Winter 2014

Improving and maintaining prediction accuracy in agent based modeling systems under dynamic environment

Inderjeet Singh Dogra
University of Windsor

IMPROVING AND MAINTAINING PREDICTION ACCURACY IN AGENT BASED MODELING SYSTEMS UNDER DYNAMIC ENVIRONMENT

by

INDERJEET SINGH DOGRA

A Thesis

Submitted to the Faculty of Graduate Studies

through the School of Computer Science

in Partial Fulfillment of the Requirements for

the Degree of Master of Science at the

University of Windsor

Windsor, Ontario, Canada

2013

IMPROVING AND MAINTAINING PREDICTION ACCURACY IN AGENT BASED MODELING SYSTEMS UNDER DYNAMIC ENVIRONMENT

by

INDERJEET SINGH DOGRA

APPROVED BY:

M. Hlynka, External Reader
Department of Mathematics and Statistics

D. Wu, Internal Reader
School of Computer Science

Z. Kobti, Advisor
School of Computer Science

28 November, 2013

Declaration of Previous Publication

This thesis includes one original paper that has been previously published in peer reviewed conference proceeding, as follows:

Thesis Chapter	Full Citation	Publication Status
Chapters 3, 4 and 5	Dogra, I. S. and Kobti, Z. 2013. Improving prediction accuracy in agent based modeling systems under dynamic environment. In *IEEE Congress on Evolutionary Computation (CEC), 2013*. IEEE, 2114-2121.	*Published*

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material in my thesis. I certify that the above material describes work completed during my registration as graduate student at the University of Windsor.

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses

the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material in my thesis.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

The dynamic and complex nature of real world systems makes it difficult to build an accurate artificial simulation. Agent Based Modeling Simulations used to build such simulated models are often oversimplified and not realistic enough to predict reliable results. In addition to this, the validation of such Agent Based Model (ABM) involves great difficulties thus putting a question mark on their effective usage and acceptability. One of the major problems affecting the reliability of ABM is the dynamic nature of the environment. An ABM initially validated at a given time stamp is bound to become invalid with the inevitable change in the environment over time. Thus, an ABM not learning regularly from its environment cannot sustain its validity over a longer period of time. This thesis describes a novel approach for incorporating adaptability and learning in an ABM simulation, in order to improve and maintain its prediction accuracy under dynamic environment. In addition, it also intends to identify and study the effect of various factors on the overall progress of the ABM, based on the proposed approach.

Dedication

I would like to dedicate this thesis to my family, especially my mother, whose prayers and immense patience and faith in me have got me this far in my life. Also, I am grateful to the Almighty God for his blessings.

Acknowledgements

I would like to thank my supervisor, Dr. Ziad Kobti, for giving me the opportunity to get an exposure of the research field. It has been a great experience working under his guidance, which got me a publication, which otherwise would have been impossible. His constant motivation, support and faith guided me in successful completion of my thesis. I would also like to appreciate him for all the financial support he has given, which helped me focus on my work and took care of my expenses.

My sincere gratitude goes to Mrs. Gloria Mensah, secretary to the director, who has always helped setup meetings with my supervisor and ensured that he always meets me, despite his busy schedule. In addition, my sincere thanks to Ms. Mandy Turkalj, who has always been there to take care of other issues related to my master's degree.

Finally, I would like to thank my parents, especially my mother who has gone through all the ups and downs; I faced during my research, along with me.

Contents

Declaration of Previous Publication iii

Abstract v

Dedication vi

Acknowledgements vii

List of Tables xii

List of Figures xiii

1 Introduction 1

1.1 ABM and Validation 2

1.2 Current Research Motivation 3

1.3 Thesis Contribution 5

1.4 Thesis Outline 6

2 Literature Review 8

2.1 ABM and Data Mining 8

2.1.1 Agent Based Modeling Simulation 9
2.1.1.1 Defining Agents in ABM 10
2.1.1.2 Designing ABM 12
2.1.1.3 Applications of ABM 14
2.1.2 Data Mining 15
2.1.2.1 Data Mining Techniques 16
2.1.2.2 Applications of Data Mining 17
2.1.3 Integrating ABM and Data Mining 19
2.1.3.1 Application of Data Mining in ABM 20
2.1.3.2 Related Work 23
2.2 Data Mining and Machine Learning 26
2.2.1 Machine Learning Techniques 26
2.2.2 Measuring Similarity between Decision Trees 29
2.2.2.1 Related Work 30
2.3 Other Background Study 32

3 Solution Framework **37**

3.1 Incorporating Learning into Agents 38
3.1.1 Review of Previous Work 38
3.1.2 Proposed Approach 39
3.2 Measuring Similarity between Decision Trees 43
3.2.1 Review of Previous Work 43
3.2.2 Proposed Approach 44
3.3 Integrated Approach 55

4 Case Study **58**

4.1 Referred Case Studies 60

4.1.1 Robocup Rescue Simulation 61

4.1.2 Ozone Level Detection 62

4.1.3 Climate Data 63

4.1.4 Enron Email Dataset 64

4.2 Selected Case Study: Golf Play 64

4.2.1 Synthetic Data Generator 66

4.2.2 Realistic Modeling of Attributes 68

4.3 Experimental Setup 68

4.3.1 Case Study based ABM 69

4.3.2 Repast: The Simulation Platform 75

4.3.3 Weka: The Data Mining Software 77

4.3.4 Model Parameters 79

5 Results and Discussion **83**

5.1 Demonstrating and Validating the Proposed Approach 84

5.1.1 Effect of Memory Length on Agents' Learning Rate 85

5.1.2 Effect of SMT on Agents' Synchronization Level 90

5.1.3 Validating the proposed method 95

5.2 Studying the effects of realistic Model Parameters 97

5.2.1 Effect of Retention Rate 98

5.2.2 Effect of Acceptance Rate 99

5.2.3 Effect of Social Influence 101

6 Conclusion and Future Work **104**

References **108**

A Copyright Permission **116**

Vita Auctoris **118**

List of Tables

4.1 Golf Play Dataset 65

List of Figures

2.1 The computer simulation design process as described by Drogoul et al. (2003); Gilbert and Troitzsch (1999). 13

2.2 *"Applying data mining for the verification and validation of agent based models"* (Baqueiro et al., 2009). 21

2.3 *"Data mining revision process applied to agent-based simulation"* (Remondino and Correndo, 2005). 23

3.1 An agent in the proposed approach. 41

3.2 A plot of subspace covered by rule sets 1 and 2 in the feature/attribute space. 50

3.3 Comparing rules R_{11} and R_{22} for structural similarity. 51

3.4 Comparing rules R_{11} and R_{12} for structural similarity. 52

3.5 Comparing rules R_{21} and R_{22} for structural similarity. 53

3.6 Comparing rules R_{21} and R_{12} for structural similarity. 54

3.7 Flowchart for the working of integrated approach. 55

4.1 Synthetic Data Generator. 67

4.2 Experimental Setup. 71

4.3 Decision Tree Editor. 73

5.1 Population of agents synchronized with environment for ABM-60. 86
5.2 Average similarity measure of agents for ABM-60. 86
5.3 Population of agents synchronized with environment for ABM-170. 87
5.4 Average similarity measure of agents for ABM-170. 88
5.5 Population of agents synchronized with environment for ABM-120. 89
5.6 Average similarity measure of agents for ABM-120. 89
5.7 Population of agents synchronized with environment for ABM-60. 91
5.8 Average similarity measure of agents for ABM-60. 92
5.9 Population of agents synchronized with environment for ABM-170. 93
5.10 Average similarity measure of agents for ABM-170. 93
5.11 Population of agents synchronized with environment for ABM-120. 94
5.12 Average similarity measure of agents for ABM-120. 94
5.13 Average Prediction Accuracy for all three ABMs. 96
5.14 Average similarity measure of agents for different values of retention rate for ABM-120. 99
5.15 Average similarity measure of agents for different values of acceptance rate for ABM-120. 100
5.16 Average similarity measure of agents with and without social influence for ABM-120. 103

Chapter 1

Introduction

Agent Based Modeling Simulation is considered as a powerful methodology for simulating real world complex systems and has proved its worth in a wide variety of disciplines, such as Social Sciences (Doran, 1998); Economics (Palmer et al., 1994); Business (Robertson, 2003) and Health Care (Kobti et al., 2006). The Agent Based Model (ABM) allows researchers and domain experts to gain an insight into the working and behavior of complex systems and provides them with immense knowledge in return. This knowledge though is an approximation of the exact knowledge, yet it highly facilitates the in-depth understanding of the modeled system (Remondino and Correndo, 2005). Over the past years, ABM has been increasingly used both at the research level and at the industry level in modeling various complex frameworks, as a solution to real world problems. Among all the applications, ABM has developed interest for itself in the area of predictive modeling (Garcia, 2005). In other words, they are used to predict the behavior of real world systems for unseen situations. Some attempts towards building predictive ABM, as mentioned by Garcia (2005), includes predicting the success of a movie, predicting human behavior in a social gathering etc. However, the lack of sophisticated calibration techniques raises a

question mark on their successful predictions. This highlights that the overall acceptability and usage of an ABM highly depends on the validation and reliability of its results. Therefore, an ABM for a real world problem giving same or similar results for a given set of input situations/conditions does not guarantee its overall correctness for all situations. The real system may act in a totally different manner when exposed to different circumstances. Thus, validation of an ABM becomes a critical and at the same times a difficult task, owing to the dynamic nature of a real system where the environment/conditions changes over time and so does the systems behavior.

1.1 ABM and Validation

Remondino and Correndo (2006) suggests that validation in ABM can be broadly divided into three categories: *Empirical Validation*, *Predictive Validation* and *Structural Validation*. *Empirical Validation* compares the results obtained from ABM simulation with those observed in real world system, thus giving an idea of how correct the model is for some given situations. However, it does not guarantee delivering accurate results for all situations that can be observed in real world, for the modeled system. *Predictive Validation* focuses on reliability of the outcomes produced by an ABM, for unseen situations that are not directly observable in real world. This kind of validation plays an important role in studying the behavior of real world system for situations that are non-repeatable. *Structural Validation* ensures that the processes used by an ABM to generate the outcome comply with those used in the real world system. This kind of validation involves examining and decomposing the model to make sure that all interacting parts are the same as the corresponding real ones. Of the three validations discussed, *Predictive Validation* has been realized in this research work.

From our research perspective, we have identified two important aspects for keeping an ABM consistently validated against its environment. These include *Improving* and *Maintaining*. Since we are measuring the validation of an ABM in terms of successful predictions it makes, therefore the terms *Improving* and *Maintaining* are used along with the prediction capability. *Improving* here refers to the process of improving prediction capability of an ABM, after it loses its validation due to an inevitable change in the environment. The *Maintaining* aspect refers to maintaining the capability of correct predictions after the change has been absorbed by ABM. This aspect plays an important role in forcing an ABM to be consistently in synchronization with its environment.

Before proceeding further, we consider it necessary to provide the context in which the term "learning" has been used with ABM, in order to avoid any misinterpretations. In our thesis work, we intend to look at this term from the perspective of acquiring new information, in the form of rules. More precisely, the phrase "Incorporating learning into agents" describing section 3.1, refers to incorporating adaptability in agents, so as to make them capable of updating their existing rules/decision tree corresponding to the changes occurring in their simulation environment. Therefore, the term "learning" is used in reference with the updations that an agent makes to its existing set of rules/decision tree, in order to absorb the changes occurring in the simulation environment.

1.2 Current Research Motivation

The above discussion highlights that validation is an important player in determining the widespread acceptability of any ABM. Thus, no matter how correctly an ABM models a real world problem, it cannot be put to use until it provides reliable outcomes. Analyzing

the nature of the problem, we identified two factors that tend to affect the validation and reliability of an ABM. These are briefed as below:

1. *Dynamic nature of environment*: The environment in a real world system changes over time in an unpredictable manner and so does the systems behavior. Under such circumstances, an ABM validated against initial environment conditions is bound to become invalid, over time, and hence loose reliability of its results. E.g. an ABM learns a rule *"if weather is sunny then play golf"* at timestamp t_1. This rule gets changed to *"if weather is sunny and its raining then don't play golf"*, at timestamp t_2. Since the ABM does not accommodate this change, therefore it continues to use the original rule and hence becomes invalid after timestamp t_2.

2. *Amount of data available related to environmental changes*: Prediction of correct outcomes by an ABM largely depends upon the amount of data it is trained upon. E.g. if there is a rule saying that *"if weather is snowy or rainy or foggy or windy then don't fly helicopter"*. However, if the weather for an input situation reports volcanic clouds then what will be the outcome of ABM? In real world, *snowy*, *rainy*, *foggy*, *windy* and *volcanic clouds* can be categorized as bad weather conditions, therefore one should not fly helicopter. Since ABM did not experience such a situation during its training, therefore it may end up making a random guess about the outcome. Thus, lack of sufficient training data again affects the reliability of an ABM.

Of the two highlighted problems, this research is motivated to handle the issue of *Dynamic nature of environment* so as:

1. To improve and maintain the overall reliability of an ABM, in terms of making correct predictions.

2. To enable an ABM to be consistently in synchronization with its environment, at all times by absorbing environmental changes.

1.3 Thesis Contribution

In this research, we introduced an approach for developing a predictive ABM that more realistically enacts and adapts according to the dynamically changing environment. The dynamic environment here refers to the incrementally updating training dataset, upon which the ABM is trained. In order to realize this goal, an ABM's agent should possess cognitive capabilities to be able to learn from its past experiences and accordingly update its behavior. It should also be able to detect a change in the environment, following which it can start synchronizing with the changed environment. Considering these requirements, an integrated approach has been developed that takes care of the following two individual aspects, to improve and maintain the overall reliability of ABM:

1. Incorporating intelligence in agents by using decision tree as the data mining technique, thereby making them capable to learn from their past experiences and update their knowledge to absorb environmental changes.

2. Using decision tree similarity measure to detect environmental changes and determine an agent's synchronization level with the environment. The later one helps in monitoring the learning behavior of agents, thus keeping a check on computational costs.

The case study described in chapter 4 demonstrates the implementation of this approach and thereafter analyze and discuss the results obtained. In addition to introducing

above approach, the other goals of this research include:

1. Study the effect of factors such as, memory length, similarity measure threshold, retention rate and acceptance rate, on the learning rate of agents and the synchronization level attained by them.

2. Study the dependency of the factors, memory length and similarity measure threshold, on the nature of the application environment, which can be either dynamic or static.

3. Study the impact of incorporating social interactions among the agents, through the use of social influence factor, on their learning rate.

1.4 Thesis Outline

Considering our research motivation and aim, described in the previous two sections, the work done in this thesis is structured into following chapters.

Chapter 2 proceeds with a background study introducing important concepts related to agent based modeling and data mining. It is then followed by a review of the work done in the field of integrating both of them, focusing specifically on the application of data mining in ABM. It is then followed by a background study done in the field of measuring similarity between two different decision trees. Lastly, a brief study related to the second problem associated with validation and reliability of an ABM, i.e. *Amount of data available related to environmental changes*, is presented.

In chapter 3, the proposed approach is presented focusing on the two aspects of

integrating learning into agents and measuring similarity between two decision trees. Following this, a flowchart is used to describe the overall working of the integrated approach.

Chapter 4 initially goes through some of the case studies that were referred before selecting the *Golf play* as the case study, highlighting the issues associated with each of them. It then describes the *Golf play* dataset along with some pre-processing done to make it suitable to be used as the case study. Following this, the experimental setup along with its components is discussed in detail that was used to implement and demonstrate the proposed approach.

Chapter 5 presents the experimental results obtained from different ABM simulation runs and discusses the effect of various factors on the learning rate and synchronization level of agents.

Finally, the last chapter outlines the conclusion and presents some of the possible future directions for this research work.

Chapter 2

Literature Review

This chapter presents the work that was surveyed to accomplish our research work. According to the research objectives, outlined in chapter 1, we consider it useful dividing the background study into two broad categories. The first category, *ABM and Data Mining*, focuses on the related work done in the field of applying data mining techniques in ABM for various purposes. The second category, *Data Mining and Machine Learning*, covers the work done in the area of comparing two decision trees for computing similarity of knowledge, represented by them. In addition, this chapter includes one more heading, i.e. *Other Background Study*, which describes the research done while pursuing this thesis. Though it is not exactly related to the research described here, but considering its importance and the time spent investigating it, we consider including it in this thesis.

2.1 ABM and Data Mining

This section provides a detailed discussion on the concepts of Agent based modeling simulation and Data mining, which forms the basis of our work. While doing so, various

important aspects related to them are explored, such as *"What are agents in an ABM?"*, *"How to design ABM?"*, *"What are the different data mining techniques?"* and *"What are their respective applications?"*. Lastly, this section throws light on integrating the two methodologies to surpass their limitations, of which, the application of data mining in ABM is considered in detail, covering the related work done in the field.

2.1.1 Agent Based Modeling Simulation

As stated by Drogoul et al. (2003), ABM has gained immense popularity in a wide variety of disciplines, replacing the traditional technologies such as micro simulation (Orcutt, 1957), object oriented simulation (Troitzsch, 1997) and individual based simulations (Harding, 1999). The authors also highlight that different level of representation is required for entities, in order to model different scenarios. A simple object oriented design may be adequate for simple computational entities; however for modeling complex entities that react to the modeling environment or the entities that are cognitive and autonomous in nature, an agent oriented design becomes necessary. Agent oriented software design and complex system design are becoming very common these days. Macal and North (2005) attributes the wide spread usage of ABM to the following reasons:

1. Nowadays the systems have become more complex, in terms of their interdependencies. Due to this it has become really difficult to analyze and model them effectively, using the traditional modeling tools.

2. ABM allows a more realistic modeling of the systems, which have always been too complex to be adequately modeled. Economic markets are one such example of these systems, where a large number of heterogeneous entities interact with each other in an unpredictable manner.

3. Data is now being organized into databases at finer levels of granularity, thus enabling micro-data to support micro-simulations.

4. The computation of large-scale micro-simulation models is now possible due to the availability of substantial computational power, these days.

Baqueiro et al. (2009) defines ABM as a methodology to map a real world system into a computer program, where real world processes are expressed in terms of algorithms and mathematical formulae. These algorithms and formulae are implemented as code in a programming language. The authors also state that ABMs allow the design of experiments to test the developed models and theoretical frameworks, under different scenarios with different parameter configurations. These experiments enable the testing of the behavior of the modeled systems for a given set of assumptions (Banks, 1998). In addition, this experimentation also provides designer with an insight of certain aspects of a complex system which otherwise are not possible to analyze using mathematical analysis.

2.1.1.1 Defining Agents in ABM

An agent is the basic component of any ABM that represents an autonomous entity, capable of reasoning about a given situation and taking decisions independently. However, there is no universally accepted definition for the term "agent" (Macal and North, 2005). Different modelers look at it from different perspectives. E.g. Bonabeau (2002) defines agent as an independent component whose behavior can vary from primitive reactive decision rules to complex adaptive intelligence. Mellouli et al. (2004) on the other hand, defines an agent to be any independent component whose behavior must be adaptive for it to be able to learn from its environment and change the behavior in response. A computer science view

of the agents is provided by Jennings (2000), highlighting essential characteristic of its autonomous behavior.

An agent can have different behaviors depending upon the real world system being modeled (Wooldridge, 2002). However, for an agent to be active and be able to make independent decisions, it should possess the following properties (Wooldridge et al., 1995):

1. *Reactivity*: the ability of an agent to correctly perceive its modeling environment and respond to the changes occurring in it, in a timely fashion.

2. *Pro-activity*: the ability of an agent to take initiative in showcasing the behavior that will satisfy the requirements of its particular objectives.

3. *Sociability*: the ability of an agent to interact with its modeling environment and with other co-existing agents to fulfill its objectives.

4. *Flexibility*: the ability of an agent to possess a range of different methods, which can be used to achieve a given objective and to recover from a failure.

5. *Autonomy*: the ability of an agent to operate without direct human intervention and possess some kind of control over its behavior and internal state.

Another important aspect of an agent is its architecture, which can be classified into four categories: Logic based agents, Reactive agents, Belief-Desire-Intention agents and Layered architectures. A detailed discussion about these architectures and their corresponding examples can be found in (Wooldridge et al., 1995).

2.1.1.2 Designing ABM

As described in (Fishwick, 1997), the general design of an ABM consists of three interdependent stages which are described as below:

1. *Designing Model*: This stage involves constructing a model that is a near replica of the real system under investigation. This stage utilizes the data collected from real world observations, in the form of numerical values and abstract concepts, to build the model. These concepts and data are then formalized using formal mathematical logic.

2. *Executing Model*: This stage involves the conversion of mathematical model, constructed in the previous stage, into computer algorithms followed by their execution. This stage produces numerical outputs, which can be referred to as simulated data.

3. *Analyzing Outcome*: This stage deals with the comparison of simulated data with the data produced by the mathematical model.

The author states that the three stages work in close connection with each other and that the entire simulation process comprises of a finite number of iterations among them.

Although the design stages defined by Fishwick (1997) seems to be complete from the perspective of an ABM design, but as pointed out by Doran (1997) an important step that would have provided a link between the design and implementation of the model is missing. This is because the task of converting a mathematical model into its corresponding computer algorithm is not a trivial one and hence Gilbert and Troitzsch (1999) attempts to

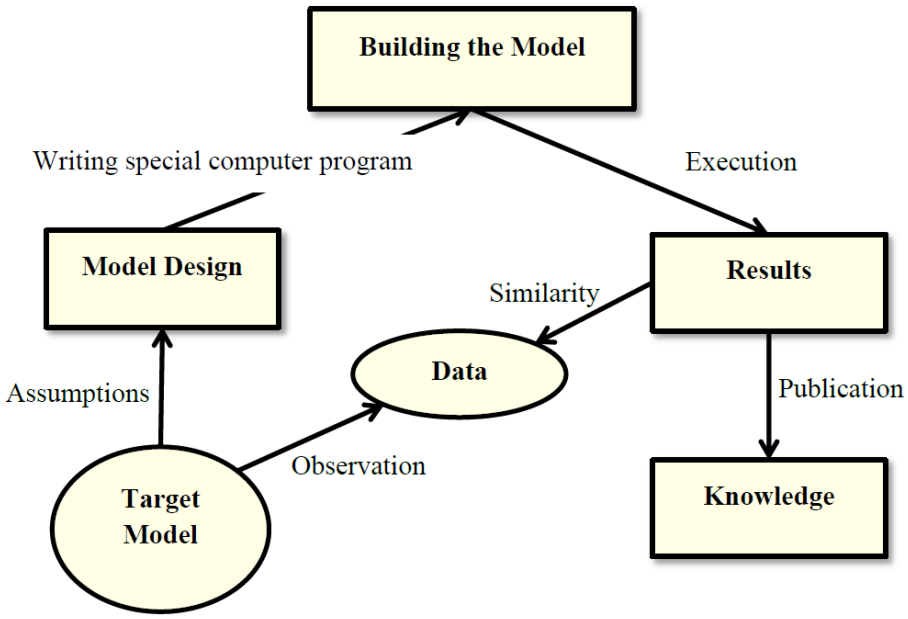

Figure 2.1: The computer simulation design process as described by Drogoul et al. (2003); Gilbert and Troitzsch (1999).

refine this design process by introducing a model building stage into it. Figure 2.1 depicts the refined design process, as referred from (Drogoul et al., 2003).

This design process turns out to be more practical in the sense that it describes the mathematical model in terms of available assumptions. The developers can write special programs for converting the theoretical model into its corresponding computer interpretation. The data generated by executing such model can be compared with the data observed from other independent sources for the purpose of validation. If the generated data set does not conform to the observed data, the entire process can be repeated with altered assumptions until a validated model is obtained.

2.1.1.3 Applications of ABM

The increased popularity of ABM has motivated its use across diverse fields of discipline. As Macal and North (2005) states, the applications of ABM vary from minimalist academic models to large-scale decision support systems. Minimalist models capture only the most salient features of a system, based on a set of idealized assumptions. Decision support models tend to be large-scale applications, which are designed to answer real-world policy questions. Following are a few specific examples of ABM applied in different fields of research:

1. *Social Science*: Yamada et al. (2006) proposed a software platform for developing applications that required the use of socially intelligent agents. Such agents possess the capability to recognize social emotions and act accordingly. The authors justified their work by implementing an application of socially intelligent agents for eLearning.

2. *Business and Economics*: Palmer et al. (1994) presented an ABM of artificial stock market. The overall market behavior is represented by the collective behavior, emerging from interactions among individual autonomous and adaptive agents. This is in contrast to the traditional models which employ rational expectations approaches. Another ABM for banking management and strategy development is presented by Robertson (2003). This model targets to explore the strategy development process for a Bank in order to maximize its output while competing with other networked banking institutions.

3. *Health Care research*: An ABM has been proposed by Kobti et al. (2006) that targets at improving the knowledge and understanding of the Drivers towards child seat re-

straint usage, on highway. The model incorporates the use of evolutionary algorithms to achieve its objectives.

4. *Manufacturing Industry*: Jennings and Bussmann (2003) discussed on the applicability of ABM in Complex Control Systems of Manufacturing industries such as vehicle assembly plants. The authors justified their idea by stating multiple scenarios where agent based software model can be applied to control complex control systems. In addition, they also highlighted the motivation behind such applications.

2.1.2 Data Mining

Data mining is the process of extracting valuable and meaningful information from tremendous amount of data, using data analysis and discovery algorithms (Kantardzic, 2011; Fayyad et al., 1996). It is one of the significant steps in the overall process of knowledge discovery in databases (KDD), where other steps like data cleaning, data pre-processing etc. are coupled with it to identify potentially useful and comprehensible patterns in data. In the literature, the terms "Data Mining" and "KDD" are generally used synonymously.

Kantardzic (2011) defines data mining in a more formal manner as: *"Data mining is a process of discovering various models, summaries, and derived values from a given collection of data."* , where the author highlights that the data mining process generally consists of five steps which are executed iteratively to extract the required knowledge. These steps include: *defining problem statement*, *collecting data*, *preprocessing data*, *selecting appropriate data mining techniques* and *interpreting results*. Based on the most common usage of data mining in different fields, it can be categorized into two broad classes as described below (Kantardzic, 2011; Fayyad et al., 1996):

1. *Predictive data mining*: It focuses on building a model that maps a situation, in terms of conditions describing it, to its corresponding output. Such a model can then be used to predict the outcome for unseen or future situations.

2. *Descriptive data mining*: It involves constructing a model that is capable of drawing out useful patterns from data, but in human understandable format. It tends to provide a description for data that is easy to comprehend.

2.1.2.1 Data Mining Techniques

Data mining consists of different methods to achieve the primary goals identified in the previous discussion. A brief introduction to some of the important methods is presented next (Kantardzic, 2011; Fayyad et al., 1996):

1. *Classification*: It is the process of determining a mapping function that can be used for classifying a new or existing data item into one of the several predefined classes, accurately (Weiss and Kulikowski, 1990). In order to construct such a classificatory function for each class, this method requires a dataset that relates data items to their corresponding classes. For example, classification of trends in financial markets (Apte and Hong, 1996), identifying objects of interest in large image databases (Fayyad, Djorgovski, and Weir, 1996) are some of its applications. Some of the classification algorithms include K-nearest neighbors (Cover and Hart, 1967), Naïve Bayes classifier (McCallum et al., 1998), Support Vector Machine (Cortes and Vapnik, 1995) and decision trees (Quinlan, 1986).

2. *Regression*: It is the process of determining a mapping function that can effectively map a data item to a real-valued prediction variable. In other words, regression in-

volves analyzing the change in the value of dependent variable by varying the value of one of the independent variables and fixing others. It also assists in understanding the relationship that can exist between the dependent variable and one or more of the independent variables. Some of the applications include, but are not limited to, predicting the amount of biomass present in a forest using remotely sensed microwave measurements, predicting consumer demand for a new product relative to the advertising expenditure.

3. *Clustering*: It is identified as one of the unsupervised learning methods in data mining. This technique works by grouping similar objects into one subset representing a distinct class (Hegland, 2004). These subsets can be mutually exclusive and exhaustive and can even be hierarchical or overlapping in nature. Example applications include identifying groups of similar costumers in marketing databases, identifying subcategories of spectra from infrared sky measurements (Cheeseman and Stutz, 1996). Hegland (2004) identifies four broad categories into which clustering algorithms can be divided. These include *Partitioning methods*, *Hierarchical clustering methods*, *Density based methods* and *Model based methods*. A detailed discussion about each of these categories, along with approaches they include, is presented in his work. The author also mentions that clustering is generally used in conjunction with classification for various data mining tasks.

2.1.2.2 Applications of Data Mining

Over the recent years, data mining has emerged as a powerful tool to analyze large amounts of data. Today, many different fields such as healthcare, research and business are developing data mining models to extract hidden information from large databases. Kantardzic

(2011) in his research throws light on some of the areas where data mining has and is being applied.

Telecommunication industry is using data mining to answer questions like *"How to retain customers when other companies offer lucrative offers and low rates?"*, *"What offers should be made to the customers to convince them into buying additional services?"* etc. Big companies like *Worldcom*, *BBC TV* and *Bell Atlantic* are mining customer related data to improve their selling power, determine the best time for broadcasting a program and selecting a suitable technician for resolving customer issues depending upon their severity.

Retail industry is another area where retailers are looking for data mining models that could tell them the latest product trend, types of product that can be sold together, the ways to retain profitable customers, customer buying behavior etc. Some of the retail companies that are utilizing the power of data mining to unveil answers to these questions are *Safeway(UK, grocery)*, *RS Components(UK, electronics)* and *Kroger Co.(USA, grocery)*.

Healthcare and biomedical research is another important area where data mining is used to analyze large amounts of patient related data and data associated with medicinal research. This area becomes critical due to the involvement of human life and hence, expects data mining tools to be more accurate with lowest possible errors. For example, the continuous analysis of time stamped data associated with a patients health can provide important information on the progress of the disease. The concept of neural networks was used by NeuroMedicalSystems and Vysis Company to perform a pap smear diagnostic aid and protein analyses for drug development, respectively.

2.1.3 Integrating ABM and Data Mining

Despite the widespread use of ABM and data mining, there are still some issues associated with each of these methodologies that hinder their effective implementation in any field of discipline. Baqueiro et al. (2009) in their work highlight the open problems/limitations of these approaches and propose to integrate them in order to overcome their limitations, to some extent. A detailed discussion about these issues is presented next.

In ABM, one of the major issues is the lack of standard techniques for its verification and validation. Due to this, the researchers from fields other than the computer science are very reluctant to use ABM as a tool to model real world systems. The verification process takes care of correct transformation of the abstract theoretical model into the computer program. Validation, on the other hand, ensures that the computer based model is an accurate representation of the modeled real world system. It does this by comparing their respective behaviors. Thus, verification and validation together adds credibility to an ABM and make its results trustable. Further, the high complexity of modeled systems makes it difficult to establish some standards for verification and validation. Despite this limitation, each model can still be verified and validated using techniques specifically designed for it.

Although there exists several pitfalls in data mining, but lack of relevant data for mining is the most significant of them. Though different organizations have abundant data, but how much of this data is readily usable for analysis is a big question. Relevant data, in terms of data mining, means the data with minimum noise, inconsistency, distortion and error. However, expecting such data for mining is a big thing because most of the data stored in data warehouses and organizational databases contains noise and is highly erroneous. As stated by the authors, a majority of data analysts' time goes in cleaning

the data and preparing it for mining. Thus, it is very difficult to accurately detect and correct erroneous data. The quality of data is further damaged when it undergoes automated processes like data integration and data pre-processing.

Following are the two ways, as introduced by the authors, in which the two approaches can be integrated to overcome each other's limitations.

1. *Applying data mining in ABM*: The authors claim that some sort of validation process can be formalized by the use of data mining in ABM. Further details on this topic are covered under the next heading of *"Application of data mining in ABM"*.

2. *Applying ABM in data mining*: With this approach, ABM can be used to generate sufficient amount of data (simulation results) for a domain specific data mining task, when real data is not available. In addition, this approach intends to promote the use of various data mining techniques in different application domains, without having to worry about the availability of real data. Thus, the use of ABM in data mining should be able to handle the data issue to a great extent, as claimed by the authors.

Since a part of our research focus on incorporating some sort of intelligence in agents, using data mining techniques, therefore a detailed discussion on the approach "Applying data mining in ABM" is presented next, along with related work done in the field.

2.1.3.1 Application of Data Mining in ABM

The authors in (Baqueiro et al., 2009) believe that some sort of validation process can be formalized for ABM by using data mining techniques in it. More precisely, they propose using data mining techniques like classification rules, clustering rules and sequential rules

on the data obtained from ABM simulation, to extract a high level behavior of the model. In other words, they intend to draw meaningful information from the simulation results using data mining techniques. Such information or behavior would be more easy and straightforward to compare with that observed in the corresponding real world system. Here, the focus is on working at the abstract level rather than looking at the low level data values. This would make the task of validation in ABM much easier, provided the outcome data is of good quality and suitable data mining techniques are used. Figure 2.2 diagrammatically represents the inclusion of data mining in ABM at the verification and validation level.

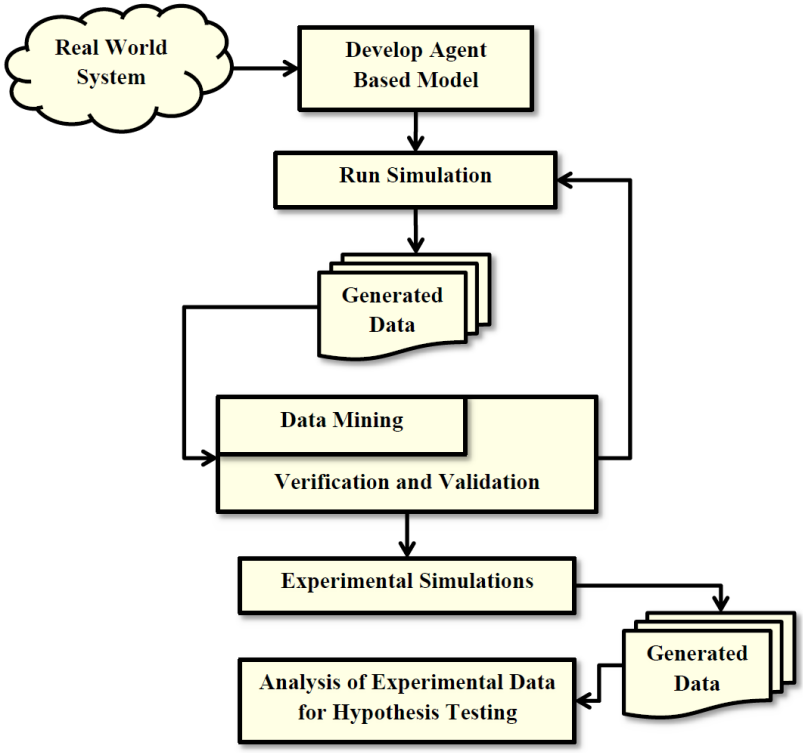

Figure 2.2: *"Applying data mining for the verification and validation of agent based models"* (Baqueiro et al., 2009).

Remondino and Correndo (2005) further classify the application of data mining techniques in ABM into *endogenous* and *exogenous* modeling.

1. *Endogenous modeling*: This type of modeling works at micro-level of ABM, i.e. at the individual agent level. In this approach, some kind of intelligence is assigned to each agent using data mining techniques, making them capable of analyzing and processing the data collected from previous simulation runs. This enables an agent to learn from its historic data and evolve according to the environment in which it acts, in much the same way as humans do. This further helps them reach a local maximum by tuning in some of the ABM simulation's initial parameters (Remondino, 2003). Classification is one such data mining technique that can be used by the agents to learn a classification model from their previously collected experience. This technique can be implemented by using decision tree learning as the machine learning algorithm.

2. *Exogenous modeling*: This class of modeling deals with macro-level of ABM, collectively looking at agents as an artificial society. Here, data mining techniques can be used to reveal interesting and unknown patterns from the simulation data and study the behavior emerging from interactions among the agents. This information can then be used to revise the model and validate it against the modeled system. The authors further elaborate this process using the diagram shown in Figure 2.3. In the first phase, data mining is used to construct a hypothesis over the real world system. Based on this hypothesis, a model is build which is then implemented and executed to produce sufficient amount of simulated data. In the second phase, data mining is again used to construct another hypothesis, but over the simulated data this time. This hypothesis, representing the implemented model, can then be compared with

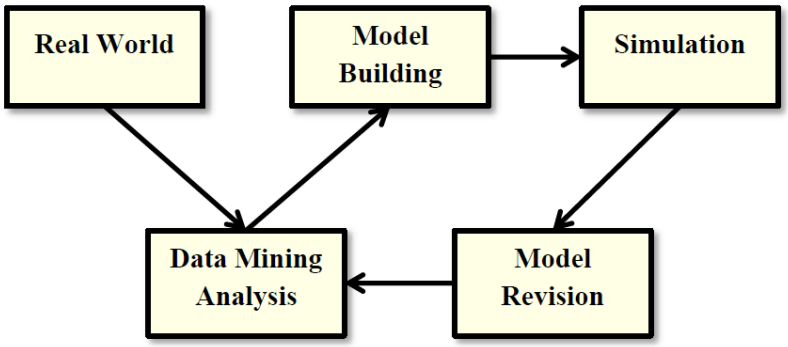

Figure 2.3: *"Data mining revision process applied to agent-based simulation"* (Remondino and Correndo, 2005).

the original one to validate the model. Depending upon the results of comparison, the model can be revised to make it map the real world system as closely as possible. This process can be repeated until the model reaches a desired convergence.

2.1.3.2 Related Work

Although, not much work has been done in the field of applying data mining in ABM (Arroyo et al., 2010), however this section still tries to cover a significant portion of the relevant work done. The work described here also forms the basis of our research.

The work done by Remondino and Correndo (2006) demonstrates the use of exogenous modeling. In their work, the authors intend to handle the problem of "parameter sensitivity" in ABMs, using data mining. The outcome of an ABM run strongly depends upon the selection of parameter values. Even a slight variation in one of the parameters value can lead to an entirely different set of results. Therefore, the model parameters need to be selected very carefully. In addition, selecting correct parameter values for the model also helps in validation of its results. As a solution to this problem, they introduce an ap-

proach called "Parameters Tuning by repeated execution". In this approach, the simulation is run multiple times with different set of parameter values. The outcome obtained in each run is analyzed using data mining techniques to determine if it conforms to the expected behavior. If it does then the selected parameter values are considered to be the suitable one, otherwise simulation is run with different parameter values. This approach helps in identifying unknown dependencies between the parameters and the final outcome. In addition, it also helps in determining the best parameter values that would ensure more realistic results, given an existing situation and enables the implementation of other important things like the what-if analysis and simulation of unreal situations. The authors use an ABM simulating the biological phenomena for some species of cicadas to highlight the working of their proposed approach.

Gostoli (2008)'s work is an example of endogenous modeling, in which the author intends to work at micro-level of ABM by empowering individual agents with a data mining technique to analyze their collected experiences. As described, an ABM is constructed for establishing common and socially shared meanings for a set of symbols, among a society of agents. These symbols are not assigned any meanings in the beginning and the agents are expected to determine their meanings by socially interacting amongst themselves. Here, each agent is assigned a decision tree learning algorithm to analyze and process the data it has collected over time. All agents are made to play a game in pairs of two. In each pair, one of the agents is marked as sender and the other one as listener. The sender agent predicts the meaning for a symbol and sends it to the listener agent, which then checks it for correctness. If it is incorrect, then listener agent sends back the correct meaning to the sender agent following which the sender agent update its memory, otherwise the sender agent continues to predict the meaning for next symbol. The agents begin by randomly

guessing the meanings for symbols. After a few runs, when enough experience has been collected by each agent, they start analyzing it using decision tree learning algorithm. Following this, each agent is able to formulate a set of rules, over time, guiding the assignment of meaning to a particular symbol. After sufficient number of simulation runs, the ABM reaches a stage where all agents share same meaning for each symbol, with small variations.

Arroyo et al. (2010)'s work is another example highlighting the use of exogenous modeling for the development and improvement of ABMs. The authors present a methodological approach describing the use of data mining techniques, along with statistical methods, in designing and initializing the model and validating it against the modeled system. In model building stage, data mining is used to construct a hypothesis of the real world by analyzing and mining the data collected from different sources such as surveys, interviews etc. During this process, the authors suggest the involvement of domain expert to ensure that data analysis results comply with domain theory. Data mining is again used to analyze the simulation results to confirm that the model behaves in the expected manner. The model undergoes modifications if the resulting hypothesis does not conform to the initial one. This process is repeated multiple times to refine the model so that it closely follows the real world system. The authors demonstrate each stage of their proposed approach using the case study of Mentat model, the details for which are mentioned in their work.

Ahmed et al. (2011) describe yet another approach which makes use of data mining in initializing various parameters of an ABM, with relevant data instead of using some random values. With this approach, the authors intend to initialize an ABM's parameters

with relevant values contained in a real world survey data, thus making the ABM mimic a given real world condition as closely as possible. Since these values are not explicitly mentioned in a survey data, therefore they emphasize on using decision tree as the data mining technique to discover them. This would ensure, to some extent, a more realistic behavior of the model and hence simulation results, with a certain degree of correctness, can be expected. The authors summarize their approach using an architecture that integrates data mining and ABM.

2.2 Data Mining and Machine Learning

This section, briefly covering the relationship between data mining and machine learning, provides a detailed discussion on different machine learning algorithms that can be used to achieve data mining tasks in various fields. Further, using decision tree as the machine learning algorithm for accomplishing the task of data mining in our research, it covers the related work done in the field of measuring similarity between two decision trees.

2.2.1 Machine Learning Techniques

Data mining and Machine learning go hand in hand as most of the data mining techniques such as classification, clustering etc., are implemented using different machine learning algorithms. Although data mining looks at a bigger picture of extracting meaningful information from vast amounts of data, but it uses machine learning algorithms as the core component of this bigger process (Mitchell, 1999). Therefore, in this section we briefly introduce some of the popular machine learning algorithms that are being used in different fields to implement data mining techniques. These algorithms include, without covering all

of them, Decision tree Learning, Artificial neural networks and Support vector machines.

1. *Decision tree learning*: This represents a divide-and-conquer approach for data mining tasks of classification and regression. It generally falls in the category of predictive data mining because of its extensive use in building predictive models by revealing and learning unseen patterns in large databases (Myles et al., 2004). Though simple, yet it is a powerful methodology that can be used both for exploratory data analysis and predictive modeling. Moreover, models constructed using decision tree learning algorithms are easy to interpret and understand which makes this approach stand apart from other pattern recognition techniques in the field. The decision trees are constructed in an iterative manner, every time selecting a set of attributes from among the remaining ones, on the basis of some test. These selected attributes constitutes the internal nodes of a decision tree, known as the test nodes. The test nodes can be selected in more than one ways, but approach given by Quinlan (1993) is the most popular one. According to this approach, in every iteration, the attributes which result in maximum information gain are selected as the test nodes. Every value of the test node, in a decision tree, is represented by a branch protruding out from it. All classes are represented as leaf nodes in the decision tree. Moreover, each path from root to leaf node in a decision tree corresponds to a rule, thus a rule set can be derived from the decision tree by traversing all its paths. These rules are generally known as the classification rules. In addition, this approach is covered under the category of supervised learning methods, which requires a set of training examples to construct a predictive model.

2. *Artificial neural networks*: This approach takes its inspiration from the structure and working of a human brain, i.e. by mapping the neuron network in a human brain into

a computational model. With this, it tends to take a big step towards replicating the cognitive capabilities of a human brain. In other words, artificial neural networks aim at generalizing the mathematical models that define the biological nervous systems (Abraham, 2005). Neurons are the basic element of any neural network, which are interlinked using connection weights so as to correctly modulate the effect of associated input signals. A weighted sum of all the input signals is calculated, over time, in order to generate a neuron impulse. The learning capability of an artificial neuron strongly depends upon the connection weights and the learning algorithm chosen. These connection weights can be assigned either explicitly using different methods or by the neural network itself. The learning algorithms come into play when neural networks have to be trained, to make them capable of selecting these weights in order to produce the expected output. In this regard, neural networks can be trained using three classes of learning algorithms, i.e. supervised, unsupervised and reinforcement learning, the details for which are mentioned in the referred work. In addition, the architecture of neural networks is generally classified into two categories, namely feed-forward networks and recurrent networks. Besides other differences, the feed-forward network does not have any feedback connections whereas they are present in the recurrent networks. More details about these architectures and the design of neural networks can be referred from (Abraham, 2005).

3. *Support vector machines*: This is yet another supervised learning method, introduced by Vladimir Vapnik and colleagues, for binary classification (Boswell, 2002). In addition to the classification tasks, this method can also be used for regression analysis. The concept underlying the working of support vector machines requires the construction of a *d-dimensional* space, where *d* refers to the total number of at-

tributes/features describing each record in the training example set. During training, each example is mapped onto this space as a point and a hyperplane or a gap, wide enough, is determined that clearly separates the two classes in the space. Here, finding such a hyperplane is one of the major tasks, which requires the data in example set to be linearly separable. After this, a new or unseen data record can be classified into one of the two classes by plotting it onto this space and determining the class category it is closer to. Although this approach works well with linearly separable data, however it can also be used for non-linear classification task by using "kernel induced feature space". In this method, the data is plotted into a higher dimensional space, enabling the clear separation of data despite its non-linear characteristics. An advantage for support vector machines is that their performance on unseen data can be measured using VC-dimension, which is not possible in other learning methods like neural networks. Further details on this approach can be referred from (Boswell, 2002).

2.2.2 Measuring Similarity between Decision Trees

While reviewing the work done in the field of applying data mining in ABM, it was observed that most of the approaches make use of decision tree as the data mining technique. Closely following the related work, we also plan to use decision tree as the data mining technique, to enable agents in the ABM to learn from their previous experiences. Thus, each agent uses a decision tree learning algorithm to mine the data it has collected over time and represent the resulting knowledge in terms of decision tree. Since one of our research objectives requires comparing the knowledge of an agent with the knowledge reflected by the environment (also represented using decision tree), therefore this section tends to cover

the related work done in the field of comparing two decision trees for similarity.

2.2.2.1 Related Work

The following paragraphs describe some of the relevant work done in the field of measuring similarity between two decision trees. In all the works described here, it is assumed that the two decision trees are obtained over the same set of features/attributes.

Serpen and Sabhnani (2006) propose to measure the similarity of knowledge represented by two decision trees by comparing the corresponding rule sets obtained from them. In this process, the rule sets are plotted into an *n-dimensional* feature space, where *n* corresponds to the total number of features/attributes, describing the rule sets. For each rule in both the rule sets, a hyperplane is established by referring the conditions described in the rule antecedent. The intersection points of these hyperplanes are used to calculate the hyper-volume of the corresponding partition/subspace of the feature space. Thus, the total hyper-volume for each of the two rule sets, computed separately, represents the partition of the feature space covered by it. Among these partitions, there are some that are commonly covered by both the rules sets. The hyper-volume of these common partitions together with the hyper-volumes of both rule sets gives a measure of similarity between the two rule sets and hence the two decision trees. This similarity is computed with respect to a particular decision class; therefore the procedure should be repeated for every class of interest. The authors present a detailed algorithm outlining the approach and explain it using examples. They also demonstrate its working using a benchmark problem from computer security domain. Although this approach is quite simple and interesting, but it incurs high computational cost, as highlighted by the authors.

The work done by Pekerskaya et al. (2006) deals with the problem of detecting changing regions between any two snapshots of spatial data, taken at different time instants. Since the access to original data is mostly limited by some constraints, therefore the authors intend to use cluster-embedded decision trees to summarize the temporal snapshots of data. These decision trees are stored in a separate database enabling the queries, regarding the change detection between any two snapshots of data, to be answered effectively and efficiently without requiring the access to actual data at those time instants. With this, the problem of detecting changing regions between two different snapshots of data is reduced to comparing two different time-stamped cluster-embedded decision trees. Therefore, the authors present an approach for comparing two cluster-embedded decision trees, a part of which involves deriving the rule sets from both the trees and computing the intersecting region between them. While computing the intersecting region, all rules from both rule sets are compared against each other to find the region commonly covered by them. The complete details about the overall approach, along with its implementation on synthetic and real data are mentioned in the referred work.

Perner (2011), apart from explaining *"How to Interpret Decision Trees"*, presents yet another approach for detecting a change in the original data set that might have occurred over the time. They intend to do this by modeling the data sets at two different time stamps as decision trees and comparing them using the concept of substructure mining. In this approach, all rules in both rule sets (derived from two decision trees) are decomposed into their respective substructures and are then compared against each other to determine the similarity. The author introduces a formula for calculating the overall similarity using this concept, the details for which are mentioned in the paper. Also included in the paper is the step by step computation of similarity measure, along with explanatory examples.

In addition, some other work done in the field includes: the approach given by Bhatt and Rao (2008) for computing the overlap between two different firewall rule sets, to ensure that the desired changes are incorporated into the new firewall rule set; the work done by Liu and Hsu (1996), as mentioned in (Serpen and Sabhnani, 2006), in which they propose a fuzzy matching technique to compare two different rule sets, one obtained from a rule induction algorithm and the other one gathered from a domain expert. This gives a measure of similarity of knowledge, extracted from the original data set and the one possessed by the domain expert. Moreover, it also helps detect any change that might have occurred in the original data since last modification. Although their approach is effective, however capturing the beliefs of a domain expert, in fuzzy terms, may involve some difficulties.

2.3 Other Background Study

This study includes the research that was done in relation to the second problem associated with the validation and reliability of an ABM, i.e. *Amount of data available related to environmental changes*, as mentioned in section 1.2. We believe that the work described here has essential ingredients to further improve the current approach, by making it respond more accurately and realistically to the environmental changes, for which there is very little or no data available. This would make the ABM map the real world situations more closely, by predicting the changes, with decent accuracy, before they actually happen or in parallel to the situation under consideration. However, due to the lack of a good and relevant case study we were not able to pursue it, but still keep it as a future work for our further research. Although, this work is not exactly related to our present research, but considering its importance and relevancy we find it worth mentioning as a part of this thesis. The

following paragraphs present a very brief introduction to this study.

For any given situation (seen or unseen), a person makes decision by using both the general knowledge and the specialized knowledge, gained from experience (Prentzas and Hatzilygeroudis, 2007). Here, the general knowledge refers to the set of rules, which the person develops over time from what he/she has known or gets to know about the domain of given situation. The specialized knowledge refers to the collection of specific experiences, which the person had in the past and which he/she can relate to the present situation. Considering this, Prentzas and Hatzilygeroudis (2007) state that the human thinking process, under complex situations, can be mimicked by integrating two knowledge representation approaches, namely rule-based reasoning and case-based reasoning. A brief description of these approaches is presented next.

Rule-based reasoning (RBR), as the name suggest, represents the knowledge in terms of facts and rules that can be reasoned using an inference engine (Prentzas and Hatzilygeroudis, 2007). This type of reasoning approach is generally used in expert systems to capture the domain knowledge effectively; MYCIN, an expert system used for medical diagnosis, is one such example of this approach. Since for a given situation, RBR uses logical reasoning on given facts and rules to deduce the outcome, therefore it is categorized as the *deductive reasoning approach* (Chi and Kiang, 1991).

Case-based reasoning (CBR) on the other hand is classified as *inductive reasoning approach*, where a generalization is made from a given set of specific cases. This generalized result can then be applied to an input situation to determine the decision/outcome for it (Chi and Kiang, 1991). In CBR, the knowledge base, generally known as case library, consists of a collection of large number of previous cases with their solutions. A

new/unseen situation is handled using the case library by finding a set of previous cases that are, to great extent, similar to the given situation (Prentzas and Hatzilygeroudis, 2007). One of the cases from this set, having the maximum similarity measure, is selected and its solution is applied to the given situation with some modifications. The authors present a detailed discussion about the advantages and disadvantages of RBR and CBR, along with different ways in which they can be integrated to produce hybrid approaches, to overcome their individual shortcomings.

Although RBR and CBR can be integrated in many different ways to replicate the human thinking process, however following paragraphs mention some of this work that, either directly or indirectly, aims at handling the situations for which either no data was available to train upon or which were never faced earlier.

Kumar et al. (2009) present a hybrid reasoning approach combining RBR and CBR, with CBR as the core system for knowledge representation and RBR as one of its module. In their approach, the authors intend to make clinical decision support system, for all domains of ICU, independent of the domain knowledge. This would considerably reduce the time spent in the collection of not so obvious domain knowledge. Further, they state that the proposed approach will be able to handle the problems of high complexity and changing medical conditions, associated with the use of purely rule-based methods. With this approach, they also intend to handle the unseen situations, for which there may not be any rules defined in the rule-base. The authors demonstrate their approach using the real data, collected from the ICU department of Sir Sunderlal Hospital.

Another approach that integrates RBR and CBR is given by Lee (2008). In this approach, the authors use RBR and CBR as independent modules, but in sequence to one

another, for internal auditing of the bank. There are two broad stages defined in the overall process, i.e. the screening stage and the auditing stage. The screening stage is handled by RBR, where suspicious transactions are screened using standard guidelines defined for the department. CBR is used at the auditing stage to determine the punishment for the employees who are involved in the suspicious transactions. This approach requires the involvement of a domain expert for the case adaptation process. Case adaptation is the process in which the solution of the case, found most similar to the current situation, is reviewed and modified so as to make it applicable to the current situation. The authors highlight the working of their approach using the internal regulations and transaction data from K-bank in South Korea.

Golding and Rosenbloom (1996) present another approach for integrating RBR and CBR to increase the overall reasoning accuracy of the system, taking into account the knowledge from different sources. RBR and CBR are used as independent modules, with CBR as the evaluator of the results produced by RBR. Initially, RBR module is used to determine the rule that is most suitable for the given input situation. This rule, instead of being applied to the situation, is critiqued by the CBR module using the negative exemplars. This critiquing process is performed in the *combination module* where negative exemplars are used to highlight that the selected rule is not applicable to the input situation. The working details of the *combination module* are covered in the paper, with elaborative examples. The authors tested their proposed approach in the domain of name pronunciation, which is an important problem in text-to-speech synthesis. They name the resulting system as *Anapron*.

In addition, the works described in (Rossille et al., 2005; Shi and Barnden, 2005)

are some other efforts, done towards integrating RBR and CBR for tackling various issues, which otherwise are difficult to handle by using purely either of the two approaches.

Chapter 3

Solution Framework

This chapter presents the theoretical framework for the proposed approach. The overall framework is intentionally divided into two sub-sections for better understanding. The first section details the approach intended towards incorporating intelligence in ABM agents, to make them capable of learning from their past experiences. This would enable them to be consistently in synchronization with their environment and continuously evolve according to it by absorbing the inevitable environmental changes. The second section presents the part of the approach that deals with measuring similarity between two decision trees, to compare the knowledge represented by them. This measure enables the agents to determine their level of synchronization with the environment and detect any changes occurring in it, regularly. Apart from this, a third section is also included in this chapter to highlight the overall working of the proposed approach, by integrating the components described in the earlier two sections. The framework outlined in this chapter is an attempt towards building a predictive ABM, with the aim of improving and maintaining the overall reliability and validation of the ABM, measured in terms of prediction accuracy. More details about prediction accuracy will be covered in chapter 5.

3.1 Incorporating Learning into Agents

3.1.1 Review of Previous Work

The previous work done in the field of applying data mining in ABM describe some of the good approaches, which are effective in their own areas of application. However, each of them has some issues that hinder their direct applicability when it comes down to using data mining in ABM for incorporating learning in agents. Below paragraphs present a re-review of the related work in this regard.

The work described in (Remondino and Correndo, 2006) is very specific in nature, in the sense that it deals with determining the parameter values of an ABM by analyzing the simulation outcome using data mining techniques. In addition, this approach works at an abstract level exemplifying the working of exogenous modeling.

The approach given by Arroyo et al. (2010) works at macro-level of ABM and focuses on improving the overall design and validation of the model. However, our research aims at improving the reliability of an ABM by working at individual agent level. Therefore, the work done by authors is out of the scope of this thesis. Moreover, as stated by the authors, their methodology is more effective in cases when large amounts of real world data is available for analysis.

Ahmed et al. (2011)'s approach does not directly incorporate the use of data mining in ABM, either at the micro-level or at the macro-level. Instead, it uses data mining to analyze the real world survey data to reveal unknown information, which is then used to initialize the parameters of an ABM. This is yet another approach, which deals with the

problem of selecting relevant values for the parameters. Although it aims at making the overall behavior of an ABM more realistic, but does not contribute towards adding any sort of intelligence either at the agent or at the ABM level.

3.1.2 Proposed Approach

Gostoli (2008) in his approach, intends to incorporate learning in ABM agents for establishing the meanings for a set of symbols, having no meanings assigned initially. These meanings are developed by making agents socially interact to share their understandings of these symbols. Over time, a common and socially accepted set of meanings is derived for these symbols. While doing so, the author misses out on two important aspects related to an ABM: first, the author did not take into account the dynamically changing environment, which is common to have in ABMs. This may result the ABM to come with a set of meanings that might have been correct, but by the time simulation ends the actual meanings of these symbols might get changed. Second, the meanings for these symbols are solely established by agents themselves, by social interactions without any guidance, i.e. there is no provision to guide them towards acquiring correct meanings. Accordingly, they might come up with a set of incorrect meanings for these symbols.

Considering the above discussion, we propose an approach that although closely follows Gostoli (2008)'s work, but intends to cover the highlighted issues. The proposed approach takes care of the dynamically changing environment in an ABM, by making agents absorb the changes occurring in the environment and evolve according to it. At the same time it uses dynamic environment as a guide for the agents, to enable them acquire and maintain correct knowledge over time by being consistently in synchronization

with it. Therefore, in addition to just interacting amongst themselves, the agents also interact with a global agent to handle these aspects. Here, the global agent, though is not a part of the ABM, is used to represent the dynamically changing environment. The knowledge contained by this agent is referred to as the global knowledge, which essentially is the knowledge represented by the dynamic environment. In addition, it is important to mention that the environmental knowledge is expressed using a decision tree, obtained over a given training dataset . In the beginning, all agents are initialized with some prior knowledge that differs from the global knowledge. Following this, with every run of the simulation, the agents tend to update their knowledge, referred to as the local knowledge, to attain higher similarity with the global knowledge and reduce the gap between the two.

In order to handle the aspects discussed in previous paragraph, an agent should be able to store and access the data related to its past experiences/behavior/knowledge. It should also be able to update these records, as and when it learns some new information or gets its existing information corrected by the global agent. In addition, it should also be equipped with some data mining technique. This will enable an agent to analyze the data in its storage and build a prediction model, in terms of meaningful rules, that will guide its behavior in the future situations. Corresponding to these requirements, Figure 3.1 helps in visualizing an agent in our approach .

As shown in Figure 3.1, each agent is assigned a definite memory where it can store the data related to its past behavior, along with the outcomes. Since this memory has a finite length, to keep low on computational resources, therefore the old records will be wiped out as new ones arrive. Further, each agent is empowered with a C4.5 decision tree learning algorithm (Quinlan, 1993) to enable it to analyze and induce learning from its

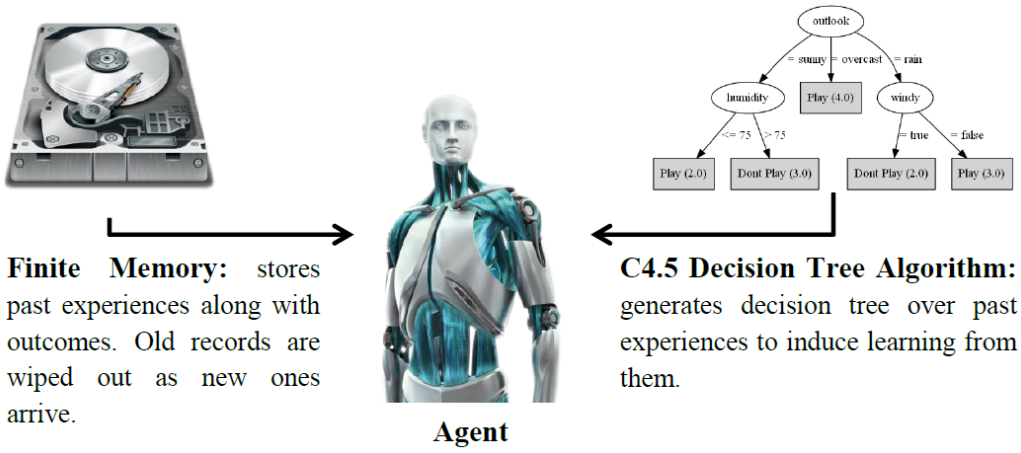

Figure 3.1: An agent in the proposed approach.

collected data. Here, it is important to state that the decision tree, among other machine learning techniques, were opted for the task of data mining due to their simplicity in use and easily comprehensible nature . Thus, our approach defines another example of *endogenous modeling*, by using decision tree as the data mining technique in ABM. Moreover, based on the use of data mining in our approach, it can further be classified as one of the *predictive data mining* types.

During the simulation run, at any given time, a change in environment/global knowledge will trigger the learning process in agents following which they will start synchronizing with the global knowledge. However, two important questions arise here, i.e. *"how will an agent detect the environmental change?"* and *"how will an agent determine if it has attained complete synchronization with the environment?"*. The answer to both the questions can be given by measuring the similarity between an agent's local knowledge and the global knowledge and defining a threshold over this value. This threshold is referred to as the similarity measure threshold (SMT). Following this, whenever an agent's similarity measure value falls below the SMT, it gets to know that a change has occurred in the envi-

ronment. Similarly, when an agent's similarity measure value either reaches or crosses the SMT then it knows that it has attained complete synchronization with the environment.

Following this, another question related to the SMT arises as *"why does a threshold needs to be defined for similarity measure value?"*. This question can be answered by considering an example case of a frequently changing environment. In such an environment, forcing agents to achieve complete synchronization with it will result in an increased and inefficient use of computational resources. This is because the agents in this case will enter into an infinite learning process, in an attempt to get synchronized with the environment. However, since the environment is changing frequently therefore learning process in agents will never stop and they will never attain stability. Considering this scenario and many others like this, it becomes important to define a threshold over the value of similarity measure so that an agent can stop synchronizing and hence stop the learning process, once its similarity measure has either reached or crossed the threshold. At this point, it does not matter if the agent attains complete synchronization or not. For example, if the SMT is set to 80% then the agent will stop learning once its similarity measure either reaches or crosses the threshold and will again start learning if it falls below that mark, depicting a change in the environment. The calculation of similarity measure is detailed in the following section.

3.2 Measuring Similarity between Decision Trees

3.2.1 Review of Previous Work

While reviewing the previous works, done in the field of comparing two decision trees to measure the similarity of knowledge represented by them, we came across some shortcomings in them. These drawbacks are briefly discussed in the following paragraphs, which review the related work once again.

The approach given by Serpen and Sabhnani (2006) involves comparing two decision trees by plotting the rule sets derived from them into the feature space, followed by measuring the subspace covered by them separately and collectively. Their approach, as stated by the authors, involves high computational cost while computing the common subspace covered by both the rule sets. This is attributed to the reason that the authors intend to divide the entire feature space into different subspaces and then determine which of these are covered individually and commonly by both the rule sets.

Pekerskaya et al. (2006)'s approach to compute similarity between two *cluster-embedded decision trees*, for detecting the regions of change between any two snapshots of data, is very specific in nature as it tends to focus on the use of *cluster-embedded decision trees* for representing the temporal snapshots of data. However, as a subpart of their overall approach, they propose a good methodology for determining the region of intersection between two decision trees. Similarly, the approach given by Bhatt and Rao (2008) works with firewall rule sets, computing the overlap between them, thus again making it specific in nature.

Perner (2011) gives yet another approach for measuring the similarity between two decision trees, but focusing on their structural aspects. Although the approach is well defined in a step by step manner and we could not find any major loopholes in it, however there is one thing the author does not elaborate on, i.e. the use of substructure mining for decomposing rules into their respective substructures. This being a major step in the overall approach makes it difficult to understand the calculation of structural similarity between two decision trees.

3.2.2 Proposed Approach

This sub-section describes an algorithm for finding similarity between an agent's local knowledge and the global knowledge. However, since an agent's local knowledge and the global knowledge are both represented using decision trees, therefore the problem of comparing their knowledge is reduced to finding similarity between two different decision trees, corresponding to it. Here, it is assumed/important to note that both the decision trees are obtained over same set of features/attributes. In other words, they are described by same set of attributes and hence correspond to the same feature space.

After having done a detail analysis of the related work in this field, we realized that there are two aspects to finding similarity between two different decision trees, i.e. semantic and structural. In simple words, semantic similarity can be defined as the aspect of the overall similarity that measures the common feature subspace covered by two decision trees for a particular decision class. Structural similarity on the other hand, as the name suggests, compares two decision trees structurally focusing on their overall structures. The approaches described in the related work either consider the semantic similarity (Serpen

and Sabhnani, 2006; Pekerskaya et al., 2006) or the structural similarity (Perner, 2011), but none tends to provide a complete similarity by taking into account both the aspects. Therefore, in our proposed approach, we intend to consolidate these two similarity components to define the overall similarity as the combination of semantic and structural similarity. This is one of the contributions of our research.

Here, we would like to answer an important question that *"why should structural similarity be considered when most of the previous approaches focus only on the semantic similarity?"*. This can be explained with the help of the following weather example.

Consider the following two rule sets obtained from different decision trees:

Rule set 1: *if outlook = overcast then play golf.*

Rule set 2: *if outlook = overcast, humidity <75, windy = false then play golf.*

Consider the following input weather conditions:

Outlook = *overcast*,

Humidity = *70*,

Temperature = *68*,

Windy = *false*

The above two rule sets are highly similar, semantically, as they give the same outcome of *play golf* for the given input conditions. However, they are structurally different, as can be observed from the number of attributes covered in the antecedent part of their respective rules. Here, considering only the semantic similarity may give misleading results by giving a higher value for similarity than it actually is. Hence, the structural similarity needs to be considered.

In our proposed approach, we intend to integrate the work done by Serpen and Sabhnani (2006) and Pekerskaya et al. (2006) in order to overcome their respective limitations of being highly cost inefficient on resources and specific in nature. More precisely, the methodology for computing intersecting region between two decision trees, described in (Pekerskaya et al., 2006), is integrated with the work detailed in (Serpen and Sabhnani, 2006) to give a more generalized approach while reducing the burden on computational resources. The algorithm for the proposed approach, integrating both semantic and structural similarity, is defined in Algorithm 1. This algorithm is divided into three parts: the first one outlines the initial preparation steps, common to both semantic and structural similarity computation. The second and third parts highlight the steps required for calculating semantic and structural similarity, respectively. Finally, the overall similarity measure is given as the average of the values of semantic and structural similarity.

Here, it is important to note that the proposed algorithm computes similarity measure between two decision trees for a particular decision class and therefore should be run separately for every decision class. Moreover, the calculation of complexity for this algorithm is not covered here, as it is out of the scope of this thesis.

Algorithm 1 *Decision Tree Similarity Finder* Algorithm

{***Initial preparation steps***}

1. Obtain the rule sets R_1 and R_2 from input decision trees DT_1 and DT_2.

2. Determine set of attributes A_1 along with their lower and upper bounds used in R_1.

3. Determine set of attributes A_2 along with their lower and upper bounds used in R_2.

4. Represent each rule in R_1 and R_2 as shown below:

$r = \{(L_1,U_1),(L_2,U_2),\ldots(L_n,U_n)\} \rightarrow class\ label$,

where L_n and U_n are the lower and upper bounds of n^{th} attribute in the rule, respectively.

Note: i^{th} rule from R_1 and R_2 will be represented as r_i and r'_i, respectively.

5. For each attribute in $A_1 \cup A_2$, check if it is present in all the rules in $R_1 \cup R_2$. If an attribute is found missing in a rule, then add an interval of $(-\infty,\infty)$ in the respective rule. This ensures that every rule in $R_1 \cup R_2$ contains all attributes.

{***Semantic similarity specific steps***}

6. Initialize *subspace_coverage* to 0.

for every rule $r_i \in R_1$ **do**

for every rule $r'_i \in R_2$ **do**

determine $(r_i \cap r'_i)$ as follows:

$$(r_i \cap r'_i) = \{(max(L_{i1},L'_{i1}),min(U_{i1},U'_{i1})),$$
$$(max(L_{i2},L'_{i2}),min(U_{i2},U'_{i2}))\ldots$$
$$(max(L_{in},L'_{in}),min(U_{in},U'_{in}))\}$$

calculate the subspace covered by $(r_i \cap r'_i)$ as follows:

$$subspace\_coverage\ +=\ (min(U_{i1},U'_{i1}) - max(L_{i1},L'_{i1}))*$$
$$(min(U_{i2},U'_{i2}) - max(L_{i2},L'_{i2}))\ldots*$$
$$(min(U_{in},U'_{in}) - max(L_{in},L'_{in}))$$

end for

end for

Note: *subspace_coverage* for $(r_i \cap r'_i)$ will be 0 if either:

a. $(r_i \cap r'_i) = null$, or

b. All attributes in $A_1 \cup A_2$ are not present in $(r_i \cap r'_i)$.

7. The final value of *subspace_coverage* gives the semantic similarity between decision trees DT_1 and DT_2, representing the common feature subspace covered by them.

{***Structural similarity specific steps***}

8. Initialize *struct_sim* to 0.

for every rule $r_i \in R_1$ **do**

  for every rule $r'_i \in R_2$ **do**

    for every attribute $a_j \in A_1 \cup A_2$ **do**

      if a_j is present in both r_i and r'_i **then**

        if $(max(L_{ij}, L'_{ij}) < min(U_{ij}, U'_{ij})$ **then**

          $struct\_sim += (U_{ij} - L_{ij})/(U'_{ij} - L'_{ij})$ {smaller diff. should be kept in numerator}

        end if

        if $(max(L_{ij}, L'_{ij}) == min(U_{ij}, U'_{ij})$ **then**

          $struct\_sim += 1$

        end if

      end if

    end for

    struct_sim = *struct_sim* / $|(A_1 \cup A_2)|$

    if new *struct_sim* value obtained for r_i is greater than previous one, then store the new value else discard it.

  end for

end for

9. Average out the *struct_sim* values for all rules r_i over $|R_1|$. Here R_1 should always correspond to the decision tree with larger rule set.

10. The final value of *struct_sim* gives the structural similarity measure between decision trees DT_1 and DT_2.

11. The overall similarity measure is calculated by averaging the values of semantic similarity and structural similarity.

The following discussion illustrates the calculation of semantic and structural similarity with the help of an example, taken from (Serpen and Sabhnani, 2006). The decision trees, for the computation of semantic similarity, are represented in terms of rule sets. In addition, it also demonstrates the working of Algorithm 1, but from a higher level following the explanations for each similarity approach.

Semantic Similarity Calculation:

Consider the following rule sets, where f_1 and f_2 are the attributes and C_1 is the decision class of interest, for which semantic similarity has to be calculated. In addition, the range of each attribute is also specified.

Rule Set 1:

$R_{11} \rightarrow$ If $f_1 \leq 15$ then C_1

$R_{21} \rightarrow$ If $f_1 > 35$ and $f_2 \leq 1.5$ then C_1

Rule Set 2:

$R_{12} \rightarrow$ If $f_1 > 15$ and $f_2 \leq 1.5$ then C_1

$R_{22} \rightarrow$ If $f_1 \leq 15$ and $f_2 > 2.7$ then C_1

Attribute Range:

f_1's range = $0 - 50$

f_2's range = $0 - 3.0$

Figure 3.2 shows the plot of above rule sets in the feature/attribute space, constructed using the attribute ranges. Here, since there are only two attributes, therefore each rule in both the rule sets is mapped onto a 2-dimensional hyperplane.

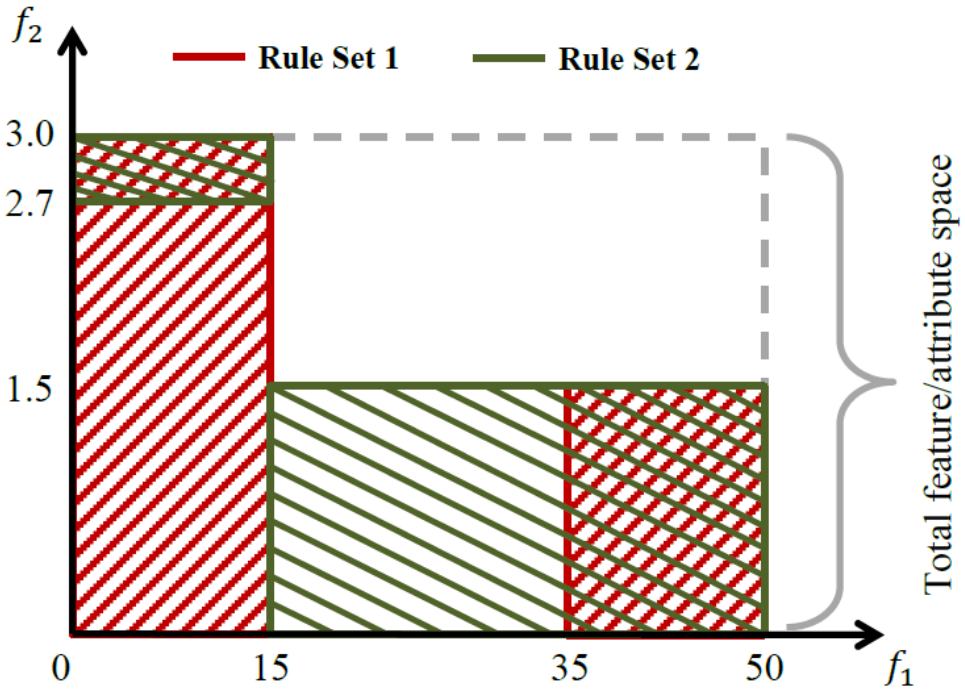

Figure 3.2: A plot of subspace covered by rule sets 1 and 2 in the feature/attribute space.

From the above feature/attribute space plot, the semantic similarity can be given as:

Semantic Similarity = Percentage of total feature/attribute space covered by overlapping regions/hyperplanes.

$$= \{(50-35)*(1.5-0)\} + \{(15-0)*(3.0-2.7)\}/(50*3.0)$$

$$= 0.18$$

Structural Similarity Calculation:

Consider the decision trees, shown in Figure 3.3, corresponding to the above rule sets 1 and 2. These decision trees were constructed using the open source graph visualization software, Graphviz [`http://www.graphviz.org/`].

1. Calculating structural similarity for rule R_{11} of rule set 1, against all rules in rule set 2.

   (a) Comparing the rules R_{11} and R_{22}, as shown in Figure 3.3, the structural similarity can be given as:

   struct_sim for $f_1 = 1$

   struct_sim for $f_2 = 0$

   struct_sim for R_{11} & $R_{22} = (1+0)/2 = 0.5$

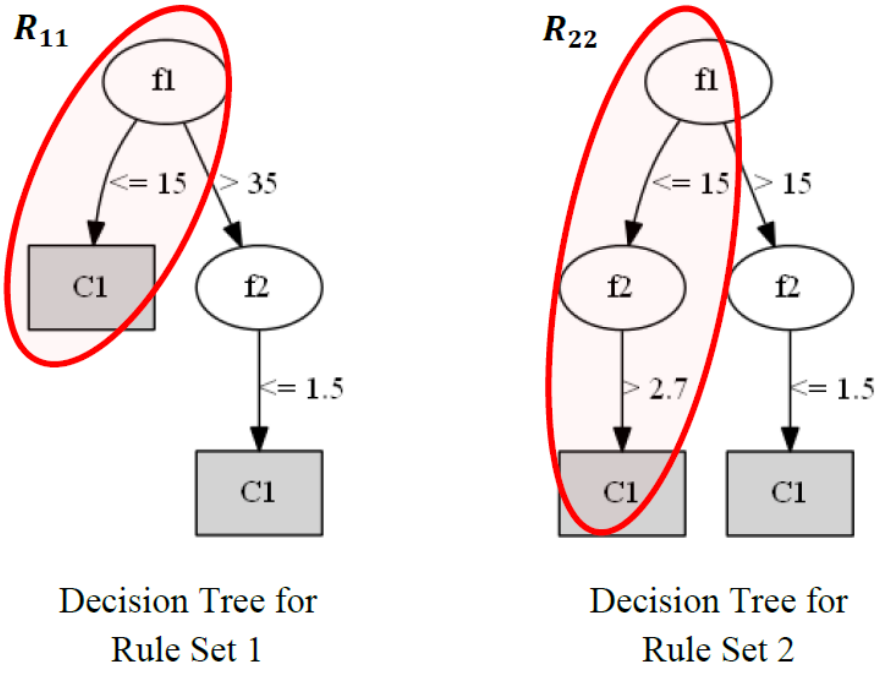

Figure 3.3: Comparing rules R_{11} and R_{22} for structural similarity.

   (b) Comparing the rules R_{11} and R_{12}, as shown in Figure 3.4, the structural similarity can be given as:

   struct_sim for $f_1 = 0$

struct_sim for $f_2 = 0$

struct_sim for R_{11} & $R_{12} = 0$

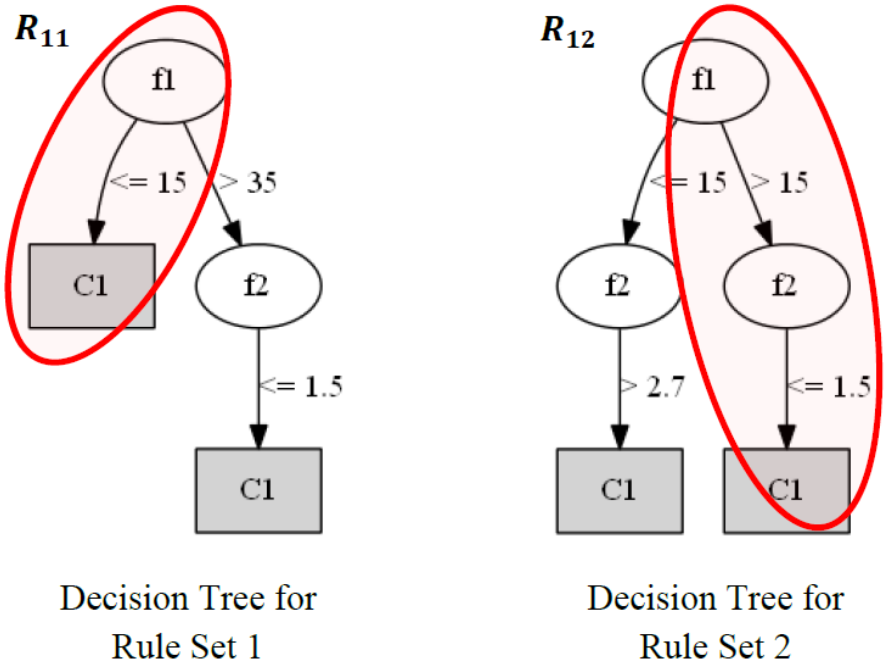

Figure 3.4: Comparing rules R_{11} and R_{12} for structural similarity.

Structural similarity for R_{11} = max(*struct_sim* for R_{11} & R_{22} , *struct_sim* for R_{11} & R_{12}) = 0.5

2. Calculating structural similarity for rule R_{21} of rule set 1, against all rules in rule set 2.

   (a) Comparing the rules R_{21} and R_{22}, as shown in Figure 3.5, the structural similarity can be given as:

   struct_sim for $f_1 = 0$

   struct_sim for $f_2 = 0$

struct_sim for R_{21} & $R_{22} = 0$

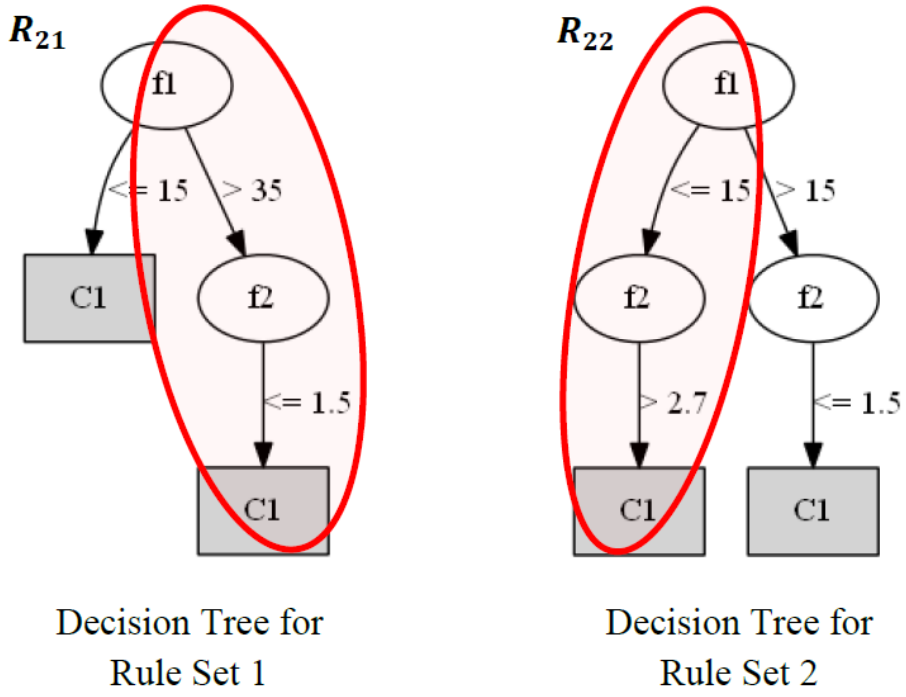

Figure 3.5: Comparing rules R_{21} and R_{22} for structural similarity.

(b) Comparing the rules R_{21} and R_{12}, as shown in Figure 3.6, the structural similarity can be given as:

struct_sim for $f_1 = \{(50-35)/(50-15)\} = 0.42$

struct_sim for $f_2 = 1$

struct_sim for R_{21} & $R_{12} = (0.42+1)/2 = 0.71$

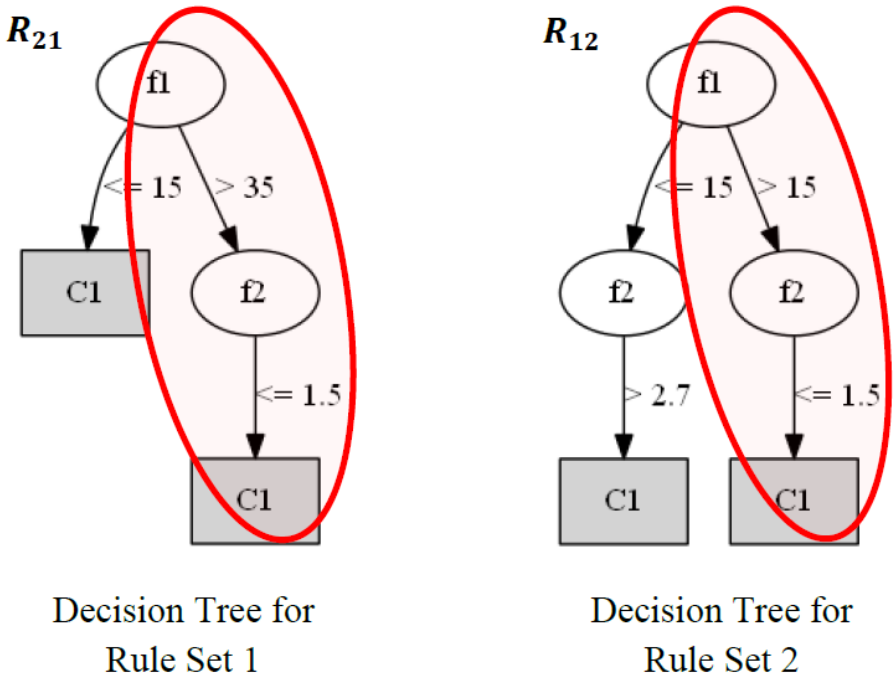

Figure 3.6: Comparing rules R_{21} and R_{12} for structural similarity.

Structural similarity for R_{21} = max(*struct_sim* for R_{21} & R_{22} , *struct_sim* for R_{21} & R_{12}) = 0.71

Structural Similarity = (*Structural similarity* for R_{11} + *Structural similarity* for R_{21}) / 2

$= (0.5 + 0.71)/2$

$= 0.605$

Overall Similarity Calculation:

Overall Similarity = ***(Semantic similarity + Structural similarity) / 2***

$= (0.18 + 0.605)/2$

$= 0.3925$

$= 39.25\%$

3.3 Integrated Approach

In this section, a flowchart is presented to highlight the working of overall approach by integrating the proposed components, explained in the previous two sections. The flowchart shown in Figure 3.7, outlines the general flow of the overall approach. Therefore, the specific components like social influence, retention rate and acceptance rate are not considered here . These components, along with their effects on the progress of the overall approach, will be discussed in detail in chapters 4 and 5.

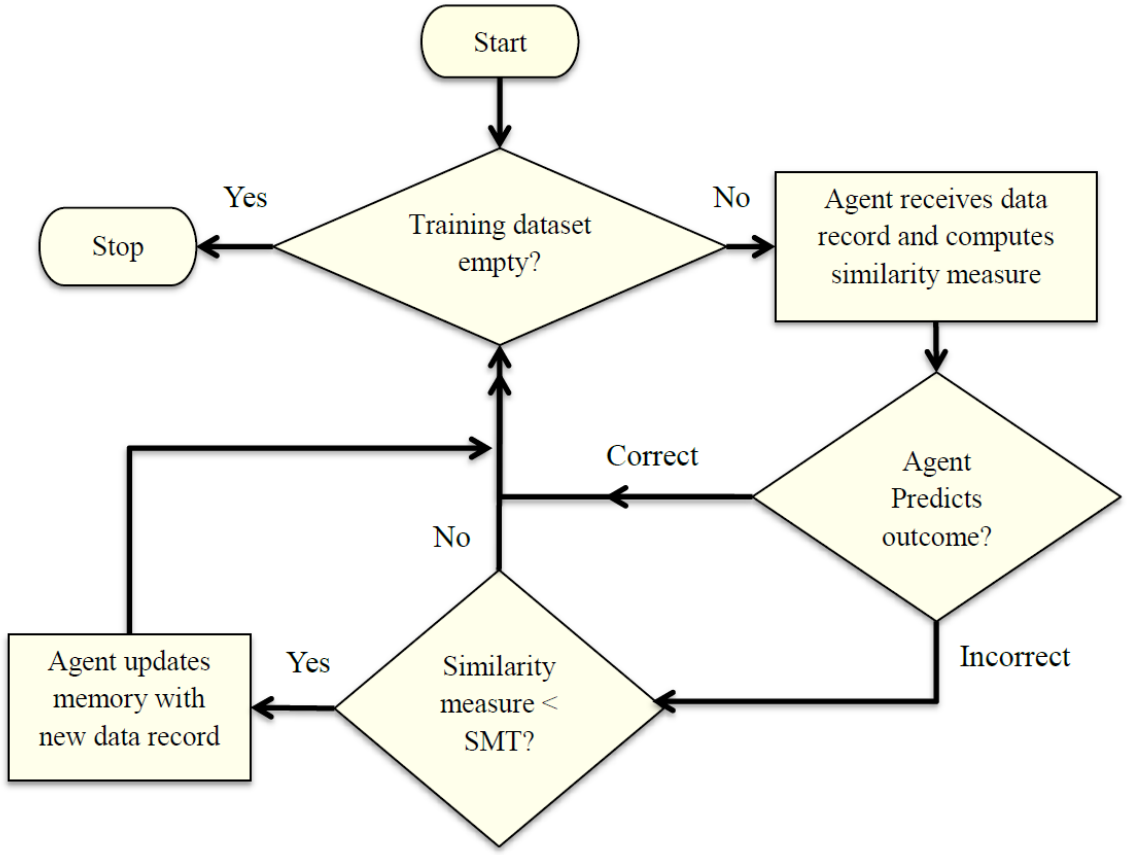

Figure 3.7: Flowchart for the working of integrated approach.

The process flow, highlighted in the flowchart, is repeated until there are no more records left in the training data set. The training data set used here is dynamic in nature, in the sense that every record represents a different set of input conditions. Further, it represents the *generated data* in the experimental setup shown in Figure 4.2 in chapter 4. This data set, when used to represent the simulation environment in terms of decision tree, fulfills the requirement of a dynamic environment in the proposed approach. Further, this decision tree is used as the global knowledge, held by the global agent.

Every time, a single record from the training data set is transmitted to all the agents in the ABM. Following this, each agent computes a similarity measure between its local knowledge, in terms of decision tree, and the global knowledge. An agent then predicts the outcome for the received record using the decision tree, obtained over its past collected experiences using the C4.5 algorithm. If the predicted outcome is correct then it continues to receive the next record, otherwise it proceeds to update its memory based on detection of an environmental change.

In case, when the outcome predicted by an agent is incorrect, it compares its similarity measure value against the SMT value. If the similarity measure value is less than the SMT value, then it indicates that a change has occurred in the environment. Following this, the agent updates its memory with the new record along with its correct outcome and generates a new decision tree to incorporate the change described by this new record. However, if the similarity measure value is greater than or equal to the SMT value, despite the prediction of a wrong outcome by the agent, then it indicates that the agent has already attained its required level of synchronization with the global knowledge/environment. Therefore, as per the proposed approach, it does not require making any changes in its memory and

hence should put a stop to its learning process. Following this, the agent moves on to receive another record from the training data set.

Proceeding this way, all agents regardless of having a different prior knowledge at the beginning or not, will attain the desired level of synchronization with the global knowledge/environment. This is attributed to their consistent detection and absorption of changes, occurring in the dynamic environment, based on the SMT value.

With this, the integrated approach presented in this section meets our research objectives, of incorporating intelligence in agents to make them capable of learning from their past experiences and enabling them to detect and absorb changes as they occur in the dynamic environment, followed by controlling their learning process.

Chapter 4

Case Study

This chapter explains in detail the case study that was used to demonstrate the working of the proposed approach and studying the effects of various factors such as retention rate, acceptance rate etc. on the same. The chapter is broadly divided into three sections, each covering a different aspect related to the case study. The first section provides a brief overview of all the case studies/datasets that were referred to illustrate the working of the proposed work. However, they could not be used due to the lack of one or more features, which are required for the desired case study. These features are discussed along the way, while we review the case studies. The second section describes in detail the case study of *Golf Play*, selected for demonstrating the proposed approach. Moreover, this section also throws light on various issues that were handled to make the case study suitable for use. Lastly, the third section explains the entire experimental setup that was established to demonstrate the approach, along with details of each and every component that contributed towards its construction. This section also covers the description of factors that tends to have an effect on the results of the constructed ABM.

Before proceeding further, we feel that it is necessary to mention some of the characteristic features of the desired case study, which would enable the effective demonstration of the proposed approach. This will also enhance the understanding of the next two following sections. These features are discussed below:

1. *Classificatory in nature*: According to this aspect, for every situation in the case study, described in terms of condition attributes, there should be a decision outcome. This decision outcome, identified by a decision attribute, can be imagined to have results in the form of *'yes'* or *'no'*, *'low'*, *'medium'* or *'high'* etc. Although, it is very unusual for the real world situations to possess classificatory nature; where instead, the outcomes tend to be more complex than just being single values and generally consist of a sequence of actions. Moreover, the outcome depends upon a number of other factors apart from the ones describing the situation. However, for the purpose of demonstrating the working of underlying concept of the proposed approach, we intentionally kept the things simple. Nonetheless, the defined approach can be modified to consider these factors, thus making it map the real world situations more realistically.

2. *Capturing dynamic information*: Since one of the goals of the proposed approach is to keep up the reliability of an ABM against the dynamic environment, therefore the information captured by the case study should be dynamic in nature. This aspect of the case study will enable the modeling of dynamic environment for the ABM. In simple words, the conditions describing a situation in the case study should get changed dynamically with time. Therefore, the conditional attributes of the case study should capture dynamic information, like the climatic conditions which changes every minute. However, finding such a case study becomes difficult since

the attributes of most of the datasets generally captures information, such as properties of an object, symptoms of a disease etc., which is more or less considered to be static because they do not change frequently with time.

3. *Complex decision outcome*: This aspect ensures that the selected case study is complex enough so that the outcome for any given situation cannot be derived using a simple formula. In other words, the dependency of outcome upon conditional attributes is complex enough to be stated in terms of formulae . The weather forecasting is one example of such a complex system, where the outcome represents a complex decision based on a large number of factors.

4. *Easily classifiable attributes*: Some case studies consist of a set of observations, where the attributes, defining each data record, cannot be classified into conditional and outcome/decision attributes. Here, the conditional attributes are the ones describing input conditions for each record and accordingly the outcome/decision attribute represent the outcome for each record. Although such case studies are useful in their own areas of application, however considering the goals of our proposed approach, it requires a case study in which the attributes are easily classifiable into conditional and outcome/decision attributes.

4.1 Referred Case Studies

Many case studies/datasets were referred, before selecting the *Golf Play* dataset as the case study, to highlight the working of the proposed approach. However, we did not find any of them suitable for use with our proposed approach, due to one or more reasons. In this section, a brief overview of all such case studies is presented, along with the reasons for not

using them. These reasons are important to study as they also constitute the requirements for the case study that would be best suited to illustrate our approach.

4.1.1 Robocup Rescue Simulation

Robocup rescue is an initiative taken towards simulating real world disaster scenarios and exploring new ways for coordinating heterogeneous agents in preventing and mitigating the disaster and in search and rescue operations (Tadokoro et al., 2000). The heterogeneous agents generally consist of fire fighters, doctors, policemen etc., which are an integral part of rescue operation in real life. One example of such a disaster can be the outbreak of a fire in the city, which is used by Song et al. (2007) as the case study to highlight their approach of making predictions in dynamic environment.

However, in their approach, the authors focus on using the fire intensity observations, gathered by multiple agents in the simulation, to predict the rate of growth of fire in different buildings. This further helps to predict, in advance, the buildings that will be burning out first. The rate of fire growth is formulated as a function of building materials and building size. The data produced from the simulation is a set of observations, perceived by different agents from different viewpoints. Therefore, the attributes defining each record, such as building_id, time, fire_intensity etc., cannot be separated into conditional and outcome attributes. Following this, the resulting data also lacks in being classificatory in nature. In addition, fire_intensity is the only dynamic attribute in the resulting dataset, whose value is being changed with time; otherwise all other attributes like building materials, building size etc., are constant. These factors prevented us from using Robocup Rescue simulation project as the case study in our approach.

4.1.2 Ozone Level Detection

This is another dataset that was referred from the UCI machine learning repository (Bache and Lichman, 2013). It captures the environment conditions, like average wind speed, relative humidity, solar radiation total for the day etc., in terms of 72 continuous attributes. Since these conditions keep on changing dynamically, therefore the dataset fulfills the requirement of being dynamic in nature. In addition, it also includes an outcome attribute to highlight the ozone level for the day, using the terms *'ozone day'* and *'normal day'*. Thus, the attributes are also separable into conditional and outcome categories.

However, there are still some issues related to this dataset, which the authors in (Zhang and Fan, 2008) intend to handle in order to build a highly accurate ozone level alarm forecasting model. One of these issues is the existence of concept imbalance in the data, following which there is an uneven distribution of outcome classes of *'ozone day'* and *'normal day'*. More precisely, only 2% or 5% of the total records test positive for the *'ozone day'*, which makes it difficult to learn a prediction model for it. Nevertheless, we tried to resolve this issue by using SMOTE (Synthetic Minority Oversampling TEchnique) and Randomize filters, available from the open source Weka library (Hall et al., 2009), in sequence. With the help of these filters, the number of records for the outcome class, *'ozone day'*, was increased and they were randomly distributed throughout the original dataset. Although the dataset was balanced out using these filters, but at the same time lost its originality because each record in the dataset represented the readings for a single day, collected over the period from 1998 to 2004. Hence, the dataset was not used as a case study in this thesis. Moreover, it was realized that the calculation of ozone level can be formulated in terms of certain relevant attributes, referred to as parameters in the authors'

work.

4.1.3 Climate Data

This dataset comprises of real world observations regarding the hourly climatic conditions in the area of Windsor, Canada. These readings were referred from the Environment Canada website `http://climate.weather.gc.ca/`, where each record captures values for various climatic parameters, such as temperature, wind speed, visibility, relative humidity etc. This dataset satisfy all the requirements for the desired case study, except being classificatory in nature, which is obvious considering the different purposes the dataset might be used for.

However, we still tried to suit it to our needs by including an outcome attribute, which can take on different values. Since this dataset was analyzed to be very similar to the selected case study of *Golf Play*; therefore the outcome values for each record were assigned based on the decision tree obtained over the selected case study. Nevertheless, proceeding this way raised a question on the validity of the resulting dataset, because despite having real data, the outcome for each record was synthetically driven following the behavior of some other case study. In addition, there were some additional attributes in this dataset, which were not covered in the selected case study, thus further making it difficult to determine the outcome for each record. Therefore, it was more efficient, in this case, to increase the number of records for the *Golf Play* case study by using the synthetic data generator. The synthetic data generator, along with the *Golf Play* case study, is discussed in detail in the next section.

4.1.4 Enron Email Dataset

Although this dataset is not much related to our work, in terms of its application and usage, yet we would like to discuss it briefly since it was one of the datasets our time was devoted to, while looking for an appropriate case study. As described by Klimt and Yang (2004), this dataset is a static collection of large number of email messages that were collected during the investigation of Enron Corporation. This dataset is publicly available and can be used to perform different research work, concerning emails, such as improving email tools, identifying spam emails etc. One of the areas where this dataset finds great applicability is email classification, which involves developing different techniques for automatically classifying email messages into user-defined folders. More details about its usage and the related work can be referred from the website `https://www.cs.cmu.edu/~enron/`.

4.2 Selected Case Study: Golf Play

After having referred all the datasets, mentioned in the previous section, and many others available on the UCI machine learning repository, the *Golf Play* dataset was found to be the most suitable one for demonstrating our approach. This dataset has been the standard, though unofficially, for evaluating the new methods developed in the field of data mining, especially in the area of decision tree learning algorithms. Quinlan (1986) also used it as the base example to demonstrate his approach of inducing decision trees, based on the concept of information gain.

Moreover , this dataset satisfy all the requirements for the desired case study, by being classificatory in nature, capturing dynamic information, having a complex decision process and easily separable attributes. This makes it an ideal dataset to play the role of

case study in our thesis. Table 4.1 shows the contents of this dataset.

No.	Outlook	Temperature ($^{\circ}F$)	Humidity ($\%$)	Windy	Result
1	sunny	85	85	false	Don't Play
2	sunny	80	90	true	Don't Play
3	overcast	83	78	false	Play
4	rain	70	96	false	Play
5	rain	68	80	false	Play
6	rain	65	70	true	Don't Play
7	overcast	64	65	true	Play
8	sunny	72	95	false	Don't Play
9	sunny	69	70	false	Play
10	rain	75	80	false	Play
11	sunny	75	70	true	Play
12	overcast	72	90	true	Play
13	overcast	81	75	false	Play
14	rain	71	80	true	Don't Play

Table 4.1: Golf Play Dataset

The dataset associates the decision of playing golf with the current weather conditions, where four attributes are used to capture the weather conditions, namely *'Outlook'*, *'Temperature'*, *'Humidity'* and *'Windy'*. Of these, only *'Temperature'* and *'Humidity'* are the numerical attributes and the remaining two are nominal attributes taking on discrete values. This dataset was referred from the University of Regina's website, at the following address: `http://www2.cs.uregina.ca/~dbd/cs831/notes/ml/dtrees/c4.5/`

c4.5_prob1.html.

Though ideal, but the dataset is too small with only 14 records, to be used effectively for highlighting the working of the proposed approach and studying the effects of various factors on its results. Moreover, the records in the dataset tend to be randomly arranged, in the sense that attributes values do not follow any pattern, which is not the case when recording climatic conditions in real world.

Therefore, in order to overcome these problems a synthetic data generator was built to increase the number of records in the original dataset, maintaining its originality and integrity at the same time. In addition, the attributes were also modeled as some function of time, which when used with synthetic data generator produce more realistic records, thus closely mapping the variations in real world climatic conditions.

4.2.1 Synthetic Data Generator

Considering the discussion in the above paragraphs, a synthetic data generator was built to boost the number of records in the original *Golf Play* dataset. While doing this, there were two main concerns, i.e. the generated data should retain the patterns inherent to the original dataset and its overall behavior should not deviate from it, in terms of outcome assigned to each record by the generator. Both the concerns were taken care of by using decision tree, obtained over the original dataset, as the mentor in the process of synthetic data generation. This is because a decision tree can reveal hidden patterns in any given dataset, therefore using one for synthetic data generation ensures that the generated data exhibits, to great extent, the same patterns as present in the original dataset (Eno and Thompson, 2008). Figure 4.1 presents a diagrammatic view of the implementation and working of the synthetic

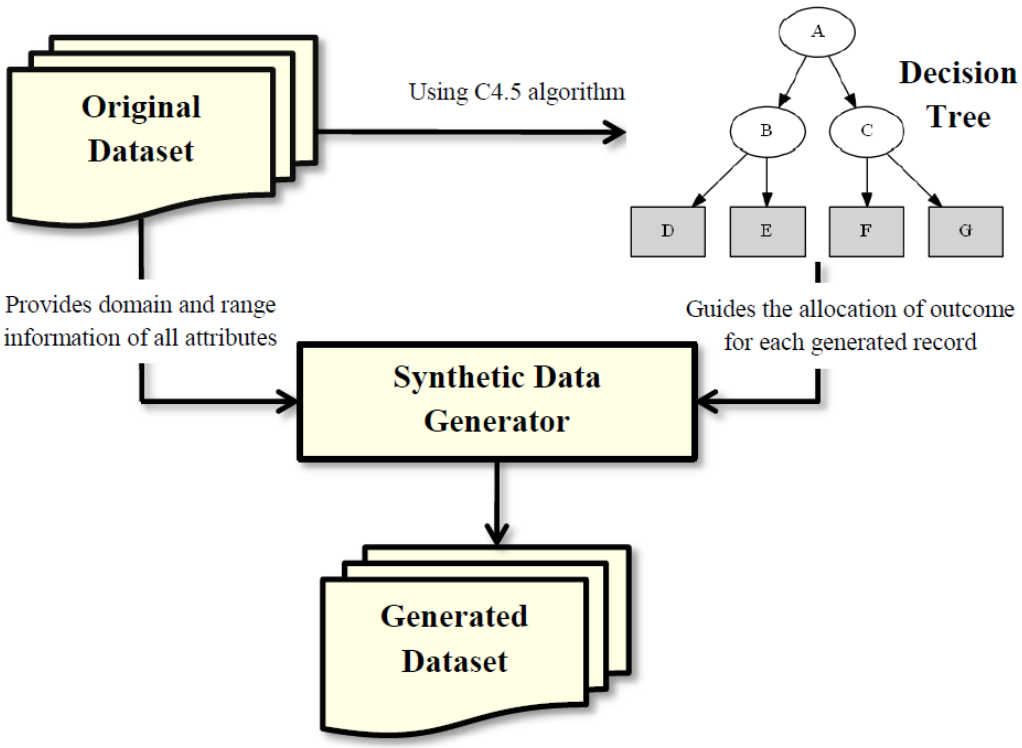

Figure 4.1: Synthetic Data Generator.

data generator.

The synthetic data generator is written completely in java, using some of the functionalities from the Weka library to handle the input dataset files in the ARFF format. As shown in Figure 4.1, it takes two inputs: the first one being the original dataset file, to retrieve information regarding the domain and range of all the attributes. The second one is the decision tree, obtained over the original dataset, to guide the assignment of outcome for each record generated by the synthetic data generator. The output consists of a large dataset containing the desired number of records. This dataset is referred to as the generated dataset and the total number of records in it can be controlled by the user.

4.2.2 Realistic Modeling of Attributes

The data produced by synthetic data generator will be used to model the dynamic environment in the proposed approach. This environment will essentially depict the weather conditions, changing over time. Since the weather conditions in real world change gradually with time rather than randomly, which is the rare case of an extreme weather, therefore the attributes describing the weather conditions should be modeled realistically enough so that their values also follow a gradual pattern of change. Considering this, the numerical and nominal attributes of the original dataset were modeled as sine function and step function respectively. This enabled to replicate the change in simulated weather conditions as close as possible to the real world scenario.

4.3 Experimental Setup

This section covers the details of the experimental setup, along with related aspects, that was used to demonstrate and verify the working of the proposed approach. Initially, we begin by outlining the ABM, constructed for the case study, which is then followed by a brief discussion about the Repast, an open source platform for agent-based modeling and simulation, which was used to implement and simulate the ABM. This is again followed by a precise description of another open source software, Weka, which was used to generate decision trees using C4.5 algorithm and perform other related data mining tasks throughout the experiment. Finally, the section ends with defining various parameters of the constructed model, which were analyzed to have an impact on its overall outcome.

4.3.1 Case Study based ABM

Emele et al. (2012)'s work describe another approach in the category of *Other Background Study* (section 2.3), however it is purposefully referred here with the intention of describing our experimental setup, from the perspective of the problem considered by the authors in it. This would enhance the understanding of the overall setup and hence provide a legitimate ground for its foundation. Moreover, it would help better understand various components used in the experimental setup and their purposes. The next paragraph briefly describes this problem and the solution proposed by the authors.

In multi-agent systems, each agent is assigned a different responsibility. Of these, granting agents with access to different resources is one of the most important tasks. The system can be imagined to have a set of resource agents that provide access to different resources on fulfillment of certain conditions. These agents, as stated by the authors, generally operate under certain policies/rules which guide their decision towards granting access to a particular resource. These policies/rules control the type of resources that can be released to an agent from other organization, the conditions and the purpose under which they can be granted and governs many other critical things related to the resources. In addition, these policies/rules possess the tendency to change with time. Thus, considering their importance and criticality, they are not publicly available to other agents, especially the ones seeking access to different resources, i.e. the seeker agents. Under such circumstances, a seeker agent would have to loop through a list of all the resource agents in order to get hold of the desired resource, thus requiring a lot of time and burdening the computational resources. In this case, having knowledge about a resource agent's policies/rules would help the seeker agent save lot of time, by shortlisting the candidate resource agents which

can most probably release the desired resource. Following this, they can further save time by leaving out the ones that may not even possess the requested resource. Therefore, as a solution, the authors propose that each seeker agent should build up a separate prediction model to acquire each resource agent's policies/rules, based on the interactions between them. This prediction model will help the seeker agent in selecting only those resource agents that carry the resource and will most probably release it, under the given conditions and usage. Thus, the problem reduces down to building a more accurate prediction model for the policies/rules, used by the resource agent, so that each seeker agent can make nearly correct predictions regarding the allocation of desired resources. Further details about the authors' work can be referred from (Emele et al., 2012). Following the discussed problem closely, Figure 4.2 presents our experimental setup.

Figure 4.2 highlights various components of the experimental setup, which will be discussed in detail. However, before proceeding further, we would like to compare this setup with the recently discussed problem, in order to assign a meaningful context to it and make it worthwhile. The *global agent*, shown in the setup, is similar in working to the resource agent, as it provides access to the golf ground by letting agents in the simulation either play or don't play golf, based on the current weather conditions. Here, only one resource allocator agent is considered in contrast to multiple agents in the problem described above. However, the approach can be extended to include multiple resource allocator agents. The agents in the simulation, as shown in Figure 4.2, can be visualized as the seeker agents, seeking access to the golf ground. In this setup, only one resource is considered, i.e. the golf ground, and with this we intend to focus on our goal of improving and maintaining the reliability and validity of the overall ABM, by having all the seeker/agents establish a highly accurate prediction model, to capture the policies/rules followed by the

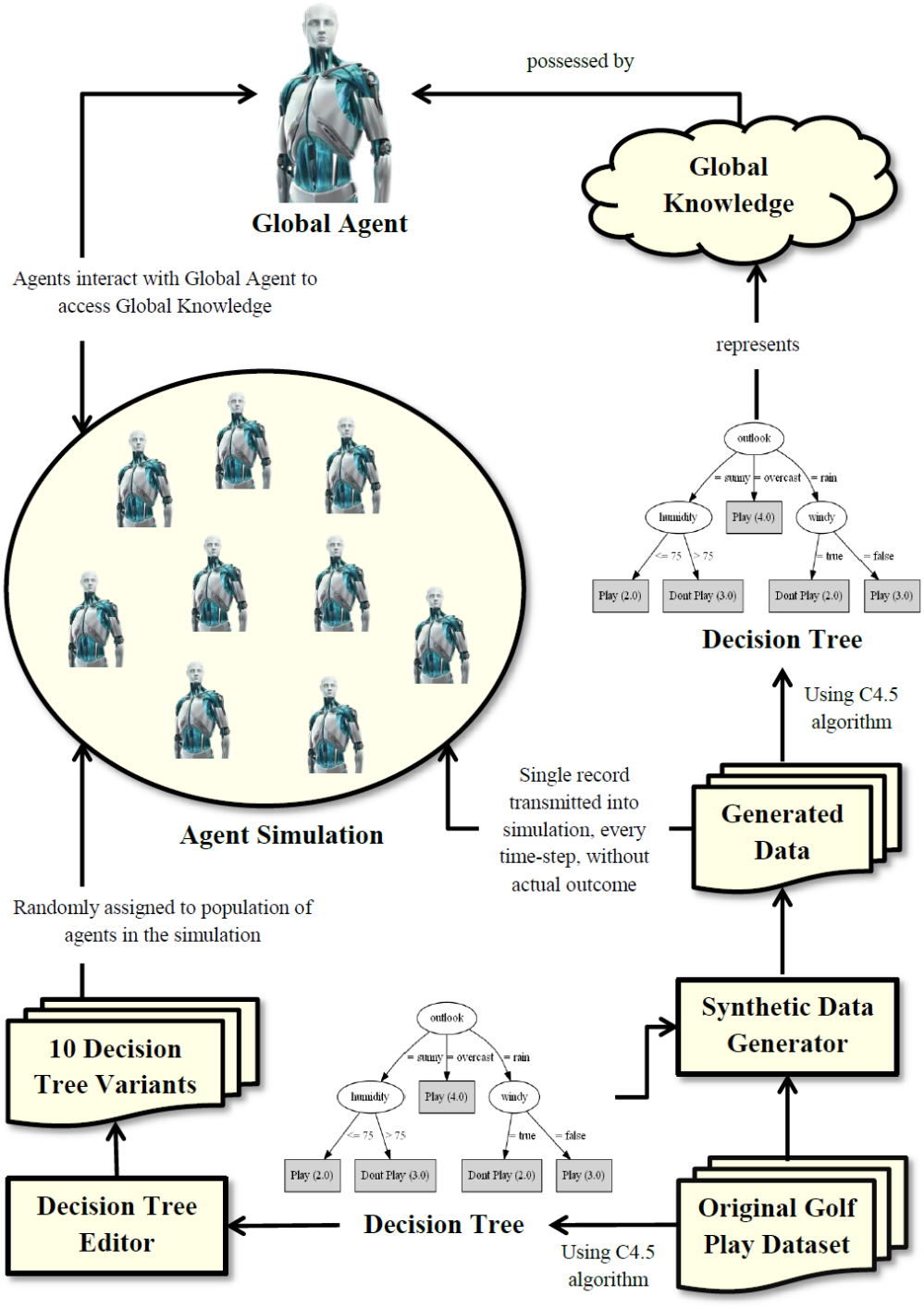

Figure 4.2: Experimental Setup.

global agent, and remain in synch with changes occurring in them. Further, the *generated data* corresponds to the policies/rules followed by the resource agent or the *global agent* in our case, which are dynamic in nature and thus keeps changing with time.

The *synthetic data generator*, as already discussed earlier, is used to extend the original dataset of *Golf Play* by generating synthetic data, using the original dataset and the decision tree induced from it. The resulting dataset, referred as the *generated data* in the setup, is used to represent the dynamic environment/policies/rules, in terms of the *global knowledge*. The *global knowledge* comprises of the decision tree, obtained over the *generated data*, which is assigned to the *global agent* for its easy access by the simulation agents, through interactions with it. The agents will be using the *global knowledge* to detect a change and determine their level of synchronization with it by comparing their own local knowledge against it. The *generated data* will also be used to model the changing weather conditions in the simulation.

The *decision tree editor* is another important component whose purpose is to generate different variants of a given decision tree. The working of this component is highlighted with the help of an example, as shown in Figure 4.3. This example consists of an input decision tree with nodes as A, B, C and outcomes as D, E, F and G. The highlighted nodes, B and C, are selected to carry out a change in the input decision tree. The *decision tree editor* then performs a simple rotate operation around the selected nodes, changing the positions of the siblings with respect to their respective parent nodes. Although this may seem to be a simple change and does not tend to have much effect on the overall decision tree, used in this example, however when decision tree represents rules, with nodes being the test attributes and branches being the tests on the values of these attributes, then even a

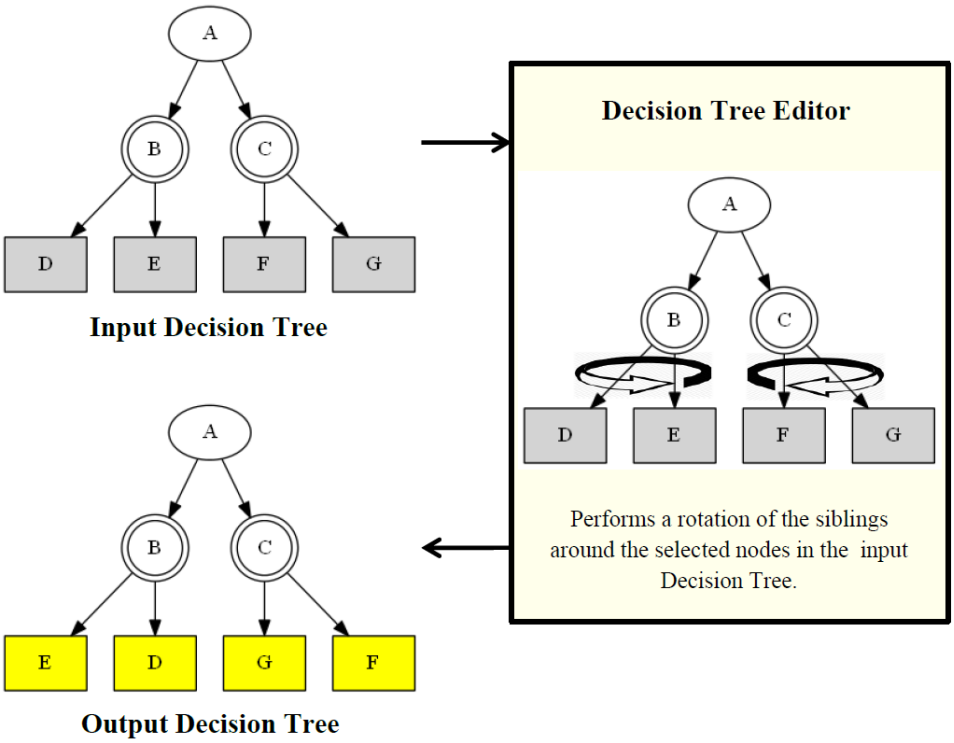

Figure 4.3: Decision Tree Editor.

slight change, similar to the one discussed, can have a huge impact on the rule set derived from the resulting decision tree. For example, a rule *if humidity <75 then play* can get changed to *if humidity <75 then don't play* with this alteration, thus completely changing the rule.

The *decision tree editor* in this setup uses the decision tree, induced from the original *Golf Play* dataset, to produce 10 different variants of the same. These variants are then randomly assigned to the agents in the simulation in order to instill each agent with a prior knowledge, different from the one reflected by the *global knowledge*. This enables to model the worst case scenario, where the global knowledge representing environment/policies/rules has recently changed and all agents are out of synch with it. This

would allow us to analyze the performance of agents during the learning process, to regain the synchronization with the global knowledge. In addition, it would also help us to study the effects of different factors/model parameters, discussed later in this section, on the progress of agents in learning and absorbing the environmental change.

Every time-step, a record from the *generated data*, describing only the weather conditions and without the actual outcome, is transmitted to all the agents in the simulation. Following this, each agent performs three tasks, as discussed earlier along with the flowchart in section 3.3 of chapter 3. In the first task, it measures the similarity between its local knowledge and the *global knowledge*. The second task involves predicting the outcome of *play* or *don't play* under the received weather conditions, using the decision tree learned from its past experiences. The last step requires an agent to update its memory depending upon the correctness of the outcome, determined by consulting the *global agent*. Here, we assume that each time-step lasts for 30 minutes, i.e. every 30 minutes a new set of weather conditions are transmitted into the simulation and each agent attempts to predict the outcome, similar to that of the *global agent*. The decision made by an agent depends upon its historical knowledge and the knowledge it learns in every time step. In addition, an agent also re-generates the decision tree following a change in its memory.

For example, consider a record from the *generated data* as *outlook = sunny, humidity <75, windy = true* (*Class = play*, not transmitted) and an agent's decision tree contains a matching rule as *outlook = sunny, humidity <75 | Class = don't play*. In this case, the agent will predict the outcome as *don't play*. However, it does not match with the actual outcome, *play*, and hence the agent will update its memory with the new piece of information. With every new record arriving into agent's memory, the oldest record will get

deleted. Following this approach, all agents should acquire the desired level of synchronization with the environment, after a sufficient number of simulation runs and thereafter become capable of predicting reliable results.

Since the experiment is setup around granting access to agents to play golf, using the *Golf Play* dataset, therefore the agents can further be visualized as people from different communities, who look forward to play golf, and the *global agent* can be imagined as the supervisor of the golf ground, who controls the access to the ground based on certain rules. These rules are further conceptualized as the *global knowledge* in our setup.

4.3.2 Repast: The Simulation Platform

Repast, *"Recursive Porous Agent Simulation Toolkit"*, is an open source software, available from `http://repast.sourceforge.net/`, that provides a framework for developing and executing agent simulations based on constructed ABMs (Collier, 2003). It provides a whole library of functions ranging from handling the creation of an agent to visualizing the simulation results. Further, it uses swarm that is another popular simulation toolkit as the base, by closely following many of the design features from it. Repast was developed by David Sallach, Nick Collier, Tom Howe, Michael North and others at the University of Chicago (Collier et al., 2003). At present, it is maintained by a non-profit organization *"Repast Organization for Architecture and Development"*, commonly known as ***ROAD***.

Repast is available in multiple flavors, such as C++, Java, Python, Microsoft .Net etc., to suit the needs of different research works. However, our experiment uses the Java version of it, known as Repast J 3.0. Since Repast uses object oriented architecture, therefore for building any agent-based simulation it requires two classes to be defined com-

pletely, i.e. the Model class and the Agent class. The Model class defines the environment of the simulation with which the agents interact during execution. In addition, the Model class also controls the events occurring in the simulation, to which the agents react while pursuing their specific goals. This class can be implemented by extending the *SimModelImpl* class contained in the *uchicago.src.sim.engine* package and overriding a bunch of its important methods, specifically *setup()*, *buildDisplay()*, *buildSchedule()* and *buildModel()*. The Agent class, on the other hand, is responsible for defining the properties, tasks and various other characteristics of an agent, depending upon the underlying problem. Further, depending upon the problem domain, an agent may either work in coordination with others or on its own, may be assigned a dedicated set of knowledge and experience to perform certain tasks, may be connected to other agents using its' assigned address or location, may be defined to work in a master slave relationship and many other possible ways in which it can be setup to work in the simulation. Many of these behaviors can be incorporated in agents using flexible template classes provided by the Repast.

The core functionality of Repast is to guide the occurrence of events in the simulation. The unit of time used to control the triggering of these events is referred to as *tick*, where a *tick* depicts one time-step or one run in the simulation. All the tasks/methods that needs to be executed at every *tick* are implemented and controlled using the objects of classes *BasicAction* and *Schedule* (available in *uchicago.src.sim.engine* package), inside *buildSchedule()* method of the Model class. Thus, *tick* is used to guide the order in which the events should occur relative to each other. The simulation data produced at every *tick* can either be stored in some output files for later analysis or can be visualized graphically using a variety of charts provided by the Repast library. These charts, consisting of histograms, bar graphs, plots etc., also prove to be useful in constraint sat-

isfaction/optimization problems, where they make it easy to analyze the change in a specific parameter or multiple parameters relative to each other. In our experiments though, *OpenSequenceGraph* from the package *uchicago.src.sim.analysis* was used to plot the simulation results. Finally, the simulation constructed using Repast can be run either from the command line or by using the graphical user interface provided along with it. Although the graphical user interface provides more control over starting, stopping and pausing the simulation, however at the same time it increases the overall running time of the simulation due to the additional work done by CPU in rendering the graphics.

Although we tried our best to cover the basic idea of Repast, but there is still much more to it that is beyond the scope of this thesis. Therefore, we recommend the reader to refer Repast website (mentioned in the beginning) for more details and the list of related publications.

4.3.3 Weka: The Data Mining Software

As mentioned on the project website `http://www.cs.waikato.ac.nz/ml/weka/`, *"Weka is a collection of machine learning algorithms for data mining tasks"*. These algorithms can be applied to the input dataset either directly or invoked from the user defined java code. Moreover, it contains tools for data pre-processing, classification, regression, clustering, association rules and visualization, which proves to be very useful while performing various data mining tasks. The Weka also contains a variety of filters that can be applied on the input dataset during the pre-processing stage. The use of two such filters, i.e. SMOTE and Randomizer, is discussed earlier in the section 4.1.2.

WEKA, *"Waikato Environment for Knowledge Analysis"*, aims to combine a vari-

ety of different machine learning algorithms under one roof, with the purpose of providing researchers an easy accessibility to these algorithms (Hall et al., 2009). As stated by the authors, this project took its motivation from earlier times, around 90s, when learning algorithms were available in different languages, required different input dataset formats and different platforms for their execution. Nowadays, Weka provides a powerful framework for the researchers to develop and test new algorithms, without worrying about the supporting infrastructure for data manipulation and scheme evaluation. Further, it has gained immense popularity, as a tool for data mining research, and in the fields of academics and business.

Weka contains several graphical user interfaces, each providing access to specific functionalities depending upon the underlying task. These interfaces are grouped into three broad categories. The first one is *"Explorer"*, which is the main interface organizing the data mining tasks under different panels. Each of these panels provides access to a subset of data mining tasks, the ones defined by pre-processing, classification, clustering etc. The second interface is named as *"Knowledge Flow"*, which allows the users to construct a complete data flow model using nodes, where, these nodes are used to specify the data sources, the type of filters to be applied on the data, the kind of data mining task to be performed, i.e. classification, clustering etc., the dataset selected for evaluation and the visualization method for analyzing the results. In addition, such data flow models can also be saved for later re-use. The third graphical user interface provided by Weka is *"Experimenter"*. This interface facilitates the comparison of predictive performance of algorithms based on different evaluation criterion, available in Weka. These experiments can involve executing multiple algorithms across different datasets and can also be run across different machines in a network, thereby reducing the computational load on any individual ma-

chine. Again, these experiments can be re-used by saving them in either XML or binary format. Further details about these interfaces along with their usage can be referred from (Hall et al., 2009). In addition, the paper also introduces some of the projects that have been constructed using Weka, thus focusing on its applicability in different domains.

Although Weka provides such powerful interfaces, however in our experiment, which was completely written in java, we used the *weka.jar* file to take advantage of Weka's different functionalities. Specifically, Weka was used to access the input dataset files, which were in ARFF format, to retrieve the domain and range information of all the attributes. In addition, all the decision trees were generated using J48 algorithm from the Weka library, which is the java implementation of the C4.5 algorithm.

4.3.4 Model Parameters

This subsection describes five model parameters that were realized to be important for demonstrating and analyzing the progress of the constructed ABM. The values of these parameters can be tuned using the Repast's graphical user interface. The different values selected for these parameters and their corresponding effects on the simulation results have been presented in detail in the next chapter, dealing with experimental results. Following paragraphs, however, cover a brief description of each of these factors.

Memory Length: This is one of the important parameters that limit the amount of data an agent can store in its memory or, in simple words, can actually remember. In real life, a person does not usually remember each and every experience s/he faces while doing a task. Instead, they tend to remember only the relevant experiences that either update their existing knowledge or provide them with new one. Therefore, closely mimicking our agent

to a real life person, this behavior is replicated by restricting the agents to remember only the relevant experiences/knowledge. This is done by assigning a definite memory length to each agent. Further, it also incurs much less burden on the computational resources, by storing only some of the records concerning an agent's overall experience. In our experiments, all agents are assumed to have the same memory size, which though is in contrast to the real life where people have different capacity to memorize things .

Similarity Measure Threshold (SMT): The value of this parameter defines the level of synchronization an agent should achieve with the environment/global knowledge. Therefore, this parameter indirectly controls the learning process in agents, by making them absorb environmental changes until the similarity measure between their local knowledge and the global knowledge reaches the desired SMT value. Considering the discussion in section 3.1.2 of chapter 3, this parameter plays a significant role in preventing agents from entering into an infinite learning process, when working under a frequently changing environment. Thus, it also helps keep in check the load on computational resources, by having its value selected depending upon the type of environment being modeled in the simulation. The type of environment, i.e. static or dynamic, here refers to the frequency with which the changes are occurring in it.

The next three parameters are closely related to each other, as they reflect some of the behaviors observed in the real world, while learning new information or a change in it. With these parameters, we intend to add a realistic touch to the constructed ABM and at the same time study their impact on its outcome. More precisely, these parameters mimic the accuracy with which people learn new information, their chances of sticking to existing information and not learning the new one, and the role of social influence in

learning the same information which they otherwise ignored, when came across directly. Here, the new information corresponds to the change in environment/global knowledge in our experimental setup.

Acceptance Rate: This parameter depicts the probability with which an agent is able to correctly acquire the new information or a change in the environment. In other words, it defines the percentage chances that an agent will capture the environmental changes without any error. For example, an acceptance rate of 60% means that an agent will correctly learn an environmental change with a probability of only 0.6, therefore in 40% of the cases, when it detects an environmental change, it would learn it incorrectly with some error. Thus, this parameter reflects, to some extent, how people deal with new information in real world. Such a behavior is attributed to many factors, such as lack of concentration, high stress, busy schedule etc.

Retention Rate: This parameter in general reflects the reluctance of an agent in learning the new information. It helps model the retaining capability in agents, thus allowing them to stick with their existing knowledge and ignoring any environmental changes, even upon their detection. The value of retention rate is again defined in percentage units, where for example, a value of 60% means that there are only 40% chances that an agent will absorb an environmental change. This parameter helps model another behavior of real world people where they intend to retain their knowledge, even upon coming across some new information or change, the reason for which can be their cultural background, lack of an influential source etc. Here, it is important to mention that all agents are assumed to have the same values for the acceptance rate and the retention rate in all the experiments performed.

Social Influence: The purpose of this parameter is to incorporate interactions among the agents, so as to enable knowledge sharing amongst them and hence analyze its effect on the outcome. In addition, this would provide another source for the agents to learn new information from, despite their reluctance (specified in terms of retention rate) to learn it directly. Social influence is modeled as a boolean parameter, where the value 'true' is used to activate it. In current experiments, only neighbors are considered for setting up a social network among the agents, using the *k-nearest neighbor* algorithm. All agents are assigned a location on a 2-D plot, on the basis of which 3 (value of k) nearest neighbors are determined for each agent. Thus, this parameter plays an important role in mapping the knowledge sharing process, considering only the neighbors, that takes place amongst the people in real world.

Chapter 5

Results and Discussion

This chapter is broadly divided into two sections. The first section deals with illustrating the working of the proposed approach along with its validation, performed in terms of prediction accuracy. In addition, the section also presents a detailed study regarding the effects of memory length and SMT on the progress of the overall ABM, briefly covering how these parameters are themselves affected by the nature of the application environment, which can be either static or dynamic. As already discussed, these types refer to the frequency with which the changes are occurring in the environment/policies/rules, represented as *global knowledge* in our setup. The second section presents the results of executing the ABM simulation with different values for the parameters, acceptance rate, retention rate and social influence, thus providing a study of its overall behavior from a realistic point of view.

Note: In this chapter, the terms environment, simulation environment and global knowledge will be used synonymously. In addition, all the results shown in sections 5.1.1, 5.1.2 and 5.1.3, along with the discussions that follow each of them, have been taken directly from

our previous publication (Dogra and Kobti, 2013), with little or no modifications.

5.1 Demonstrating and Validating the Proposed Approach

We intend to demonstrate the working of the proposed approach while studying the effect of different values of memory length for agents and the SMT, on their learning rate and synchronization level attained with the environment/*global knowledge*, respectively. Here, the learning rate of an agent in general refers to its capability to absorb an environmental change quickly. Considering the purpose of this section, the parameters acceptance rate, retention rate and social influence were set to the values of 100%, 80% and 'false', respectively. In addition, the decision tree for the *global knowledge* was obtained over the entire *generated data*, thus assuming the availability of entire dataset to the *global agent* at once, which is in contrast to the real world where data is available as a continuous stream.

A population of 20 agents was used for all the experiments in this section, along with 1000 synthetic data records, generated using *synthetic data generator*. Consequently, every simulation ran for 1000 time steps receiving a new data record from the *generated data*, per time step. Thus, a particular time step corresponds to the received record number and therefore, allows us to use the terms time step and record number interchangeably in the following discussion.

The simulation results were plotted using two graph formats shown in Figure 5.1 and Figure 5.2. The first graph in Figure 5.1 plots the total number of agents in synchronization with the environment/*global knowledge*, per time step. More precisely, it highlights the total number of agents, among the population of 20 agents, whose predicted outcome

is same as the actual outcome (referred from the *global agent*), per time step. The second type of graph shown in Figure 5.2 gives an idea of the average learning rate of agents, by plotting the average of similarity measure values of all the agents in every time step. The slope of this graph depicts how fast/slow the agents are learning and adapting to the environmental change.

5.1.1 Effect of Memory Length on Agents' Learning Rate

In order to study the effect of memory length on the cumulative learning rate of agents, the SMT was set to 100%. This enabled all agents to achieve complete synchronization with the *global knowledge* and thus resulting in an average similarity measure of 100%. The ABM simulation was run using three different values for the memory length, i.e. 60, 170 and 120. These simulation runs are referred to as ABM-60, ABM-170 and ABM-120 respectively and the results obtained from them are shown in Figures 5.1 and 5.2, Figures 5.3 and 5.4, Figures 5.5 and 5.6, respectively, along with a brief discussion following each of them.

ABM-60:

In ABM-60, all agents were synchronized with the global knowledge by learning from nearly 362 records and the average similarity measure also reached the value of 100% in the same time. This highlights a faster learning rate in agents with short memory length. However, the population of synchronized agents dropped down to 0 several times in the latter half of the simulation and the same was observed with the average similarity measure curve where values were dropped to 80% and below. Thus, despite exhibiting a fast learning rate depicted by the higher slope value of average similarity measure curve, ABM-60 show instability in maintaining its synchronization level over a longer period of time. The reason

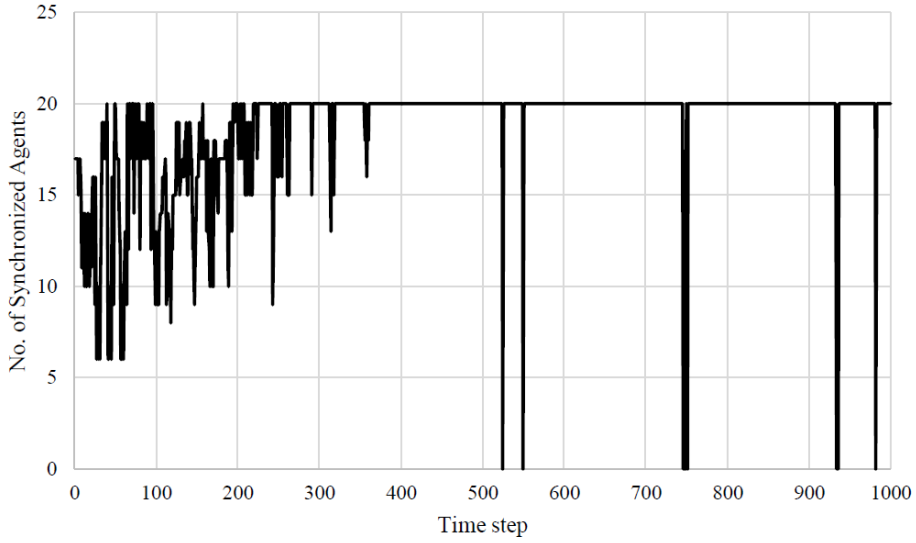

Figure 5.1: Population of agents synchronized with environment for ABM-60.

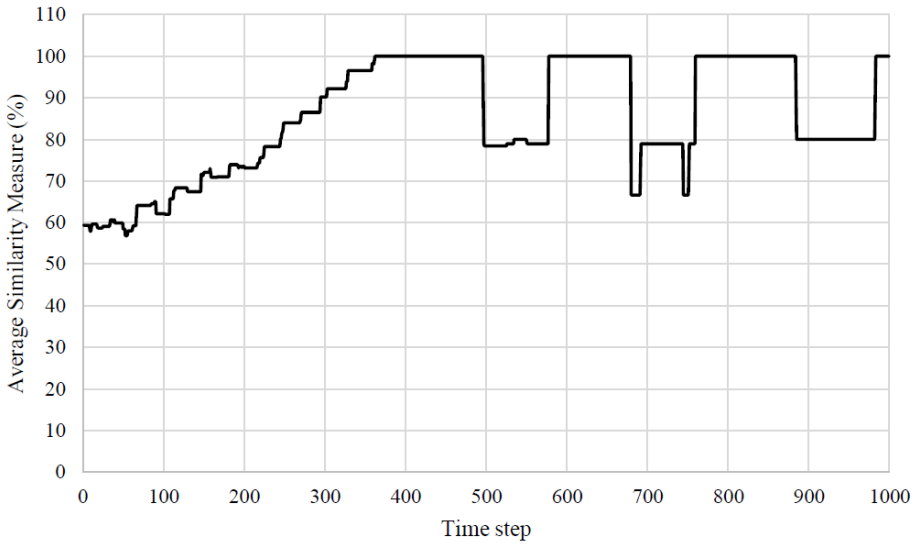

Figure 5.2: Average similarity measure of agents for ABM-60.

for such an outcome is the extremely small memory size, due to which an agent is not able to retain much of its past experiences for a longer time and keeps only the most recent data in the memory. Such a behavior may be desired in situations where environment changes very frequently and a fast learning rate is required with stability being a less priority.

ABM-170:

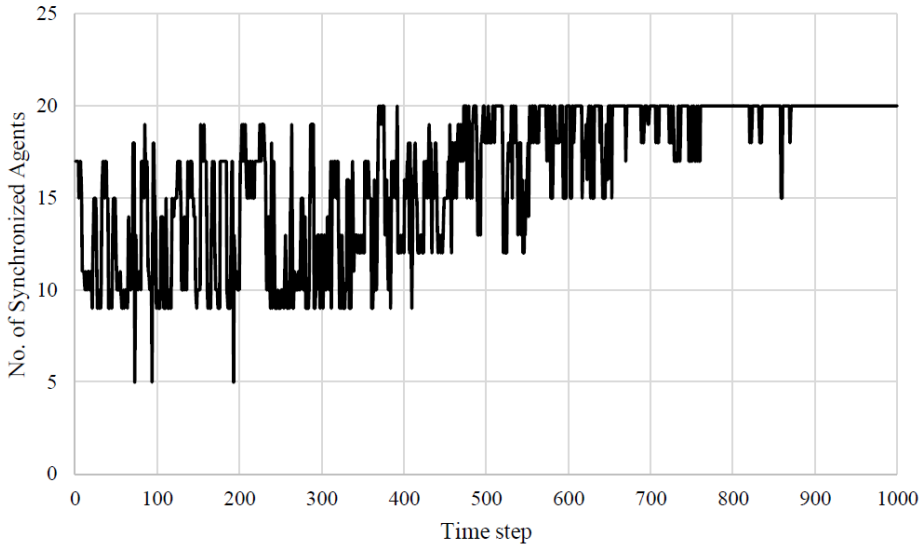

Figure 5.3: Population of agents synchronized with environment for ABM-170.

The results for ABM-170 highlight a massive increase in the average learning time of an agent, as indicated by the smaller value of slope for average similarity measure curve. It takes almost 860 records, more than double the number required in ABM-60, for all agents to attain synchronization with the environment. Similarly, it takes nearly 948 records for the average similarity measure to reach 100%. The reason behind such a huge increase is the extremely large memory size. Due to this, an agent holds on to its past experiences for a very long time and requires sufficient number of records, containing new

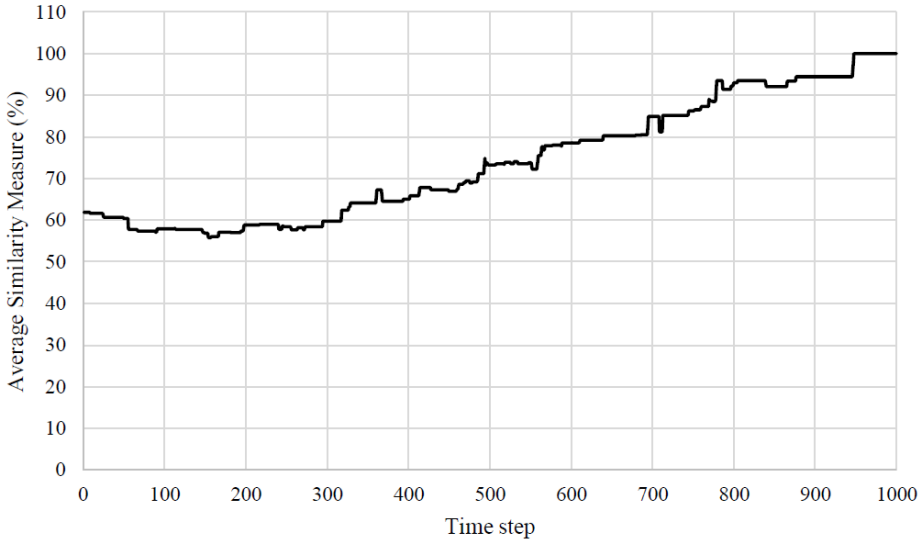

Figure 5.4: Average similarity measure of agents for ABM-170.

information, to update its behavior. However, once synchronized, the agents in this case show high stability in maintaining their synchronization level. Such a behavior is more suitable in situations where environment does not change frequently and a high stability is required over a longer period of time.

ABM-120:

The above two simulation results, for ABM-60 and ABM-170, highlights that smaller memory size ensures fast learning with an unstable behavior and larger memory size results in a highly stable behavior with a slow learning rate. Therefore, there exists a trade-off between fast learning and stable behavior and hence the last test was performed using ABM-120. The slope of average similarity measure curve highlights a decent learning rate among the agents, using nearly 630 records to achieve synchronization with the environment. The number of records required in this case, lies approximately in between the number required in ABM-60 and ABM-170. The agents also show good stability in

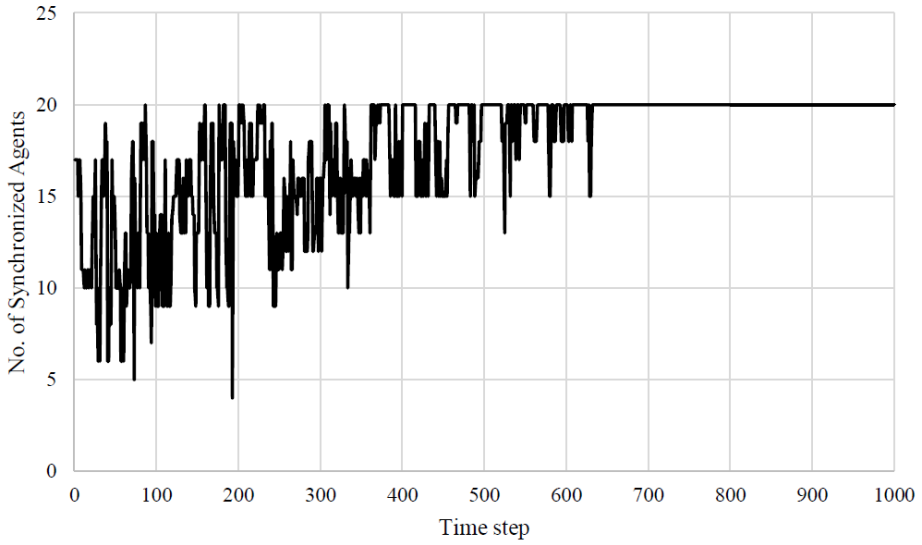

Figure 5.5: Population of agents synchronized with environment for ABM-120.

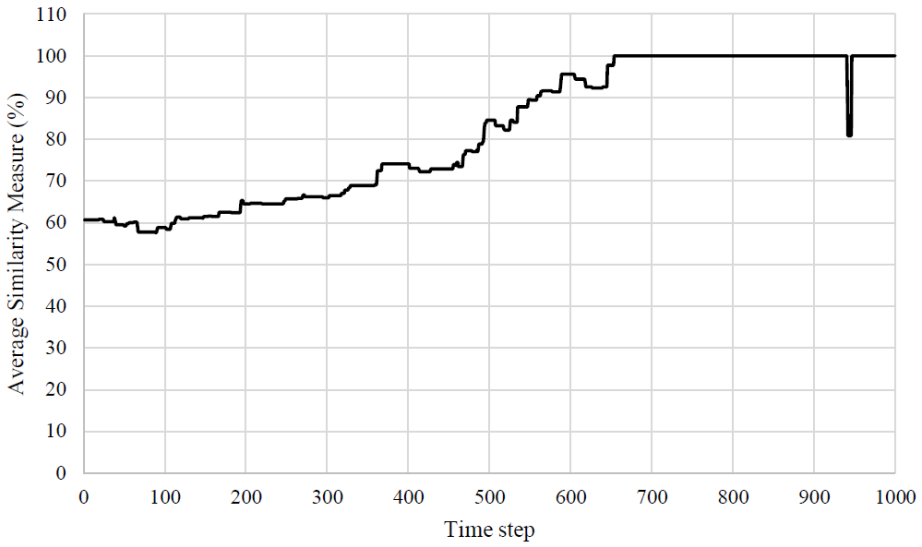

Figure 5.6: Average similarity measure of agents for ABM-120.

their behavior.

From the above results, it can be concluded that the selection of memory length affects the learning rate of agents, and depends upon the application environment being modeled. For a dynamic environment, agents with smaller memory size will show a good learning rate. For a stable environment, larger memory size for agents will offer better stability in their behavior. But, a memory size not too small and not too large, should offer acceptable results in all situations.

5.1.2 Effect of SMT on Agents' Synchronization Level

All the simulations, i.e. ABM-60, ABM-170 and ABM-120, were run for two different values of SMT, specifically 100% and 80%. However, the results for SMT as 100% are already shown in the previous subsection. Therefore, this subsection presents the results obtained by setting SMT as 80%, which is a more realistic value considering the excessive use of resources incurred in achieving complete synchronization with the environment (refer discussion in section 3.1.2). The simulation results obtained by running ABM-60, ABM-170 and ABM-120 are shown in Figures 5.7 and 5.8, Figures 5.9 and 5.10, Figures 5.11 and 5.12 respectively, along with a discussion on each of them.

With SMT as 80%, the agents will stop updating their memory once their similarity measure value either reaches or crosses this value. Each agent's local knowledge will only be a partial reflection of the global knowledge and therefore, in every time step the predicted outcome of only some of the agents will be correct. Following this, the population of synchronized agents will continuously vary over time. Here, the range in which the number of synchronized agents varies is of prime concern, as it reflects the overall synchro-

nization level of agents. The wider this range, the less synchronized and more unstable the agents will be and the narrower this range, the more stable and hence more synchronized the agents will be. The similarity measure of agents generally crosses the threshold value before their learning process is stopped. Therefore, the average similarity measure value, in the results shown below, tends to be higher than the threshold value.

ABM-60:

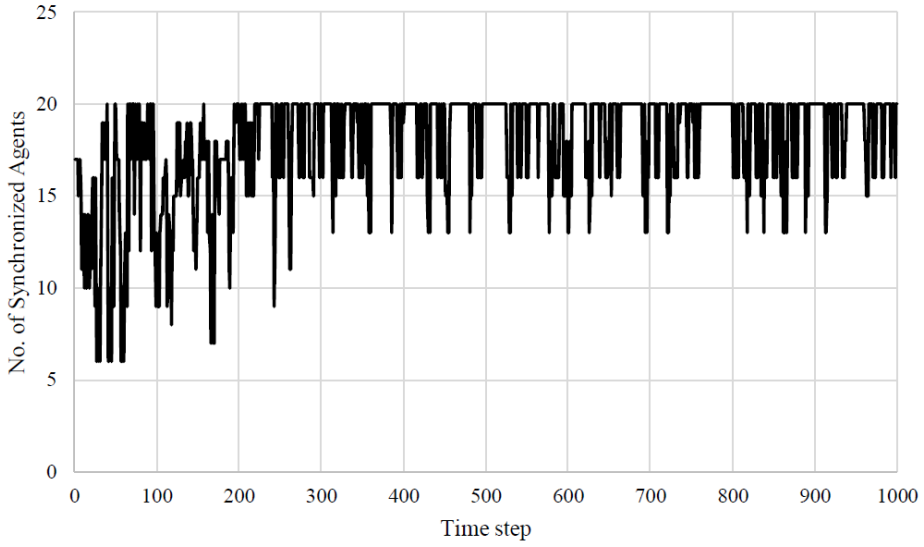

Figure 5.7: Population of agents synchronized with environment for ABM-60.

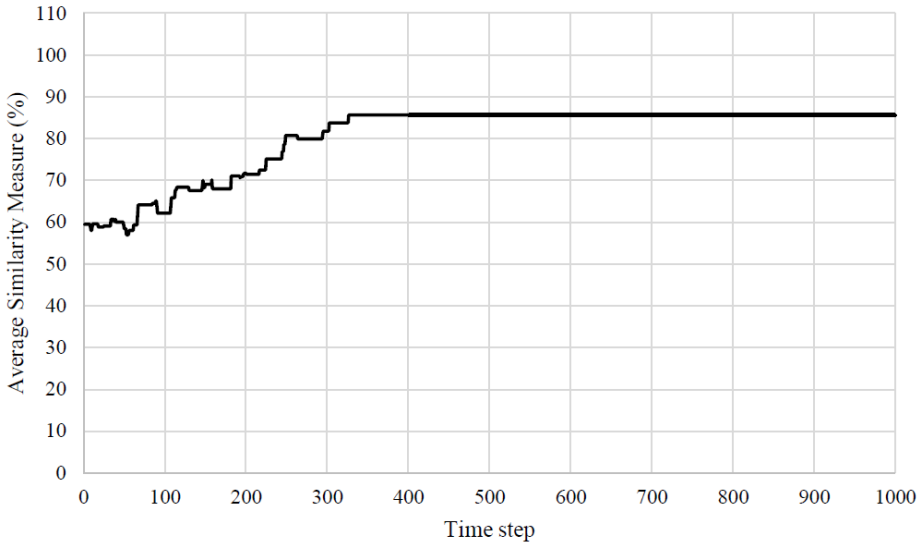

Figure 5.8: Average similarity measure of agents for ABM-60.

With ABM-60, the number of synchronized agents varied between 16 and 20 for a major part of the simulation, with the number dropping down to 13 several times. Such high variations depict unstable behavior of the agents due to shorter memory length and the steep curve for average similarity measure again highlights fast learning among them.

ABM-170:

The results for ABM-170 show that after approximately 480 time steps, the variation in number of synchronized agents got restricted to a range of 13 to 20 and further learning, after nearly 660 time steps, restricted the range to 17 to 20. This again highlights that even though the learning is slow, agents are able to retain more of their past knowledge and a higher number of agents are in synchronization than ABM-60. The slope of average similarity measure curve again depicts a slower learning process in agents with high

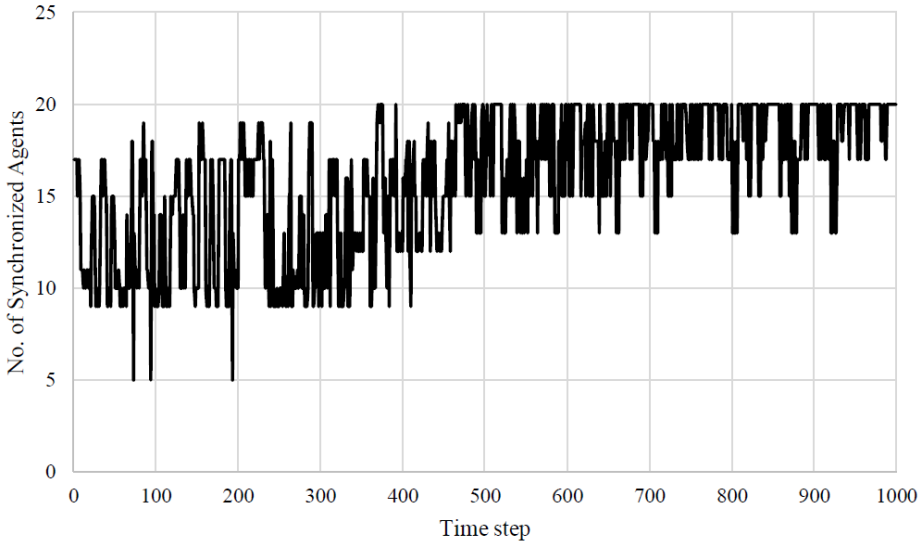

Figure 5.9: Population of agents synchronized with environment for ABM-170.

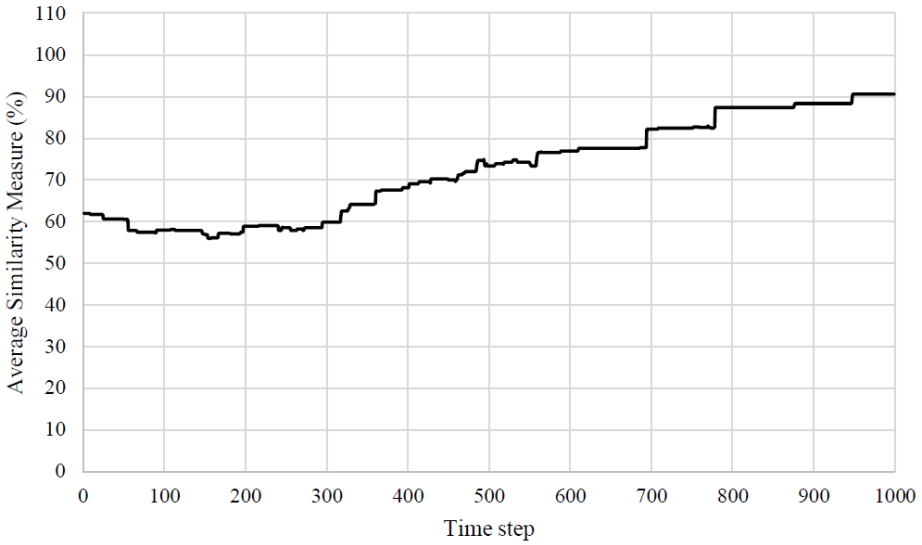

Figure 5.10: Average similarity measure of agents for ABM-170.

memory storage.

ABM-120:

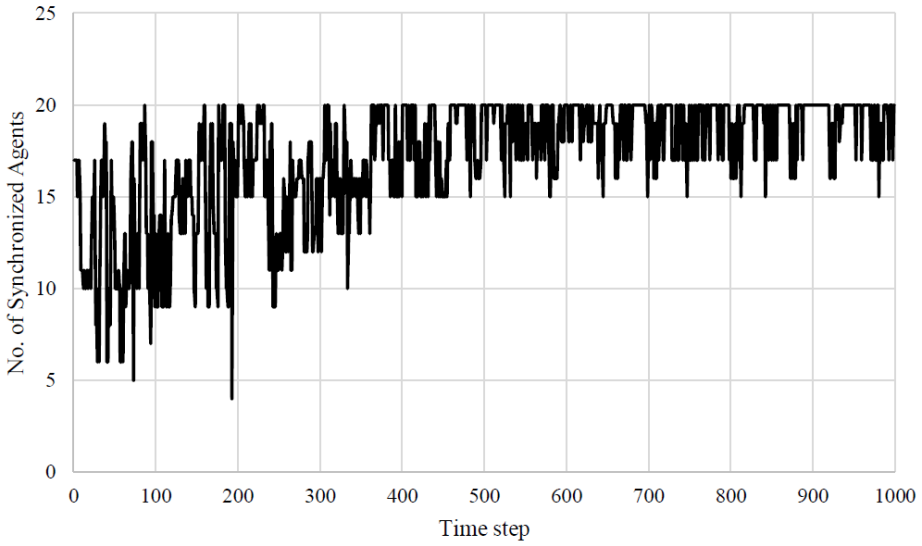

Figure 5.11: Population of agents synchronized with environment for ABM-120.

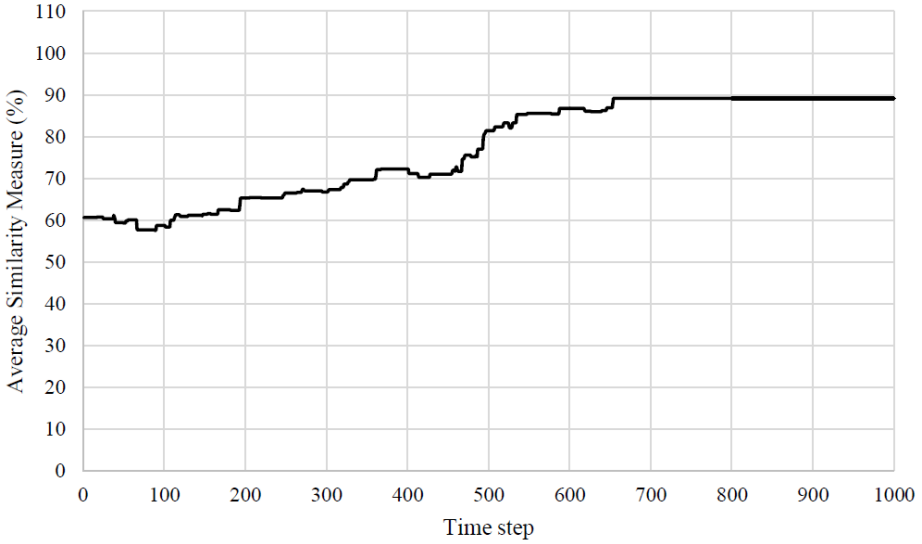

Figure 5.12: Average similarity measure of agents for ABM-120.

With ABM-120, the variation in the number of synchronized agents acquired a range of 15 to 20 almost after 370 time steps and this range further got restricted to 17 to 20 for the major part of the remaining simulation, with lower bound of the range falling down to 15 a few times. This indicates a stable behavior of the agents with learning rate, as indicated by the slope of average similarity measure curve, lying in between that of ABM-60 and ABM-170.

From the above results, it can be concluded that the SMT value controls the synchronization level of an agent with the environment. Further, its value also depends upon the application environment being modeled. For a frequently changing environment, a smaller value of SMT ensures less burden on the computational resources. For a more stable environment, a large value of SMT enables agents to achieve higher level of synchronization with the environment.

5.1.3 Validating the proposed method

This subsection attempts to validate the proposed framework by using prediction accuracy as a measure for determining the reliability each of the above three ABMs, in terms of outcomes predicted by them for a test dataset. The prediction accuracy is defined as the ratio of correct predictions over the sum of correct and incorrect predictions, measured for a particular outcome class (Olson and Delen, 2008). Within this test, the method of 10-fold cross validation was used on the *generated data* of 1000 records, to calculate an average value of prediction accuracy for each ABM at two different time instants, i.e. at the beginning of the simulation (before synchronization) and after the ABM was synchronized with the environment. The value of SMT was set to 80% during the test.

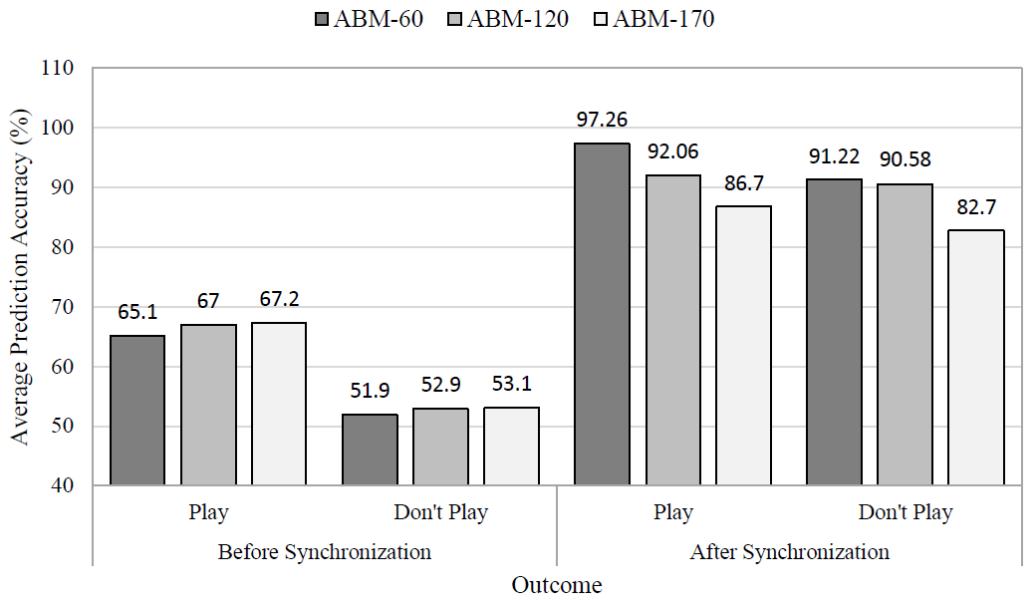

Figure 5.13: Average Prediction Accuracy for all three ABMs.

To be more specific, the value of average prediction accuracy for an ABM, in this case, depicts the accuracy with which it can correctly predict the outcome for an unseen set of weather conditions. In every calculation, 500 records were used as the training data (i.e. *global knowledge* or environment) for synchronizing the agents and the remaining records were used as the test data for predicting their outcomes.

As shown in Figure 5.13, the prediction accuracy for all three ABMs increases after they are synchronized with the environment. This justifies their reliability in terms of predicting more accurate results for unseen situations. Moreover, after synchronization, the prediction accuracy of ABM-120 is close enough to that of ABM-60 and much greater than that of ABM-170. This outcome can be verified by referring Figures 5.7 and 5.8, Figures 5.9 and 5.10, Figures 5.11 and 5.12, where for the first 500 records, ABM-60 is the most synchronized with the environment with an approximate average similarity measure of 86%, ABM-170 is the least synchronized with an approximate average similarity mea-

sure of only 73% and the synchronization level of ABM-120 lies in between the two with an approximate average similarity measure of 81%. Hence, the results shown in Figure 5.13.

5.2 Studying the effects of realistic Model Parameters

This section intends to study the impact that some of the real life factors, described by acceptance rate, retention rate and social influence, could have on the learning rate of agents and the synchronization level achieved by them. Although we could not get access to any real world case study, however through the results presented in this section we aim to provide an approximation of the working of these parameters in real world. Thus, it gives a good idea regarding what to expect when applying the proposed methodology to a realistic situation, taking these parameters into account.

The simulation results in this section were obtained using different values for the parameters, acceptance rate, retention rate and social influence. Moreover, the *generated data* was considered as a continuous stream of data, getting updated incrementally, and consequently the decision tree representing the *global knowledge* was also updated constantly. Further, the section considers only the average similarity measure plot, shown in Figure 5.2 graph type, as the basis for analyzing the effect of these parameters. This is because the agents' average learning rate and average level of synchronization achieved are both reflected by this graph. In addition, the memory length and SMT values were set to 120 and 80%, respectively. Once again, a population of 20 agents was considered for all the experiments in this section, except for the social influence parameter where a population of 50 agents was used. The following subsections describe in detail the results obtained for each of these parameters.

5.2.1 Effect of Retention Rate

The simulation ABM-120 was run for three different values of retention rate, specifically 70, 80 and 90. The corresponding results obtained are referred to as RR-70, RR-80 and RR-90, respectively and are plotted together in a single graph, shown in Figure 5.14, for an easy comparative analysis. In all the simulation results shown in Figure 5.14, the acceptance rate was kept constant at 100%.

From the definition of retention rate, higher values for this parameter should make an agent less likely to accept the new information, sticking firmly to its existing knowledge, and vice-versa. Similar behavior is portrayed by the results shown in Figure 5.14, where the average similarity measure for agents in RR-70 is the earliest to cross the SMT value and for RR-90 it never reaches the SMT value, thus requiring more records. On the other hand, the performance of average similarity measure for RR-80 lies approximately in between the two. Following this, an important observation can be made here, i.e. the agents initialized with smaller value of retention rate require lesser number of records to reach the SMT, as compared to the agents with higher values of retention rate. Specifically, RR-70 required only 300 records to achieve desired synchronization level, RR-80 required approximately 455 records and RR-90 never reached the SMT. Thus, the results go well with our hypothesis, depicting the obvious, i.e. agents with higher retention rate are more likely to ignore the new information as compared to their counterparts. In addition, the average learning rate reflected by the slope of each plot also follows the similar trend, where it is highest for agents in RR-70, lowest for agents in RR-90 and follows a decent rate in RR-80, again lying somewhere in between the previous two.

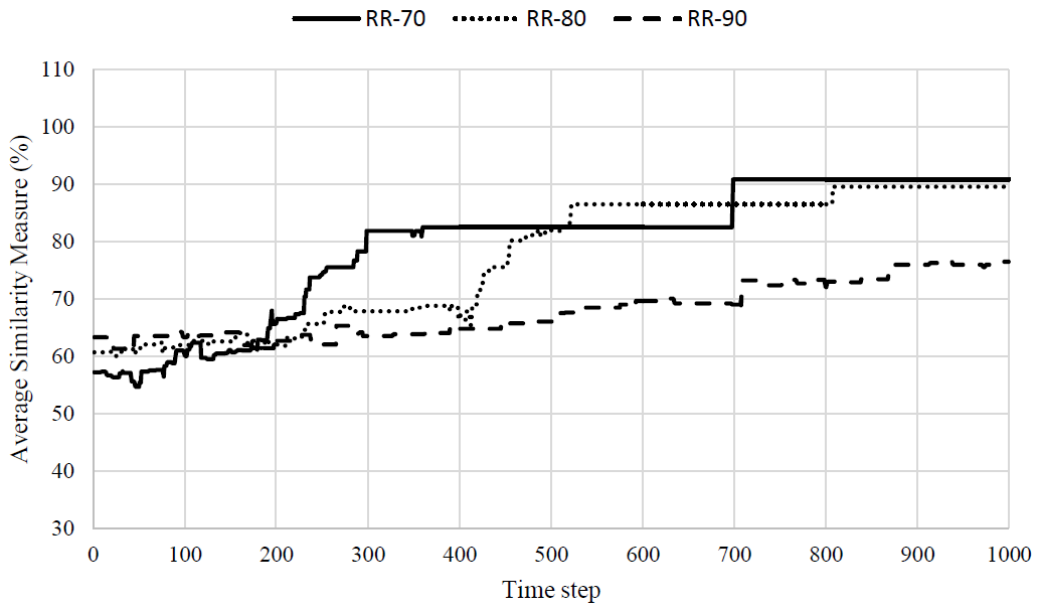

Figure 5.14: Average similarity measure of agents for different values of retention rate for ABM-120.

5.2.2 Effect of Acceptance Rate

This subsection presents the simulation results obtained by running ABM-120 for three different values of the acceptance rate. The results obtained for these values, i.e. 60, 80 and 100 are referred to as AR-60, AR-80 and AR-100, respectively, and are shown in Figure 5.15. In this case, for all three simulation runs the value of retention rate was opted as 80%, considering it to be a trade-off between the other two values.

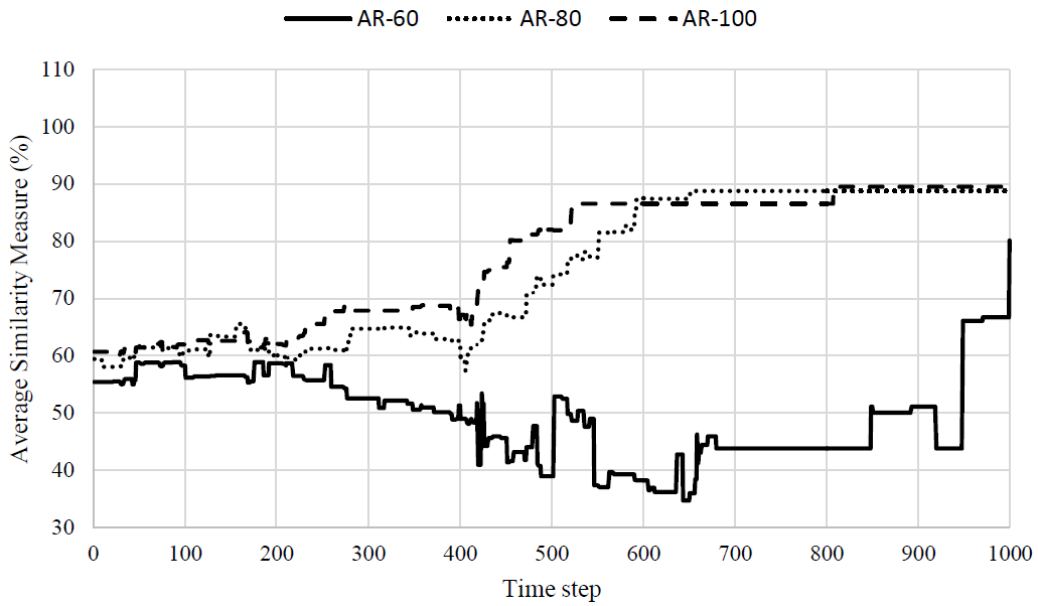

Figure 5.15: Average similarity measure of agents for different values of acceptance rate for ABM-120.

Following the discussion on acceptance rate, it is expected that an agent with a higher value for this parameter will learn the new information with least possible error and an agent with lesser value is more prone to making errors while doing the same. The results shown in Figure 5.15 follow a similar trend, where AR-100 agents are the fastest to reach the SMT achieving a maximum average similarity measure of 89% and AR-60 agents are the slowest ones that just manage to reach the SMT. The performance of AR-80 agents, though lies in between the two, but is closer to AR-100 achieving a maximum average similarity measure of 88%. Although the results for AR-80 and AR-100 closely follow each other, however it can still be concluded that the agents with high acceptance rate tend to achieve a higher level of synchronization with the environment, owing to learning incoming information with more accuracy.

The result for AR-60 shows deterioration in the average similarity measure for the first 600 records. The reason for such a behavior is that initially all agents possess a knowledge that is different from the global knowledge, and despite their effort to update their knowledge in every time step, they end up learning incorrect information in 40% of the cases. However, once the agents have gone through sufficient number of records, they start making an improvement towards their individual synchronization level and consequently the overall average similarity measure also starts improving. Thus, the results justify our hypothesis that higher the value of acceptance rate in agents, the more accurately they tend to learn the new information and require less time or data records to achieve desired level of synchronization with the environment. Also, a higher level of synchronization, marked by high value for average similarity measure, is achieved by these agents. Further, the slopes of each plot in Figure 5.15 clearly indicate that the AR-100 agents show highest learning rate, closely followed by the learning rate of AR-80 agents. However, AR-60 agents show a very poor learning rate in contrast to the other two.

5.2.3 Effect of Social Influence

The aim of this subsection is to study the impact of knowledge sharing among the agents, considering the parameters retention rate and acceptance rate. The value for these parameters was set to 80%, as it depicts the trade-off between the other values considered for them. Also, the selected values, considering the results in Figure 5.14 and Figure 5.15, tend to provide better and more realistic results as compared to the other values. For this case specifically, a population of 50 agents was considered since the number 20 was too small to setup any neighborhood network amongst the agents. Each agent was linked to its 3 closest neighbors based on their respective locations, using the *k-nearest neighbor*

algorithm. Altogether, the ABM-120 simulation was run twice, with and without the social influence and the results obtained are referred to as SI-Enabled (SI-E) and SI-Disabled (SI-D), respectively.

The main purpose of incorporating social influence and hence the social interactions was to enable knowledge sharing among the agents, with the expectation that the agents will learn faster requiring lesser number of records to attain the desired synchronization level, as compared to when they are not socially active. However, the results shown in Figure 5.16 contradict with this hypothesis. In the results, the agents achieve the SMT value faster in the case when no social interactions exist, pointing towards the negative effect of social influence on the agents' learning. This indicates that the knowledge shared by the agents in a social network may not always be correct. The reason for such a behavior is that the knowledge of agents in the beginning was different from the *global knowledge*, and under such situation allowing them to share knowledge would lead to the spread of incorrect knowledge. Due to this, it requires more time/data records for an agent to acquire enough information regarding the change. Thus, despite learning from the *global agent* and through social interactions, the overall time taken by the average similarity measure to reach SMT is increased noticeably. Therefore, it can be inferred that social influence will have a positive effect on the working of agents, only when considerable number of agents possess the correct knowledge.

Regarding the learning rate of agents in SI-E and SI-D, no clear distinction can be made between the two as they closely follow each other. For the first 400 records, the learning rate of SI-D agents is less than that of SI-E agents, but this situation reverses following just a few time steps after that, with a sharp increase in the slope of SI-D, marking

a substantial increase in its learning rate. On the other hand, the slope of SI-E does not change much depicting a consistent learning rate overall. Further, it can be stated that the agents involved in social interactions, though tend to learn slowly, but are steadier with their progress in comparison to their counterpart.

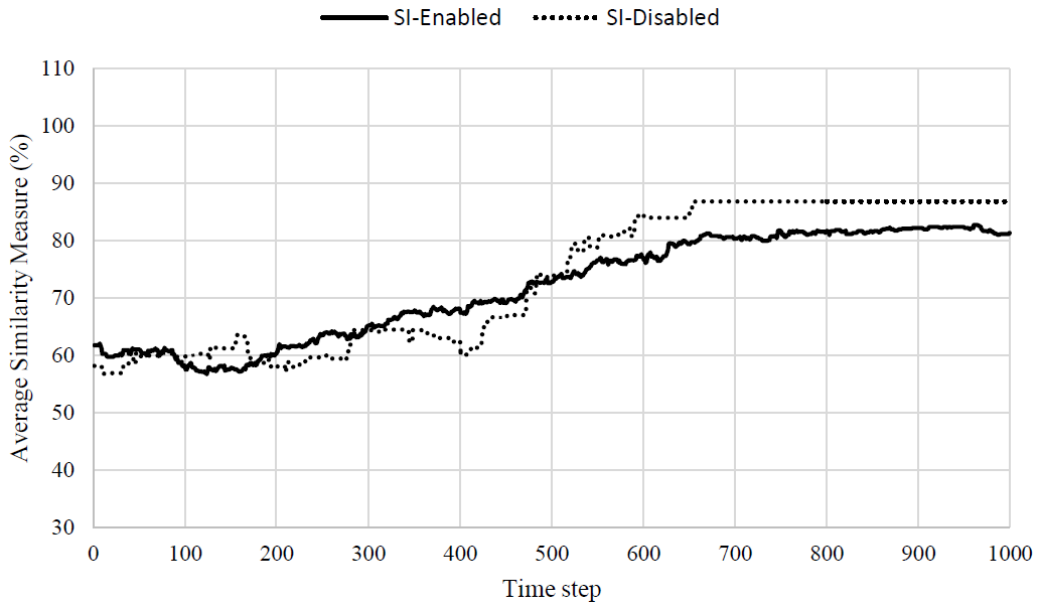

Figure 5.16: Average similarity measure of agents with and without social influence for ABM-120.

Chapter 6

Conclusion and Future Work

In this thesis we introduced an approach to handle the issue of maintaining the validity and reliability of an ABM, measured in terms of prediction accuracy, to a great extent considering the dynamics of the environment. Adding to this, the methodology also takes care of improving the overall prediction accuracy of the ABM, following a change in the simulation environment. Two important aspects were identified related to our goal in this research. The first one deals with incorporating some kind of intelligence in agents to make them capable of inducing learning from their past experiences. The second one deals with detecting any changes occurring in the simulation environment, followed by determining an agent's synchronization level with it. These aspects have been covered in detail in chapter 3, along with a flowchart describing the overall working of the approach. Corresponding to the first aspect, the agents were assigned a definite memory length to store their experiences and empowered with C4.5 decision tree learning algorithm to learn from them. For the second feature, an integrated algorithm is presented to measure the similarity between two decision trees, combining both the semantic and structural factors related to it. The working of this algorithm assumed that the knowledge possessed by an agent and the one reflected

by the environment are both represented in terms of decision trees. Further, the working of the proposed approach is illustrated using the case study of *Golf play*, the details for which, including the reasons behind its selection, are mentioned in chapter 4.

Through the experimental results, we intend to study the effect of various factors/parameters on the learning rate and the synchronization level achieved by an agent. Specifically, the learning rate refers to the rate at which an agent can adapt to environmental changes and the synchronization level indicates how similar its knowledge is to that represented by the environment (i.e. *global knowledge*). The analysis of these parameters also helps demonstrate the working of proposed framework under different settings.

The experimental results for different values of memory length and SMT highlight that a shorter memory length ensured fast learning among the agents with poor stability in their synchronized status and a larger memory length though offered better stability, but required considerable time in adapting the environmental change. Thus, a trade-off was observed between learning rate and stable behavior, following which agents with memory length, not too short and not too large, exhibited a decent learning rate with an acceptable stability in their overall synchronized status. Further, a higher value of SMT enabled agents to achieve high level of synchronization with the environment and vice-versa. In addition, the selection of values for memory length and SMT was found to depend upon the application environment being modeled, since it was analyzed that for a frequently changing environment a smaller value for memory length and SMT will incur less burden on the computational resources. The results for average prediction accuracy point towards the effectiveness of our approach in making an ABM adaptable, thus enabling it to evolve with the changing environment. Here, all three ABMs, ABM-60, ABM-120 and ABM-170,

managed to acquire high prediction accuracy, averaged on different test datasets, despite being initialized with a different prior knowledge at the beginning.

Further, the effects of some of the realistic parameters were also studied on the agents' learning rate and synchronization level. As expected, the experimental results show that for a lower value of retention rate and a higher value of acceptance rate, the agents exhibit a higher learning rate achieving a higher level of synchronization with the environment. However, for a value too high and too low for retention rate and acceptance rate, the agents require more records to reach SMT and show a significant deterioration in their learning rate, respectively. Thus, the results justify the hypothesis that lower retention and higher acceptance enable an agent to accept new information with more accuracy. The results for social influence, on the other hand, highlight the negative effect it can have on the overall learning rate and synchronization level of an ABM, when majority of agents possess incorrect knowledge. Instead of increasing the learning rate and achieving higher level of synchronization, as expected, the results portray an opposite behavior. Therefore, it can be inferred that for social influence to work effectively, a considerable number of agents should possess the correct knowledge.

Although the work presented in this thesis is a small contribution towards handling the issue of reliability and validation in ABMs under dynamic environment, nonetheless, we believe that it provides a significant step towards pursuing different research work in the future. Some of these are described below:

1. The current approach lacks the use of domain knowledge and solely depends upon the available dataset for training the agents. Although many hidden patterns can be revealed from this dataset, however the agents, when trained on it, will still lack the

knowledge possessed by the domain experts. Thus, incorporating domain knowledge into the current approach can improve the predictions made by agents, especially for the unseen situations.

2. The ABM based on current approach lacks the responsiveness, expected from a model, in order to make it closely emulate the real world changes. This is because the agents in the proposed approach learn a real world change much later after it has occurred, as their only source for learning these changes is the training dataset. Therefore, the proposed approach shows a lag in learning these changes and hence cannot be used to anticipate any changes beforehand. This gives another direction in which the current work can be extended, i.e. make an ABM predict the changes even before they occur in the real world. We believe that the background study described in section 2.3 will be useful in proceeding with this idea, as it discusses about the human way of making decisions in any given situation.

3. The current approach can also be used as the basis for developing a self-evolvable ABM, in which a portion of the input data is used for building the model and the remaining fraction is used as the test data to determine its performance. This way as more data comes in, the corresponding model evolves itself accordingly. Here, the proposed work can be analyzed for a faster implementation of such an ABM.

References

ABRAHAM, A. 2005. Artificial neural networks. *Handbook of measuring system design*, 901–908.

AHMED, S., KOBTI, Z., AND KENT, R. D. 2011. Predictive data mining driven architecture to guide car seat model parameter initialization. In *Intelligent Decision Technologies*. Springer, 789–797.

APTE, C. AND HONG, S. J. 1996. Predicting equity returns from securities data with minimal rule generation. In *Advances in knowledge discovery and data mining*, P. S. U. Fayyad, G. Piatetsky-Shapiro and R. Uthurusamy, Eds. American Association for Artificial Intelligence, Menlo Park, Calif, 541–560.

ARROYO, J., HASSAN, S., GUTIÉRREZ, C., AND PAVÓN, J. 2010. Re-thinking simulation: a methodological approach for the application of data mining in agent-based modelling. *Computational and Mathematical Organization Theory 16,* 4, 416–435.

BACHE, K. AND LICHMAN, M. 2013. UCI machine learning repository.

BANKS, J. 1998. *Handbook of Simulation*. John Wiley & Sons, Chichester.

BAQUEIRO, O., WANG, Y. J., MCBURNEY, P., AND COENEN, F. 2009. Integrating data mining and agent based modeling and simulation. In *Advances in Data Mining. Applications and Theoretical Aspects*. Springer, 220–231.

BHATT, S. AND RAO, P. 2008. Enhancements to the vantage firewall analyzer. Tech. rep., Technical Report HPL-2007-154R1, HP Laboratories.

BONABEAU, E. 2002. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America 99,* 3, 7280–7287.

BOSWELL, D. 2002. Introduction to support vector machines. Online: http://www.work.caltech.edu/boswell/IntroToSVM.pdf.

CHEESEMAN, P. AND STUTZ, J. 1996. Bayesian classification (autoclass): Theory and results. *Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, Cambridge, MA,*, 153–180.

CHI, R. H. AND KIANG, M. Y. 1991. An integrated approach of rule-based and case-based reasoning for decision support. In *Proceedings of the 19th annual conference on Computer Science*. ACM, 255–267.

COLLIER, N. 2003. Repast: An extensible framework for agent simulation. *The University of Chicagos Social Science Research 36*.

COLLIER, N., HOWE, T., AND NORTH, M. 2003. Onward and upward: The transition to repast 2.0. In *First Annual North American Association for Computational Social and Organizational Science Conference, Pittsburgh, PA, USA*.

CORTES, C. AND VAPNIK, V. 1995. Support-vector networks. *Machine learning 20,* 3, 273–297.

COVER, T. AND HART, P. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory, 13,* 1 (January), 21–27.

D'INVERNO, M. AND LUCK, M. 2000. Sociological agents for effective social action. In *Proceedings of the Fourth International Conference on Multi-Agent Systems, 2000*.

IEEE Computer Society, 379–380.

DOGRA, I. S. AND KOBTI, Z. 2013. Improving prediction accuracy in agent based modeling systems under dynamic environment. In *IEEE Congress on Evolutionary Computation (CEC), 2013*. IEEE, 2114–2121.

DORAN, J. 1997. From computer simulation to artificial societies. *Transactions of the Society for Computer Simulation International 14,* 2, 69–77.

DORAN, J. 1998. Social simulation, agents and artificial societies. In *Proceedings of International Conference on Multi Agent Systems, 1998.* IEEE, 4–5.

DROGOUL, A., VANBERGUE, D., AND MEURISSE, T. 2003. Multi-agent based simulation: Where are the agents? In *Multi-agent-based simulation II*. Springer, 1–15.

EMELE, C. D., NORMAN, T. J., ŞENSOY, M., AND PARSONS, S. 2012. Exploiting domain knowledge in making delegation decisions. In *Agents and Data Mining Interaction*. Springer Berlin Heidelberg, 117–131.

ENO, J. AND THOMPSON, C. W. 2008. Generating synthetic data to match data mining patterns. *Internet Computing, IEEE 12,* 3, 78–82.

FAYYAD, U., PIATETSKY-SHAPIRO, G., AND SMYTH, P. 1996. From data mining to knowledge discovery in databases. *AI magazine 17,* 3, 37.

FAYYAD, U. M., DJORGOVSKI, S. G., AND WEIR, N. 1996. From digitized images to online catalogs: data mining a sky survey. *AI magazine 17,* 2, 51–66.

FISHWICK, P. A. 1995. Simulation model design. In *Proceedings of the 27th conference on Winter simulation*. IEEE Computer Society, 209–211.

FISHWICK, P. A. 1997. Computer simulation: growth through extension. *Transactions of the Society for Computer Simulation 14,* 1, 13–24.

GARCIA, R. 2005. Uses of agent-based modeling in innovation/new product development

research*. *Journal of Product Innovation Management 22,* 5, 380–398.

GILBERT, N. AND TROITZSCH, K. 1999. *Simulation for the Social Scientist*. Open University Press.

GOLDING, A. R. AND ROSENBLOOM, P. S. 1996. Improving accuracy by combining rule-based and case-based reasoning. *Artificial Intelligence 87,* 1, 215–254.

GOSTOLI, U. 2008. A cognitively founded model of the social emergence of lexicon. *Journal of Artificial Societies and Social Simulation 11,* 1, 2.

HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., AND WITTEN, I. H. 2009. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter 11,* 1, 10–18.

HARDING, A. 1999. Modeling techniques for examining the impact of population ageing on social expenditure. In *Conference on the Policy implications of the Ageing of Australia's Population, Melbourne, NATSEM*.

HEGLAND, M. 2004. Data mining techniques. *Acta Numerica*, 313–355.

JENNINGS, N. R. 2000. On agent-based software engineering. *Artificial intelligence 117,* 2, 277–296.

JENNINGS, N. R. AND BUSSMANN, S. 2003. Agent-based control systems: Why are they suited to engineering complex systems? *IEEE control systems 23,* 3, 61–73.

KANTARDZIC, M. 2011. *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons.

KLIMT, B. AND YANG, Y. 2004. The enron corpus: A new dataset for email classification research. In *Machine learning: ECML 2004*. Springer, 217–226.

KOBTI, Z., SNOWDON, A. W., RAHAMAN, S., DUNLOP, T., AND KENT, R. D. 2006. A cultural algorithm to guide driver learning in applying child vehicle safety restraint.

In *IEEE Congress on Evolutionary Computation, 2006. CEC 2006.* IEEE, 1111–1118.

KUMAR, K. A., SINGH, Y., AND SANYAL, S. 2009. Hybrid approach using case-based reasoning and rule-based reasoning for domain independent clinical decision support in icu. *Expert Systems with Applications 36,* 1, 65–71.

LEE, G. H. 2008. Rule-based and case-based reasoning approach for internal audit of bank. *Knowledge-Based Systems 21,* 2, 140–147.

LIU, B. AND HSU, W. 1996. Post-analysis of learned rules. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence AAAI-96, AAAI Press/MIT Press, Menlo Park, CA*. Vol. 1. Citeseer, 828–834.

MACAL, C. M. AND NORTH, M. J. 2005. Tutorial on agent-based modeling and simulation. In *Proceedings of the 37th conference on Winter simulation*. WSC '05. Winter Simulation Conference, 2–15.

MCCALLUM, A., NIGAM, K., ET AL. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*. Vol. 752. Citeseer, 41–48.

MELLOULI, S., MOULIN, B., AND MINEAU, G. 2004. Laying down the foundations of an agent modelling methodology for fault-tolerant multi-agent systems. In *Engineering Societies in the Agents World IV*. Springer, 275–293.

MITCHELL, T. M. 1999. Machine learning and data mining. *Communications of the ACM 42,* 11, 30–36.

MYLES, A. J., FEUDALE, R. N., LIU, Y., WOODY, N. A., AND BROWN, S. D. 2004. An introduction to decision tree modeling. *Journal of chemometrics 18,* 6, 275–285.

OLSON, D. L. AND DELEN, D. 2008. *Advanced data mining techniques*. Springer Berlin

Heidelberg.

ORCUTT, G. H. 1957. A new type of socio-economic system. *The Review of Economics and Statistics 39,* 2, 116–123.

PALMER, R., BRIAN ARTHUR, W., HOLLAND, J. H., LEBARON, B., AND TAYLER, P. 1994. Artificial economic life: a simple model of a stockmarket. *Physical D: Nonlinear Phenomena 75,* 1, 264–274.

PEKERSKAYA, I., PEI, J., AND WANG, K. 2006. Mining changing regions from access-constrained snapshots: a cluster-embedded decision tree approach. *Journal of Intelligent Information Systems 27,* 3, 215–242.

PERNER, P. 2011. How to interpret decision trees? In *Advances in Data Mining. Applications and Theoretical Aspects*. Springer, 40–55.

PRENTZAS, J. AND HATZILYGEROUDIS, I. 2007. Categorizing approaches combining rule-based and case-based reasoning. *Expert Systems 24,* 2, 97–122.

QUINLAN, J. R. 1986. Induction of decision trees. *Machine learning 1,* 1, 81–106.

QUINLAN, J. R. 1987. Generating production rules from decision trees. In *IJCAI*. Vol. 87. Citeseer, 304–307.

QUINLAN, J. R. 1993. *C4.5: programs for machine learning*. Vol. 1. San Francisco: Morgan kaufmann.

REMONDINO, M. 2003. Agent based process simulation and metaphors based approach for enterprise and social modeling. *Agent-Based Simulation 4th Proceedings, SCS Europe 4*, 93–97.

REMONDINO, M. AND CORRENDO, G. 2005. Data mining applied to agent based simulation. In *Proceedings of the 19th European Conference on Modelling and Simulation, Riga, Latvia*.

REMONDINO, M. AND CORRENDO, G. 2006. Mabs validation through repeated execution and data mining analysis. *International Journal of Simulation Systems, Science & Technology 7,* 6, 10–21.

ROBERTSON, D. A. 2003. Agent-based models of a banking network as an example of a turbulent environment: the deliberate vs. emergent strategy debate revisited. *Emergence 5,* 2, 56–71.

ROSSILLE, D., LAURENT, J.-F., AND BURGUN, A. 2005. Modelling a decision-support system for oncology using rule-based and case-based reasoning methodologies. *International journal of medical informatics 74,* 2, 299–306.

SERPEN, G. AND SABHNANI, M. 2006. Measuring similarity in feature space of knowledge entailed by two separate rule sets. *Knowledge-Based Systems 19,* 1, 67–76.

SHI, W. AND BARNDEN, J. A. 2005. How to combine cbr and rbr for diagnosing multiple medical disorder cases. In *Case-Based Reasoning Research and Development*. Springer Berlin Heidelberg, 477–491.

SONG, A., PADGHAM, L., AND CAVENDON, L. 2007. Prediction in dynamic environment: Robocup rescue exploration.

SYMEONIDIS, A. L., CHATZIDIMITRIOU, K. C., ATHANASIADIS, I. N., AND MITKAS, P. A. 2007. Data mining for agent reasoning: A synergy for training intelligent agents. *Engineering Applications of Artificial Intelligence 20,* 8, 1097–1111.

TADOKORO, S., KITANO, H., TAKAHASHI, T., NODA, I., MATSUBARA, H., SHINJOH, A., KOTO, T., TAKEUCHI, I., TAKAHASHI, H., MATSUNO, F., ET AL. 2000. The robocup-rescue project: A robotic approach to the disaster mitigation problem. In *Proceedings of the IEEE International Conference on Robotics and Automation, 2000. ICRA'00.* Vol. 4. IEEE, 4089–4094.

TROITZSCH, K. G. 1997. Social science simulation-origins, prospects, purposes. *Lecture Notes in Economics and Mathematical Systems*, 41–54.

WEISS, S. M. AND KULIKOWSKI, C. A. 1990. Computer systems that learn: Classification and prediction methods from statistics, neural networks, machine learning and expert systems.

WOOLDRIDGE, M. 2002. *An Introduction to MultiAgent Systems*. John Wiley & Sons, Chichester.

WOOLDRIDGE, M., JENNINGS, N. R., ET AL. 1995. Intelligent agents: Theory and practice. *Knowledge engineering review 10,* 2, 115–152.

YAMADA, R., NAKAJIMA, H., BRAVE, S. B., MALDONADO, H., LEE, J.-E., NASS, C., AND MORISHIMA, Y. 2006. An implementation of socially-intelligent agents providing emotional support and its application. In *IEEE International Conference on Systems, Man and Cybernetics, 2006. ICSMC'06.* Vol. 1. IEEE, 322–326.

ZHANG, K. AND FAN, W. 2008. Forecasting skewed biased stochastic ozone days: analyses, solutions and beyond. *Knowledge and Information Systems 14,* 3, 299–326.

All the images used in the Figures were referred from Google image library, and the decision trees shown were drawn using the open source graph visualization software, Graphviz [http://www.graphviz.org/].

Appendix A

Copyright Permission

In order to obtain permission to use our previous publication, we sent an email to the General Chair of the IEEE CEC 2013 conference, Carlos A. Coello Coello (ccoello@cs.cinvestav.mx), and the IEEE Intellectual Property Rights Office (copyrights@ieee.org). The reply from both the parties directed us to obtain the requested permission online using RightsLink®, a web-based permission service for which IEEE has partnered with Copyright Clearance Center. Further details about obtaining the permission can be referred from `http://www.ieee.org/publications_standards/publications/rights/index.html`.

As a requirement by IEEE for using our previously published work, we include the following reference. In addition, a screenshot for the permission obtained is provided on the next page.

©2013 IEEE. Reprinted, with permission, from Inderjeet Singh Dogra and Ziad Kobti, *"Improving prediction accuracy in agent based modeling systems under dynamic environment"*, IEEE Congress on Evolutionary Computation (CEC), June-2013.

Vita Auctoris

Inderjeet Singh Dogra was born in 1987 in New Delhi, India. He received his Bachelors degree from Guru Gobind Singh Indraprastha University in Information Technology in 2009. Currently he is a Master's Candidate under Dr. Kobti's supervision at the School of Computer Science in University of Windsor, Ontario, Canada. He is expected to graduate in December 2013.