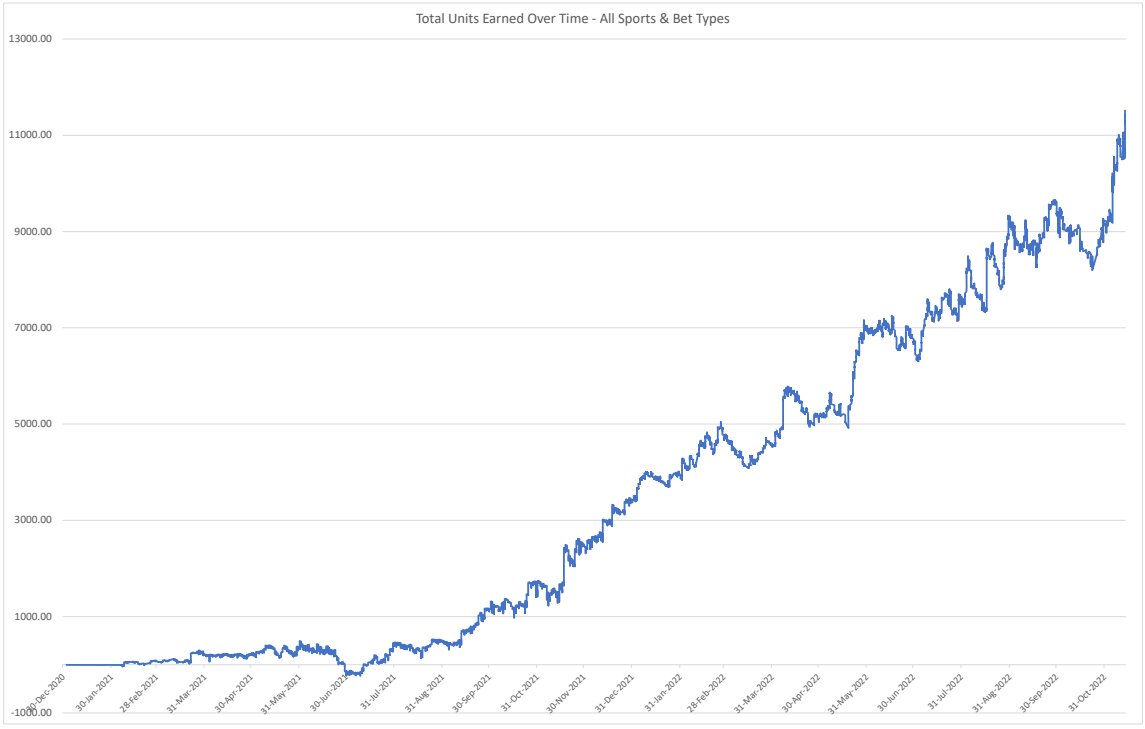

Figure 7.5: Total units earned over time for all sports tested on, moneyline and spread bets, 2021-01-1 through 2022-11-15

intervals, a limitation which can be explored in the future.

7.3.2 Sports Betting

The WoC–Bots classification method has been applied to sports betting with some success over a 20–month period and most major sports. The primary focus has been on spread and moneyline bets, with no success in predicting the total score in any major sports. In total there have been 10,500+ bets wagered with the value of each bet being between 0.5% and 1.0% of available units. The system started with 150 units and accumulated over 11,300 units over the 20–month period from January 2021 through November 2022. As expected in a highly volatile, real world use case, there is variability in the results over shorter periods of time, but over any 60+ day period of time the results are strictly increasing.

On October 1st, 2021 the prediction method switched to the meta–swarm mode instead of relying only on predictions made by WoC–Bots. Other sources of information included in the calculation are described in Section 7.2.1. At all times, predictions were allowed to be ignored in extraordinary cases, such as late–breaking injury news or unexpected local events, e.g. city–wide protests in some event location. Less than 3% of predictions were ignored. When predictions were ignored, no action was taken on the event. Other news and information was assumed to be accounted for in the betting line offered, injury reports, and team roster information for the event.

Figure 7.5 shows the total units returned by this betting system over this time period. Note that around June 1st, 2021 the system did briefly go negative. Over this period of the year the only major sports being played are MLB and soccer, both sports that have proven to be more volatile and more difficult for this method to predict. Additional volatility was suspected to come from data inconsistencies. All major sports in the preceding year disallowed fans, had shortened seasons, or were playing condensed seasons to account for lost time due to the COVID–19 pandemic. This impacted sports and teams differently, with a larger impact seen in cardio–intensive sports such as soccer, and on teams with older players as compared with the league average. In all sports, except the NHL, the home field advantage enjoyed prior to the COVID–19 pandemic was reduced.

Tables 7.3 and 7.4 show the total units wagered, units returned beyond the units wagered, the return on investment (ROI), the total bets, total winning bets, and the win rate for WoC–Bots (January 1st, 2021 through September 30th, 2021) and the meta–swarm (October 1st, 2021 through November 15th, 2022). In sports betting the ROI is dictated not just by a win or loss, but also by the odds offered by the sportsbook. Both WoC–Bots and the meta–swarm have tended to prefer underdog bets, allowing for a lower win rate while maintaining a positive ROI.

The win rate in both tables can be thought of as the accuracy of the predictions. Notice that the win rate is below 50% for spread and moneyline bets using both WoC–Bots and the meta–swarm. The default spread line offered by a sportsbook will be set to -110 odds, but most sportsbooks allow bettors to pick their own spread line with different odds. For example, a default line could be `team-xyz` (-5.5) for -110 odds, meaning that `team-xyz` needs to win by 6 points or more to win the bet. The sportsbooks will also offer options such as (-6.5) for +105 odds or (-7.5) for +115 odds. When the prediction has a high associated confidence value we can take a custom line with positive odds on a spread bet, allowing for a win rate below 50% to still be profitable. This can also be seen for moneyline bets; the win rate for moneyline bets using

Table 7.3: WoC Bets: Risk, Return and Win Rate (Accuracy) for Spread and Moneyline Bets.

Values	Spread Bets	Moneyline Bets
Units Risked	8212	18532
Units Returned	628.5	668
ROI	9.5%	3.6%
Total Bets	1161	3282
Winning Bets	572	1481
Win Rate	49.06%	45.12%

Table 7.4: Meta-Swarm: Risk, Return and Win Rate (Accuracy) for Spread and Moneyline Bets.

Values	Spread Bets	Moneyline Bets
Units Risked	48159	80408
Units Returned	4863.5	5364.5
ROI	10.1%	6.7%
Total Bets	2376	3225
Winning Bets	1174	1296
Win Rate	49.33%	40.19%

the meta–swarm actually drops, from 45.15% using WoC-Bots only, to 40.19%. However, the ROI increases from 3.6% using WoC-Bots to 6.7%. This method became better at selecting underdog winners, increasing the ROI despite the decrease in win rate.

Some testing has been done to improve win rate by biasing the wager selection to prefer favorites. Wagers on favorites, due to the lower payout, were typically placed only when there was an associated high confidence rating in the prediction. In order to select more wagers on favorites, medium confidence predictions for favorites were also wagered on. This testing improved the win rate but also showed poor overall results, with negative ROI despite the increase in win rate. The only exception to this is the last 15-20% of both the NBA and MLB seasons where many teams know their playoff positions and the favorites tend to win more frequently than during the early and middle parts of the seasons. This is, however, reflected with worse odds on favorites, which requires a significantly higher win rate to maintain a positive ROI.

Figure 7.6 shows the total units returned for the 2021/2022 NCAA basketball season. The NCAA basketball season goes through three distinct phases. Each conference has slightly different start dates for each phase, but all are within a week of each other. The distinct phases are shown on the figure with dividing lines representing the end of each phase. Phase 1 involves teams playing non–league games against teams they do not often face. These games are often in a tournament format with less rest between games compared with phase 2. The method correctly identified that underdogs tend to win these match-ups and the predictions reflected this. Phase 2, starting around the end of December, involves league play, where teams play other

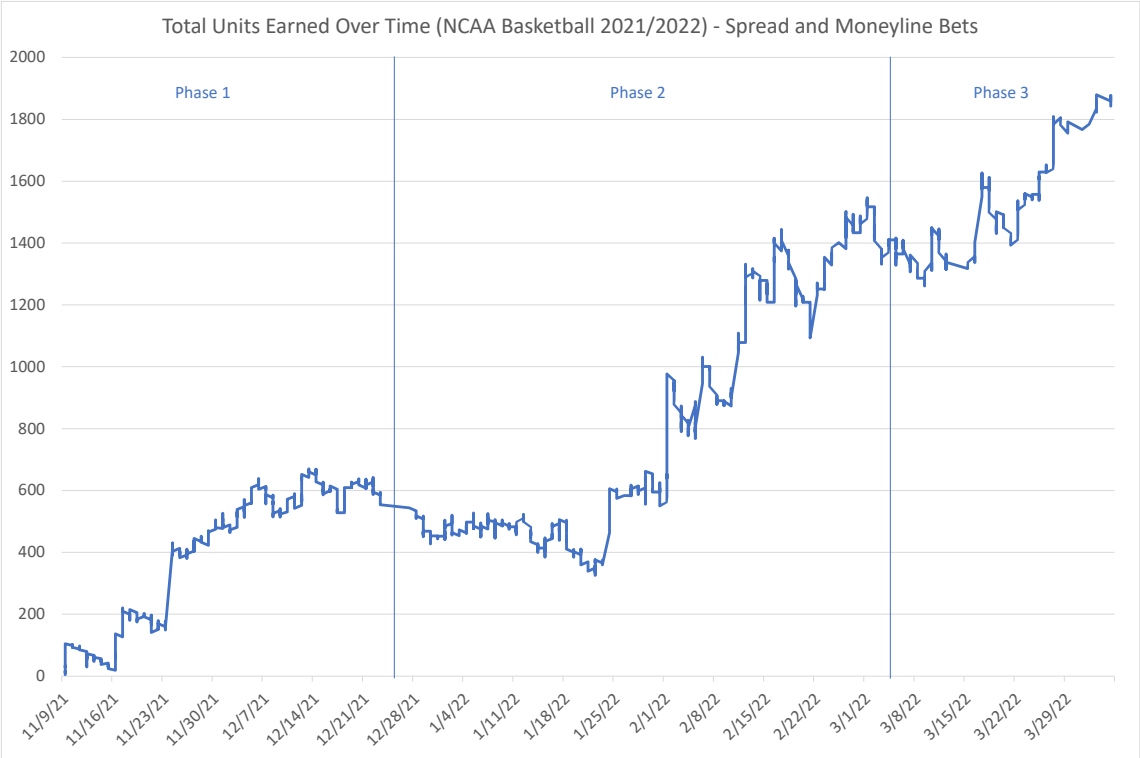

Figure 7.6: Total units earned over time for NCAA basketball during the 2021/2022 season.

teams they are familiar with. This method had difficulty with this transition, losing units from late December through late January before correcting itself before the end of this phase, through February. Phase 3 of the season involves league championships and "March Madness", where the best teams compete against each other. The method continued to earn units throughout this time, but with more volatility than other phases. As more data becomes available the expectation is that volatility will be reduced during these transition periods. However, college sports in the United States currently allow players to more easily transfer between universities, which had not been the case in the past, reducing the value of historical training data. There is not currently enough data to determine if a star player transferring from a smaller team will continue to be a star on a bigger team, or how team dynamics will change after the transfer.

Figures 7.7 and 7.8 show return for NFL wagers over the full 2021-2022 season and the first 2/3rds of the 2022-2023 season. While the 2022-2023 was on–going at the time of this analysis, the figure is included to show consistency across multiple years. The NFL is more episodic than other sports, with the majority of events occurring on Sundays, with single games played on Monday and Thursday, and occasionally on Saturdays late in the season. This is reflected in the figures with bigger jumps and drops on specific days instead of a more continuous curve. Performance was similar over both seasons, with a similar ROI for both

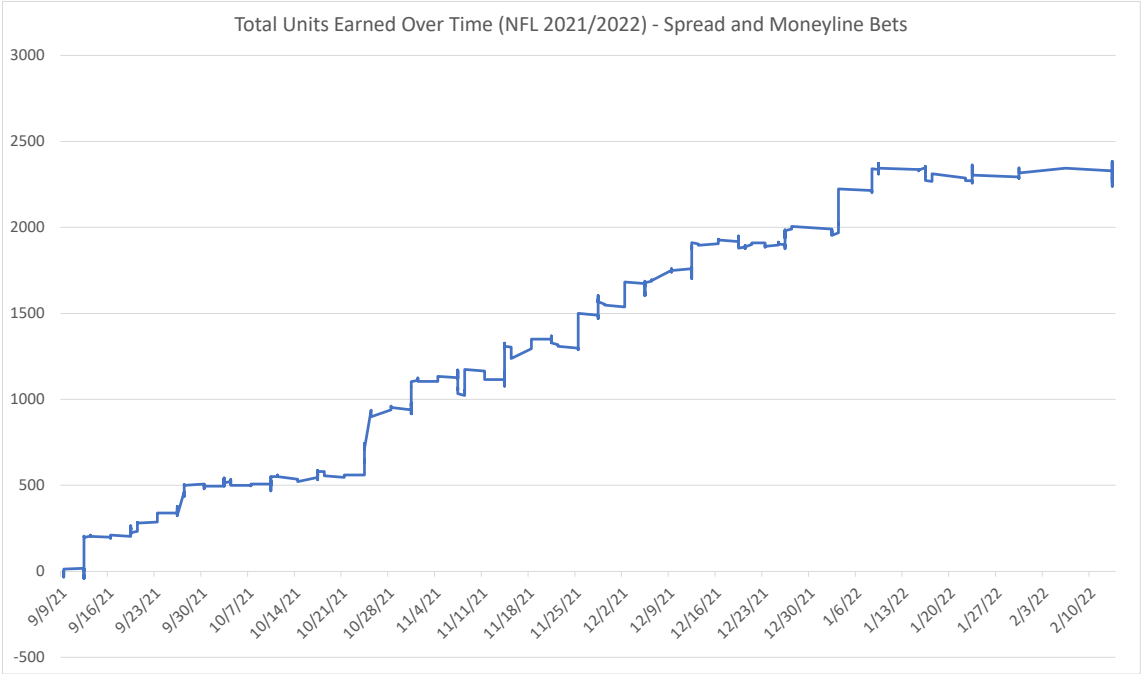

Figure 7.7: Total units earned over time for the NFL during the 2021/2022 season.

spread and moneyline bets.

7.4 Discussion

The WoC–Bots classification approach has demonstrated competitive performance with other state–of–the–art classification methods for multiple different datasets with a varied number of samples and input features. While random forest and a deep neural network implementation out–performed WoC–Bots in maximum accuracy on the tested datasets, WoC–Bots out–performed AdaBoost and logistic regression methods in two of the three tested datasets and performed similarly to all of the top methods in the third dataset. WoC–Bots' accuracy, precision, and recall metrics tracked similarly with the other classification methods; XGBoost, AdaBoost, random forest, DNN, and WoC–Bots all showed higher recall scores and lower precision scores on the breast cancer dataset. Similarly, the same methods showed higher precision and lower recall scores for the Hollywood dataset, indicating WoC–Bots are learning similar information as the other classification methods. While WoC–Bots prediction runtime is the slowest of the methods shown in this comparison, runtime optimization has not yet been a focus. This is something that can and should be improved in future work; for example, the interaction arena can be optimized to a series of matrix multiplication operations for each pair of interacting agents. Further, this optimization step would also reduce the

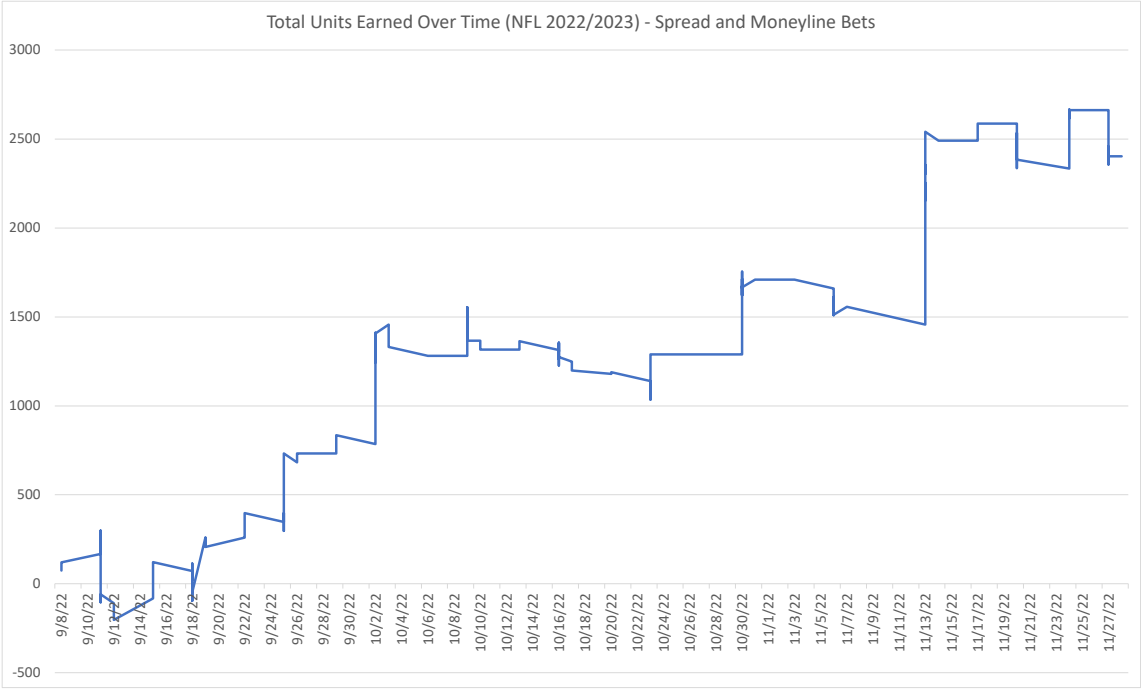

Figure 7.8: Total units earned over time for the NFL during the 2021/2022 season.

runtime during the swarming phase. The current implementation allows for easy behavioral modification while testing a new approach, but reducing the inference time is an intended future optimization.

As discussed in Chapter 5, WoC–Bots allow for incrementally expanding the input to the classification analysis which is useful when the final set of features is unknown or constantly expanding. The real world results shown with sports betting made use of this feature frequently to incorporate new data from recent games, as new data sources were discovered or new metrics were developed to track player and team performance. Individual WoC–Bots were quickly updated to include results from the most recent games and new bots were generated to represent novel data sources or statistical metrics. An additional use of the distributed meta–swarm discussed in Chapter 6 is the ability to include external prediction sources into the overall prediction, after the WoC–Bots have made their initial prediction. We have shown this reduces the overall variance in accuracy, precision, and recall in all of the datasets tested, while also producing the best accuracy of all methods tested on the Hollywood dataset. WoC–Bots, primarily using the meta–swarm extension, have been applied to a real world problem, making predictions about the outcomes of sporting events.

From January 1st, 2021 through November 15th, 2022 WoC–Bots and the meta–swarm have produced positive results across most major sports on spread and moneyline bets. Analysis of the external sources used in the meta–swarm portion show only FiveThirtyEight to be profitable for soccer and NFL betting over the

same time period. Some "betting systems" within The Action Network have shown historic profitability but are no longer profitable. More recently developed systems require an expensive monthly subscription and were not investigated, though it is reasonable to expect that at least some of these systems are also profitable on current sporting events. However, most major sports betting prediction systems, including those from within the sporting industry (ESPN, Yahoo! Sports), are not profitable, showing WoC–Bots perform quite well in a very challenging environment.

Chapter 8. Summary & Future Directions

I have demonstrated a novel, multi–agent, binary classification method based on Wisdom–of–Crowds, WoC–Bots, which use a honeybee–derived swarm mechanism for opinion aggregation. This approach is competitive with other, state–of–the–art, classification methods while demonstrating multiple, significant advantages over existing classification methods. WoC–Bots are distributable across multiple hardware nodes, can incrementally include new features without retraining existing agents, and the opinion aggregation mechanism is flexible and competent, effectively aggregating WoC–Bot predictions, and synthesizing predictions from external sources. External predictions can be incorporated into agents without revealing input data, or the methods from which the predictions were generated, preserving the privacy of any source data while allowing for diversity in classification method. WoC–Bots are ideally suited for use in environments where source data cannot be shared externally, and where multiple institutions are working together on classification tasks, using different and unique classifications methods locally. WoC–Bots are, for example, well–suited for classification tasks using private medical data with patients distributed across multiple different medical institutions.

Swarm–based decision making has shown remarkable performance in nature and the application of the honeybee–derived mechanism used in this work has shown itself to be a competent aggregation mechanism. This method, additionally, can be used to generate confidence values for each prediction, aiding in decision making and enabling stratification of samples into different confidence categories, significantly improving the average accuracy, precision, and recall for a subset of samples, as seen in Chapter 4.

8.1 Chemotaxis–based self–organization

In Chapter 2 I present a chemotaxis–inspired architecture for self–organizing primitives based on virtual cells which emit chemicals into the environment, detectable by neighboring cells. Primitives start with an initial random configuration and stochastically follow the explicit, mathematical cumulative concentration field to reliably form user–specified shapes using local only interactions. By enforcing a constraint on a single statistical moment when generating the initial swarm of primitives, it is possible to consistently generate specific shape outcomes, effectively directing global coordination through local only interactions. These types of systems, with self–directed, local only interactions are robust and adaptable, allowing for simple primi-

tives to perform complex work in environments where fault tolerance is vital and global control is limited or impossible.

8.1.1 Future Work

This work generates initial conditions which bias the primitives locations based on one of four statistical moments, based on processing and investigating unbiased simulation results. The unbiased simulations require significant manual sorting and identification of resulting shapes. This can be improved using a computer vision approach to identify and sort resulting shapes. Additional work should investigate if another method to organize agents prior to the start of the simulation allows for uniformly random agents to move themselves into biased initial conditions, using chemical concentration fields, before switching to a different chemical field which directs the agents to robustly form macroscopic shapes.

8.2 Wisdom of Crowd Bots

Chapter 3 introduces the WoC–Bots architecture and social interaction environment and algorithms. WoC–Bots are based in theories from Wisdom–of–Crowds, where the opinion of a large and diverse group will typically be more accurate than any individual opinion within the group. WoC–Bots are small and simple agents, each initialized with a subset of knowledge learned through training a small and shallow multi–layer perceptron network. Each agent then joins other agents in a social interaction arena, sharing information about their current prediction, what features they were trained on, and their historical performance data. The social interaction process allows for global information sharing and prediction formation through local only interactions.

8.2.1 Future Work

The existing architecture is modular, allowing for different algorithms to be used for initialization and movement within the interaction arena, different shapes for the arena, and behavioral modification of agents such that they are more or less willing to share information during interactions. Existing work uses a simple and random initialization of agents within the arena; future work should explore predictive performance when biasing agents to be initialized in specific locations of the arena based on the information each agent is trained on, or past performance of specific agents. Ensuring that high performing agents are uniformly distributed within the arena may produce better predictive performance through sharing more accurate information. Additional work should investigate the movement mechanism and environment shape to deter-

mine if restricting movement, or clustering agents in specific locations has a positive impact on predictive performance. Finally, modifying agent behavior may allow for a richer interaction experience which better models behavior of animals and humans which, often, are not incentivized to provide truthful and helpful information to their peers.

An existing limitation of this architecture relates to the types of input features. An input feature which shows no correlation to the outcome *unless* it is seen with at least one other input feature may not show any predictive value using this approach. Due to the random feature distribution and random social interaction it is not possible to guarantee any set of n features will be seen by the same agent. When knowledge exists about a given feature space it is possible to direct specific features to specific agents, however this is not a general solution to all problems. Experimentation using bacteria DNA to predict the dosage of antibiotic required to neutralize the bacteria presented significant difficulty, with results typically no better than the null model. Specifically, the task was to identify the dosage of antibiotic based on n–mers, or chunks of DNA codons, sized as 3–mers and 4–mers. No individual n–mer contains enough information to accurately make a classification. The random distribution of features across agents and random information dispersal during the agent interaction period made it impossible to guarantee specific sets of n–mers would be known by any specific agent. Without knowing some specific set of n–mers existed in the data it was impossible to consistently classify the amount of antibiotic necessary to neutralize the bacteria. Future work should investigate if this problem is solvable using this approach, and how to best initialize the agents to guarantee that all features are available to agents during the interaction period. This would involve modifying the initialization and movement algorithms to guarantee all agents interact at least once without adding significant compute overhead with random movements during the interaction period.

8.3 Swarm–based opinion aggregation

In Chapter 4 I introduced a swarm–based opinion aggregation mechanism for classifying breast cancer patients as node–negative or node–positive. This mechanism, modeled on honeybee foraging behavior, produces predictions with an associated confidence score, derived from the number of iterations required for the agents to agree on a prediction. Compared with existing aggregation methods, the swarm–based mechanism reduces predictive variance while improving classification performance. In use, this mechanism randomly selects 20% of agents to act as 'scouts', presenting their predictions to the remaining 80% of agents. An iterative swarming process iteratively lowers the decision threshold until a prediction is made with an

associated confidence score, or a weighted vote is taken from all agents – representing a low–confidence prediction. This aggregation mechanism significantly out–performed the weighted voter model, while lowering predictive variance, and successfully stratifying samples into confidence intervals with clear predictive performance differences.

8.3.1 Future Work

The distributed nature of this method causes specific limitations around the type of classification which is possible, where experimental results have shown predictive performance limitations for non–binary classification problems. Both multi–class output from the swarm, as well as an iterative approach where the swarm outputs a binary prediction for each pair of available classes have been investigated. For example, an iterative approach with three possible output classes will generate a binary prediction between class 1 and class 2, class 1 and class 3, and class 2 and class 3. If any of the binary predictions meets a defined threshold, a prediction is made, otherwise the top classes are again compared in a binary fashion. Neither the multi–class nor the iterative approach have consistently performed well for non–binary classification problems. Future work should look to address this limitation, possibly by extending the swarming process to better incorporate the iterative approach where more than two classes can be compared.

8.4 Incremental Feature Addition

Chapter 5 introduced incrementally adding new and previously unknown feature to an existing classification task without retraining existing agents. The social interaction and swarming phases of this approach allow for a variable number of participating agents, allowing for newly generated agents, representing the new input data, to be included in the classification task. New agents are created with a mix of new and previously known features; each agent has its MLP network trained, and is then inserted into subsequent interaction and swarming steps. This approach has shown it can learn new information from additional features, improving predictive performance.

8.4.1 Future Work

Incrementally incorporating new features was tested on two different datasets, with the Hollywood dataset showing some predictive performance difference when comparing incrementally including new features with fully retraining all agents on all of the available features (1.04%). The breast cancer dataset test has shown minimal predictive performance differences under the same circumstances (0.12%). Future work

should investigate why the Hollywood dataset showed larger increase when retraining all agents to determine if there are specific data characteristics which lead to larger performance differences when incrementally updating the input compared with fully retraining all agents.

8.5 Distributed Design

This classification approach has three distinct phases; (1) initial MLP training, (2) social interaction, and (3) swarm aggregation. Each of the the three phases has been distributed across multiple hardware nodes, with runtime decreasing as more nodes are available when a sufficient number of agents are present. Distributing the swarm initially showed poor performance due to high network transfer overhead and low compute usage. Work on this problem lead to the development of a meta–swarm, where predictions from each node are encapsulated into an agent, replacing the MLP belief for each agent. Using this method, runtime for the swarming phase was reduced dramatically while predictive performance remained similar on two of the three tested datasets.

8.5.1 Future Work

Future investigation should work towards reducing the network transfer time. This work uses an implicit transfer mechanism, using shared network storage. It is likely that an explicit approach using more efficient means, such as data serialization in Java, would lead to a more time–efficient network operation. Additional work needs to be completed to determine why there was a predictive performance drop for the third dataset, predicting Hollywood movie success, when using the meta–swarm approach. Specifically, it should be determined if there are limitations on the types of input features, the size of the dataset, or the type of prediction being made where the meta–swarm produces inferior results to that of single–node swarming.

8.6 Method Comparison & Application

Chapter 7 presents detailed comparisons with existing state–of–the–art classification methods encompassing deep learning, ensemble learning, and statistic modeling. Testing was conducted using three different datasets, described throughout this dissertation, and results showed comparable performance with all tested methods, with WoC–Bots out–performing AdaBoost and logistic regression, while being slightly out–performed by a deep neural network implementation and an ensemble method: random forest.

Also presented was a real world application of this approach, testing predictive performance on sports betting across major sports, on spread and moneyline wagers. This method was tested over a 20–month

period, showing a consistent ROI on all sports, starting with 150 units and ending with 11,300 units. The meta–swarm developed for distributing the swarm phase of the system was further extended to incorporate external predictions into the system prior to the interaction and swarm phases, boosting spread bet ROI from 9.5% (WoC–Bots only) to 10.1% and moneyline ROI from 3.6% (WoC–Bots only) to 6.7%.

8.6.1 Future Work

WoC–Bots demonstrated the slowest inference time of the tested methods; runtime optimization should become a focus of future optimization. Agents in both the interaction and swarming phases move throughout a simulated arena. Both environments can be optimized to a series of mathematical matrix operations on the attributes each agent stores internally. Movement within the interaction arena can also be pre–computed, reducing the time it takes during inference to move agents to new locations, a step which currently happens sequentially on a single compute thread.

Significantly more work should be investigated for a sports betting application of this approach. The most successful approach has been picking underdog winners, for a lower average win rate but a higher return on investment. Finding the right balance between win rate and ROI, and biasing the swarm mechanism to produce appropriate confidence scores to pick the right wagers is an interesting optimization problem I plan on continuing to investigate. Further investigation should build upon future work in developing a multi–class prediction system to better take advantage of sports with non–binary outcomes.

8.7 Conclusion

I presented a novel approach to binary classification developed around a modular, multi–agent–based architecture. This approach allows for classification of samples with incomplete data which demonstrates robust predictive performance. This method allows for incrementally updating input data with new, previously unknown features, without retraining existing agents. Each of the three phases, training, social interaction, and swarm aggregation, have been effectively distributed across multiple compute nodes, decreasing training and inference time. Importantly, an additional algorithm was developed, the meta–swarm, which allows for predictions from distributed sources to be included in the classification task, without compromising the privacy of the original input data, while also allowing for flexibility in classification method from each input source. This dissertation demonstrated fundamental contributions to the field of computer science and successfully demonstrated a real world usage of this method.

8.8 Publications & Talks

8.8.1 Publications

1. S. Grimes, L. Bai, A.W.E. McDonald, and D.E. Breen, "Directing Chemotaxis-Based Spatial Self-Organization via Biased, Random Initial Conditions," International Journal of Parallel, Emergent and Distributed Systems, Vol. 34, No. 4, April 2019, pp. 380-399.

2. S. Grimes and D.E. Breen, "Woc-Bots: An Agent-Based Approach to Decision-Making," Applied Sciences, Vol. 9, No. 21, November 2019, p. 4653.

3. S. Grimes, M.D. Zarella, F.U. Garcia and D.E. Breen, "An Agent-based Approach to Predicting Lymph Node Metastasis Status in Breast Cancer," Proc. IEEE International Conference on Bioinformatics and Biomedicine, pp. 1315-1319, December 2021.

4. S. Grimes and D.E. Breen, "A Multi–Agent Approach to Binary Classification Using Swarm Intelligence," Future Internet, Vol 15, No. 1, January 2023, p. 36.

8.8.2 Talks

1. "WoC-bots: A Multi-agent Approach to Predicting Lymph Node Metastasis from Primary Breast Tumors", presented at Pathology Informatics Summit, May 2021

2. "An Agent-based Approach to Predicting Lymph Node Metastasis Status in Breast Cancer," presented at IEEE International Conference on Bioinformatics and Biomedicine, December 2021

Bibliography

[1] Hervé Abdi and Lynne J Williams. "Principal component analysis". *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), pp. 433–459.

[2] Esther Abels et al. "Computational pathology definitions, best practices, and recommendations for regulatory guidance: a white paper from the Digital Pathology Association". *The Journal of pathology* 249.3 (2019), pp. 286–294.

[3] Saeed Hamood Alsamhi et al. "Blockchain-empowered security and energy efficiency of drone swarm consensus for environment exploration". *IEEE Transactions on Green Communications and Networking* 7.1 (2022), pp. 328–338.

[4] Shun-ichi Amari. "Backpropagation and stochastic gradient descent method". *Neurocomputing* 5.4-5 (1993), pp. 185–196.

[5] Laura Auria and Rouslan A Moro. "Support vector machines (SVM) as a technique for solvency analysis". 811 (2008).

[6] Devanshu Awasthi. *Exploratory Analysis of Movies on TMDB*. https://rstudio-pubs-static.s3.amazonaws.com/369891_b123051c3cb64da5a6d22a8d0b6e0d84.html. Accessed: 2019-10-19.

[7] L. Bai. "Chemotaxis-based Spatial Self-Organization Algorithms". PhD thesis. Philadelphia, PA: Drexel University, 2014.

[8] L. Bai. "Self-Organizing Primitives for Automated 2D Shape Composition". MA thesis. Philadelphia, PA: Drexel University, 2008.

[9] L. Bai and D.E. Breen. "Chemotaxis-inspired cellular primitives for self-organizing shape formation". In: *Morphogenetic Engineering: Toward Programmable Complex Systems*. Springer, 2012. Chap. 9, pp. 209–237.

[10] L. Bai, M. Eyiyurekli, and D. Breen. "An Emergent System for Self-Aligning and Self-Organizing Shape Primitives". *Proc. IEEE International Conference on Self-Adaptive and Self-Organizing Systems Conference*. 2008, pp. 445–454.

[11] L. Bai, M. Eyiyurekli, and D. Breen. "Automated Shape Composition Based on Cell Biology and Distributed Genetic Programming". *Proc. Genetic and Evolutionary Computation Conference*. 2008, pp. 1179–1186.

[12] L. Bai, R. Gilmore, and D.E. Breen. "Predicting Spatial Self-Organization with Statistical Moments". *Proc. Spatial Computing Workshop of the AAMAS Conference*. 2014, Article 2.

[13] L. Bai et al. "Self-Organized Sorting of Heterotypic Agents Via a Chemotaxis Paradigm". *Science of Computer Programming* 78.5 (2013), pp. 594–611.

[14] Emmanuel Barranger et al. "An axilla scoring system to predict non-sentinel lymph node status in breast cancer patients with sentinel lymph node involvement". *Breast cancer research and treatment* 91 (2005), pp. 113–119.

[15] Matthias Becker. "Swarm learning for decentralized healthcare". *Der Hautarzt* 73.4 (2022), pp. 323–325.

[16] Madeleine Beekman and FLW Ratnieks. "Long-range foraging by the honey-bee, Apis mellifera L." *Functional Ecology* 14.4 (2000), pp. 490–496.

[17] Joyce E Berg, Forrest D Nelson, and Thomas A Rietz. "Prediction market accuracy in the long run". *International Journal of Forecasting* 24.2 (2008), pp. 285–300.

[18] Joyce E Berg and Thomas A Rietz. "Prediction markets as decision support systems". *Information systems frontiers* 5 (2003), pp. 79–93.

[19] Gérard Biau and Erwan Scornet. "A random forest guided tour". *Test* 25.2 (2016), pp. 197–227.

[20] Jacobus C Biesmeijer and Thomas D Seeley. "The use of waggle dance information by honey bees throughout their foraging careers". *Behavioral Ecology and Sociobiology* 59 (2005), pp. 133–142.

[21] E. Bonabeau et al. "Three-dimensional architectures grown by simple 'stigmergic' agents". *BioSystems* 56 (2000), pp. 13–32.

[22] Jair Cervantes et al. "A comprehensive survey on support vector machine classification: Applications, challenges and trends". *Neurocomputing* 408 (2020), pp. 189–215.

[23] Tianqi Chen et al. "Xgboost: extreme gradient boosting". *R package version 0.4-2* 1.4 (2015), pp. 1–4.

[24] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

[25] J.A. Davies. *Mechanisms of Morphogenesis: The Creation of Biological Form*. Amsterdam: Elsevier, 2005.

[26] Arthur De Vany. *Hollywood economics: How extreme uncertainty shapes the film industry*. Routledge, 2003.

[27] Rijul Dhir and Anand Raj. "Movie success prediction using machine learning algorithms and their comparison". *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*. IEEE. 2018, pp. 385–390.

[28] R. Doursat. "Organically grown architectures: Creating decentralized, autonomous systems by embryomorphic engineering". In: *Organic Computing*. Ed. by R.P. Würtz. Springer Verlag, 2008. Chap. 8, pp. 167–200.

[29] R. Doursat. "The Growing Canvas of Biological Development: Multiscale Pattern Generation on an Expanding Lattice of Gene Regulatory Nets". In: *Unifying Themes in Complex Systems*. Springer Verlag, 2008, pp. 205–210.

[30] Stephan Dreiseitl and Lucila Ohno-Machado. "Logistic regression and artificial neural network classification models: a methodology review". *Journal of biomedical informatics* 35.5-6 (2002), pp. 352–359.

[31] Qianzhou Du et al. "CrowdIQ: A New Opinion Aggregation Model". *Proc. 50th Hawaii International Conference on System Sciences*. 2017, pp. 535–542.

[32] P. Eggenberger. "Evolving Morphologies of Simulated 3D Organisms Based on Differential Gene Expression". *Proc. 4th European Conference on Artificial Life*. 1997, pp. 205–213.

[33] E. Eisenbach et al. *Chemotaxis*. London: Imperial College Press, 2004.

[34] Şeyda Ertekin, Cynthia Rudin, and Haym Hirsh. "Approximating the crowd". *Data mining and knowledge discovery* 28.5-6 (2014), pp. 1189–1221.

[35] M. Eyiyurekli et al. "A Computational Model of Chemotaxis-based Cell Aggregation". *BioSystems* 93.3 (Sept. 2008), pp. 226–239.

[36] Maris Fessenden. *How 'Waves' Rippling Through Bird Flocks Help Them Escape Predators*. Accessed on March 3rd, 2023. Mar. 2015.

[37] K.W. Fleischer. "Investigations with a Multicellular Developmental Model". *Proc. Artificial Life V*. 1996, pp. 389–408.

[38] K.W. Fleischer and A.H. Barr. "A Simulation Testbed for the Study of Multicellular Development: The Multiple Mechanisms of Morphogenesis". *Proc. Artificial Life III*. 1994, pp. 389–408.

[39] Mudasir A Ganaie et al. "Ensemble deep learning: A review". *Engineering Applications of Artificial Intelligence* 115 (2022), p. 105151.

[40] Philip Garnett and John Bissell. "Modelling social networks reveals how information spreads" (2013). http://theconversation.com/modelling-social-networks-reveals-how-information-spreads-18776.

[41] Victor Gerling and Sebastian Von Mammen. "Robotics for Self-Organised Construction". *Foundations and Applications of Self* Systems, IEEE International Workshop on*. 2016, pp. 162–167.

[42] S.F. Gilbert. *Developmental Biology*. 10th. Sunderland, MA: Sinauer Associates, Inc., 2013.

[43] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[44] Sean Grimes and David E Breen. "Woc-bots: An agent-based approach to decision-making". *Applied Sciences* 9.21 (2019), p. 4653.

[45] Sean Grimes et al. "Directing chemotaxis-based spatial self-organisation via biased, random initial conditions". *International Journal of Parallel, Emergent and Distributed Systems* 34.4 (2019), pp. 380–399.

[46] Jialiang Han et al. "Demystifying swarm learning: A new paradigm of blockchain-based decentralized federated learning". *arXiv preprint arXiv:2201.05286* (2022).

[47] Jochen Hartmann et al. "Comparing automated text classification methods". *International Journal of Research in Marketing* 36.1 (2019), pp. 20–38.

[48] Mahdi Hashemi. "Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation". *Journal of Big Data* 6.1 (2019), pp. 1–13.

[49] Reid Hastie and Tatsuya Kameda. "The robust beauty of majority rules in group decisions." *Psychological Review* 112.2 (2005), p. 494.

[50] Michael G Hinchey, Roy Sterritt, and Chris Rouff. "Swarms and swarm intelligence". *Computer* 40.4 (2007), pp. 111–113.

[51] P.E. Hotz. "Combining developmental processes and their physics in an artificial evolutionary system to evolve shapes". In: *On Growth, Form and Computers*. Ed. by Kumar S. and P.J. Bentley. Academic Press, 2003, pp. 302–318.

[52] M.A. Hsieh and V. Kumar. "Pattern generation with multiple robots". *Proc. IEEE International Conference on Robotics and Automation*. Orlando, Florida, 2006, pp. 2442–2447.

[53] *Swarm Intelligence*. http://www.esa.int/gsp/ACT/ai/projects/swarm.html. Retrieved October 23, 2017. 2017.

[54] A. Jadbabaie, J. Lin, and A.S. Morse. "Coordination of groups of mobile autonomous agents using nearest neighbor rules". *IEEE Trans. on Automatic Control* 48.6 (2003), pp. 988–1001.

[55] David S Khoury, Mary R Myerscough, and Andrew B Barron. "A quantitative model of honey bee colony population dynamics". *PloS one* 6.4 (2011), e18491.

[56] Eunji Kim et al. "Fault detection and diagnosis using self-attentive convolutional neural networks for variable-length sensor data in semiconductor manufacturing". *IEEE Transactions on Semiconductor Manufacturing* 32.3 (2019), pp. 302–309.

[57] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980* (2014).

[58] Holbrook E Kohrt et al. "New models and online calculator for predicting non-sentinel lymph node status in sentinel lymph node positive breast cancer patients". *BMC cancer* 8.1 (2008), pp. 1–15.

[59] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.

[60] Anders Krogh. "What are artificial neural networks?" *Nature biotechnology* 26.2 (2008), pp. 195–197.

[61] HMC Bee Lab. *The Waggle Dance*. Accessed on March 3rd, 2023. July 2014.

[62] Jae Won Lee et al. "An extensive comparison of recent classification tools applied to microarray data". *Computational Statistics & Data Analysis* 48.4 (2005), pp. 869–885.

[63] Yan-Wei Lee et al. "Axillary lymph node metastasis status prediction of early-stage breast cancer using convolutional neural networks". *Computers in Biology and Medicine* 130 (2021), p. 104206.

[64] Guoqiang Li, Peifeng Niu, and Xingjun Xiao. "Development and investigation of efficient artificial bee colony algorithm for numerical function optimization". *Applied soft computing* 12.1 (2012), pp. 320–332.

[65] Li Li et al. "A review of applications in federated learning". *Computers & Industrial Engineering* 149 (2020), p. 106854.

[66] Shu-Hsien Liao and Chih-Hao Wen. "Artificial neural networks classification and clustering of methodologies and applications–literature analysis from 1995 to 2005". *Expert Systems with applications* 32.1 (2007), pp. 1–11.

[67] Gen-Min Lin et al. "Machine learning based suicide ideation prediction for military personnel". *IEEE journal of biomedical and health informatics* 24.7 (2020), pp. 1907–1916.

[68] Adam Lipowski and Dorota Lipowska. "Roulette-wheel selection via stochastic acceptance". *Physica A: Statistical Mechanics and its Applications* 391.6 (2012), pp. 2193–2196.

[69] Ryan Lowe et al. "Multi-agent actor-critic for mixed cooperative-competitive environments". *arXiv preprint arXiv:1706.02275* (2017).

[70] Ibrahim Malkoc. *The Movies Dataset*. http://ibomalkoc.com/movies-dataset/. Accessed: 2019-08-23.

[71] M. Mamei, M. Vasirani, and F. Zambonelli. "Experiments of morphogenesis in swarms of simple mobile robots". *Applied Artificial Intelligence* 18.9 (2004), pp. 903–919.

[72] Seyedali Mirjalili. "Genetic algorithm". *Evolutionary Algorithms and Neural Networks: Theory and Applications* (2019), pp. 43–55.

[73] R. Nagpal. "Programmable self-assembly using biologically-inspired multiagent control". *Proc. 1st International Joint Conference on Autonomous Agents and Multiagent Systems: part 1*. 2002, pp. 418–425.

[74] R. Nagpal, A. Kondacs, and C. Chang. "Programming Methodology for Biologically-Inspired Self-Assembling Systems". *Proc. AAAI Spring Symposium on Computational Synthesis*. 2003, pp. 173–180.

[75] Clement Nartey et al. "Blockchain-IoT peer device storage optimization using an advanced time-variant multi-objective particle swarm optimization algorithm". *EURASIP Journal on Wireless Communications and Networking* 2022.1 (2022), pp. 1–27.

[76] Brett Naul et al. "A recurrent neural network for classification of unevenly sampled variable stars". *Nature Astronomy* 2.2 (2018), pp. 151–155.

[77] Radford M Neal. "Slice sampling". *Annals of Statistics* (2003), pp. 705–741.

[78] William S Noble. "What is a support vector machine?" *Nature biotechnology* 24.12 (2006), pp. 1565–1567.

[79] Emilio Soria Olivas. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques: Algorithms, Methods, and Techniques*. IGI Global, 2009.

[80] Abraham Othman. "Zero-intelligence agents in prediction markets". *Proc. 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Volume 2*. 2008, pp. 879–886.

[81] Abraham Othman and Tuomas Sandholm. "When do markets with simple agents fail?" *Proc. 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1*. 2010, pp. 865–872.

[82] Scott E Page. *The Difference: How the Power of Diversity Creates Better Groups, Firms, Schools, and Societies-New Edition*. Princeton University Press, 2008.

[83] Maria Palombini et al. "Opportunities and Challenges of Swarm AI for Decentralized Clinical Research Panel Session". *Blockchain in Healthcare Today* 5.Multimedia Special Issue (2022).

[84] Bijaya Ketan Panigrahi, Yuhui Shi, and Meng-Hiot Lim. *Handbook of swarm intelligence: concepts, principles and applications*. Vol. 8. Springer Science & Business Media, 2011.

[85] David C. Parkes and Sven Seuken. "Prediction Markets". In: *Introduction to Economics and Computation: A Design Approach*. Cambridge University Press, 2016. Chap. 18.

[86] L. Pettazzi, D. Izzo, and S. Theil. "Swarm navigation and reconfiguration using electrostatic forces". *Proc. 7th International Conference On Dynamics and Control of Systems and Structures in Space*. 2006, pp. 257–268.

[87] Rolf Pfeifer, Max Lungarella, and Fumiya Iida. "Self-organization, embodiment, and biologically inspired robotics". *Science* 318.5853 (2007), pp. 1088–1093.

[88] M. Prokopenko, ed. *Guided Self-Organization: Inception*. Berlin: Springer, 2014.

[89] Priya Ranganathan, CS Pramesh, and Rakesh Aggarwal. "Common pitfalls in statistical analysis: logistic regression". *Perspectives in clinical research* 8.3 (2017), p. 148.

[90] David Reby et al. "Artificial neural networks as a classification method in the behavioural sciences". *Behavioural processes* 40.1 (1997), pp. 35–43.

[91] C.W. Reynolds. "Flocks, herds and schools: A distributed behavioral model". *Proc. SIGGRAPH '87*. 1987, pp. 25–34.

[92] Nicola Rieke et al. "The future of digital health with federated learning". *NPJ digital medicine* 3.1 (2020), p. 119.

[93] E. Rimon and D.E. Koditschek. "Exact Robot Navigation Using Artificial Potential Functions". *IEEE Trans. on Robotics and Automation* 8.5 (1992), pp. 501–518.

[94] Louis Rosenberg. "Artificial Swarm Intelligence, a Human-in-the-loop approach to AI". *Proc. 13th AAAI Conference on Artificial Intelligence*. 2016.

[95] Louis Rosenberg, Niccolo Pescetelli, and Gregg Willcox. "Artificial Swarm Intelligence amplifies accuracy when predicting financial markets". *Proc. IEEE 8th Annual Conference on Ubiquitous Computing, Electronics and Mobile Communication*. 2017, pp. 58–62.

[96] Omer Sagi and Lior Rokach. "Ensemble learning: A survey". *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.4 (2018), e1249.

[97] Oliver Lester Saldanha et al. "Swarm learning for decentralized artificial intelligence in cancer histopathology". *Nature Medicine* 28.6 (2022), pp. 1232–1239.

[98] H. Sayama. "Robust morphogenesis of robotic swarms". *IEEE Computational Intelligence* 5.3 (2010), pp. 43–49.

[99] H. Sayama. "Swarm chemistry". *Artificial Life* 15.1 (2009), pp. 105–114.

[100] Robert E Schapire. "Explaining adaboost". In: *Empirical inference*. Springer, 2013, pp. 37–52.

[101] Skipper Seabold and Josef Perktold. "Statsmodels: econometric and statistical modeling with python". *Proc. 9th Python in Science Conference*. 2010, pp. 57–61.

[102] T. Sekimura et al., eds. *Morphogenesis and Pattern Formation in Biological Systems*. Tokyo: Springer, 2003.

[103] Wei-Min Shen et al. "Hormone-inspired self-organization and distributed control of robotic swarms". *Autonomous Robots* 17.1 (2004), pp. 93–105.

[104] K. Sims. "Interactive Evolution of Dynamical Systems". *Proc. 1st European Conference on Artificial Life*. 1992, pp. 171–178.

[105] Marjolein L Smidt et al. "Can the Memorial Sloan-Kettering Cancer Center nomogram predict the likelihood of nonsentinel lymph node metastases in breast cancer patients in the Netherlands?" *Annals of surgical oncology* 12.12 (2005), pp. 1066–1072.

[106] Jaime Lynn Speiser et al. "A comparison of random forest variable selection methods for classification prediction modeling". *Expert systems with applications* 134 (2019), pp. 93–101.

[107] K. Stoy and R. Nagpal. "Self-reconfiguration using directed growth". *Proc. Internatioanl Symposium on Distributed Autonomous Robotic Systems*. 2004, pp. 3–12.

[108] Shizhao Sun et al. "On the depth of deep neural networks: A theoretical view". *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 1. 2016.

[109] J Surowiecki. *The Wisdom of the Crowds*. New York City: Anchor Books, a division of Random House, 2005.

[110] D Team et al. "Deeplearning4j: Open-source distributed deep learning for the JVM". *Apache Software Foundation License* 2 (2018).

[111] Valery Tereshko and Andreas Loengarov. "Collective decision making in honey-bee foraging dynamics". *Computing and Information Systems* 9.3 (2005), p. 1.

[112] G. Theraulaz and E. Bonabeau. "A brief history of stigmergy". *Artificial Life* 5 (1999), pp. 97–116.

[113] G. Theraulaz and E. Bonabeau. "Coordination in Distributed Building". *Nature* 269 (1995), pp. 686–688.

[114] G. Theraulaz and E. Bonabeau. "Modeling the Collective Building of Complex Architectures in Social Insects with Lattice Swarms". *Journal of Theoretical Biology* 177 (1995), pp. 381–400.

[115] Lisa Torrey and Jude Shavlik. "Transfer learning". In: *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

[116] Yaniss Touahmi et al. "Congestion avoidance for multiple micro-robots using the behaviour of fish schools". *International Journal of Advanced Robotic Systems* 9.3 (2012), p. 67.

[117] Gabriele Valentini, Heiko Hamann, and Marco Dorigo. "Self-organized collective decision making: The weighted voter model". *Proc. International Conference on Autonomous Agents and Multi-agent Systems*. 2014, pp. 45–52.

[118] I Van den Hoven et al. "Value of Memorial Sloan-Kettering Cancer Center nomogram in clinical decision making for sentinel lymph node-positive breast cancer". *Journal of British Surgery* 97.11 (2010), pp. 1653–1658.

[119] RFD Van la Parra et al. "Validation of a nomogram to predict the risk of nonsentinel lymph node metastases in breast cancer patients with a positive sentinel node biopsy: validation of the MSKCC breast nomogram". *Annals of surgical oncology* 16.5 (2009), pp. 1128–1135.

[120] Kimberly J Van Zee et al. "A nomogram for predicting the likelihood of additional nodal metastases in breast cancer patients with a positive sentinel node biopsy". *Annals of surgical oncology* 10.10 (2003), pp. 1140–1151.

[121] T. Vicsek et al. "Novel Type of Phase Transition in a System of Self-Driven Particles". *Physical Review Letters* 75.6 (1995), pp. 1226–1229.

[122] Karl Von Frisch. *The dance language and orientation of bees*. Harvard University Press, 2013.

[123] Sebastian Von Mammen and Christian Jacob. "The evolution of swarm grammars-growing trees, crafting art, and bottom-up design". *IEEE Computational Intelligence Magazine* 4.3 (2009), pp. 10–19.

[124] Justin Werfel, Kirstin Petersen, and Radhika Nagpal. "Designing Collective Behavior in a Termite-Inspired Robot Construction Team". *Science* 343.6172 (2014), pp. 754–758.

[125] S. Wiggins, ed. *Introduction to Applied Nonlinear Dynamical Systems and Chaos, 2nd Edition*. New York: Springer, 2003.

[126] David H Wolpert. "Stacked generalization". *Neural networks* 5.2 (1992), pp. 241–259.

[127] Raymond E Wright. "Logistic regression." (1995).

[128] Sheng Kung Yi et al. "The wisdom of the crowd in combinatorial problems". *Cognitive Science* 36.3 (2012), pp. 452–470.

[129] Jaehong Yoon et al. "Lifelong learning with dynamically expandable networks". *arXiv preprint arXiv:1708.01547* (2017).

[130] Jason Yosinski et al. "How transferable are features in deep neural networks?" *arXiv preprint arXiv:1411.1792* (2014).

[131] M.F. Zakaria, H.S. Choon, and S.A. Suandi. "Object shape recognition in image for machine vision application". *International Journal of Computer Theory and Engineering* 4.1 (2012), pp. 76–80.

[132] Mark D Zarella et al. "Lymph node metastasis status in breast carcinoma can be predicted via image analysis of tumor histology". *Anal Quant Cytopathol Histpathol* 37.5 (2015), pp. 273–285.

[133] Cha Zhang and Yunqian Ma. *Ensemble machine learning: methods and applications*. Springer, 2012.

[134] Chongsheng Zhang et al. "An up-to-date comparison of state-of-the-art classification algorithms". *Expert Systems with Applications* 82 (2017), pp. 128–150.

[135] Yuanyuan Zhang et al. "Adaptive convolutional neural network and its application in face recognition". *Neural Processing Letters* 43.2 (2016), pp. 389–399.

[136] Hongyu Zhu et al. "Benchmarking and analyzing deep neural network training". *2018 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE. 2018, pp. 88–100.

[137] Yan-fei Zhu and Xiong-min Tang. "Overview of swarm intelligence". *Proc. IEEE International Conference on Computer Application and System Modeling*. Vol. 9. 2010, pp. V9–400.