Designing Robust Trust Establishment Models with a Generalized Architecture and a Cluster-based Improvement Methodology

Julian Templeton

A thesis submitted to the Faculty of Engineering
in partial fulfillment of the requirements for the degree of

MASTER OF COMPUTER SCIENCE

in Applied Artificial Intelligence

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

In Multi-Agent Systems consisting of intelligent agents that interact with one another, where the agents are software entities which represent individuals or organizations, it is important for the agents to be equipped with trust evaluation models which allow the agents to evaluate the trustworthiness of other agents when dishonest agents may exist in an environment. Evaluating trust allows agents to find and select reliable interaction partners in an environment. Thus, the cost incurred by an agent for establishing trust in an environment can be compensated if this improved trustworthiness leads to an increased number of profitable transactions. Therefore, it is equally important to design effective trust establishment models which allow an agent to generate trust among other agents in an environment. This thesis focuses on providing improvements to the designs of existing and future trust establishment models.

Robust trust establishment models, such as the Integrated Trust Establishment (ITE) model, may use dynamically updated variables to adjust the predicted importance of a task's criteria for specific trustors. This thesis proposes a cluster-based approach to update these dynamic variables more accurately to achieve improved trust establishment performance. Rather than sharing these dynamic variables globally, a model can learn to adjust a trustee's behaviours more accurately to trustor needs by storing the variables locally for each trustor and by updating groups of these variables together by using data from a corresponding group of similar trustors.

This work also presents a generalized trust establishment model architecture to help models be easier to design and be more modular. This architecture introduces a new transaction-level preprocessing module to help improve a model's performance and defines a trustor-level postprocessing module to encapsulate the designs of existing models. The preprocessing module allows a model to fine-tune the resources that an agent will provide during a transaction before it occurs. A trust establishment model, named the Generalized Trust Establishment Model (GTEM), is designed to showcase the benefits of using the preprocessing module.

Simulated comparisons between a cluster-based version of ITE and ITE indicate that the cluster-based approach helps trustees better meet the expectations of trustors while minimizing the cost of doing so. Comparing GTEM to itself without the preprocessing module and to two existing models in simulated tests exhibits that the preprocessing module improves a trustee's trustworthiness and better meets trustor desires at a faster rate than without using preprocessing.

Acknowledgements

Throughout my studies, I have been lucky and grateful for the support given by many. I want to firstly thank my supervisor, Dr. Thomas Tran, for his support throughout the duration of my studies. He has always provided encouragement for my work and has provided insight to greatly help me improve my research abilities. Through his guidance and from the many other professors whom I have had the pleasure of learning from, I have been able to gain skills that will help me throughout my future.

I also want to thank my mother Micheline, my father Scott, my brother Nathan, and the rest of my family for their support and encouragement throughout my life and studies. They have always helped motivate me to try my hardest and produce high quality work. Finally, I want to thank my friends who have always provided me with their encouragement and support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

In the field of artificial intelligence, there have been significant research advancements regarding Multi-Agent Systems (MASs) and the agents which reside within these MASs. The agents that are being researched have become more autonomous in their decision-making abilities using various modules which have been increasing in complexity, allowing these software entities to be defined as intelligent agents [1]. This has led to the increase of MAS-related research, as discussed in [2, 3]. Since these autonomous agents collaborate with each other in singular or distributed MASs, they are becoming increasingly concerned with the trustworthiness of other agents. An untrustworthy agent may take advantage of other agents in the environment, leading to poor collaboration within the environment. This can lead to real-world costs if the agents are selling and buying products within the MASs. The concept of trust is not new in the field of Computer Science and there are many different types of trust research [4]. When specifically focused on MAS-related trust research, surveys such as [5] exhibit that there has been significant research regarding how agents can evaluate the trustworthiness of other agents, but there has not been as substantial of an amount of research regarding how agents can work to improve their trust within the environment.

When discussing the different types of agents that reside within the environment, two types will be discussed throughout this thesis. Trustors are agents that request/consume services from other agents in the environment. Consequently, the agents that provide services to the

trustors are referred to as trustees. Agents can be both a trustor and a trustee at any given time step in an environment. Trustors utilize a trust evaluation model to help calculate trust values to represent the trustworthiness of trustees. These trust values can help a trustor with selecting which trustees should be selected as interaction partners when the trustor desires one or more tasks to be performed. To become more trustworthy in environments and to receive more interactions from trustors, trustees utilize a trust establishment model to understand how they should modify their behaviours towards specific trustors in the environments.

Both the trust evaluation model and the trust establishment model are part of a proposed trust management module that is utilized by intelligent agents. This trust management module contains the core decision-making components to help intelligent agents make better decisions and to help with the communication between agents [6]. In [6], trust establishment is proposed as a crucial component of trust management since it allows an agent to become more trustworthy, and hence more prevalent, in MASs. Thus, agents equipped with this trust management module will be able to evaluate and improve trust in an environment by using the functions of the various components in the module, where these components can differ between agents. Although not the topic of the thesis, it is important to note that the trust management module allows agents to directly communicate with each other in potentially distributed MASs via solutions such as [7]. Since the agents and the MASs that they reside within can vary depending on their implementations, depending on the properties of the environment that they reside in, and depending on the domains in which they are operating, it is important to appropriately select which trust evaluation and trust establishment models should be used for a specific environment. Two examples of domains where agents use the trust management components are e-commerce environments [8] and smart grid environments [9].

Despite there not being significant research regarding trust establishment models, there are several different models that have been proposed. These models attempt to increase the overall trust of trustees and are generally designed to manage the amount of Utility Gain (UG) that is provided to the trustors. The UG is the amount of resources that the trustee spends on the transaction with the trustor. Thus, effective trust establishment models balance the trade-off between the UG being provided to trustors and the trustee's trust among trustors in the environment.

This chapter will describe the work performed throughout this thesis to improve the current state of trust establishment model research and will present the overall structure of the thesis. The work that will be presented includes a cluster-based method to improve the performance of specific trust establishment models and a generalized trust establishment model architecture which will help with the design and performance of trust establishment models. This cluster-based technique and the generalized architecture will help formalize the trust establishment process and provide robust methods for balancing the various trade-offs that must be considered within the trust establishment process.

1.2 Motivations

As trust establishment model research has progressed, many new ideas have been incorporated into the proposed models to help improve their general performance. Robust models, such as the Integrated Trust Establishment (ITE) model [10], utilize both the direct and indirect feedback from trustors to be used and updated via Reinforcement Learning to dynamically improve the trustworthiness of a trustee to other trustors. One of ITE's important update components uses α and β values to specify the rates of improvement and disimprovement that the trustee needs to adjust itself by when performing a task for a trustor. These values are updated dynamically by ITE based on the direct feedback received from the trustor, or that is approximated by ITE, after the trustee performs the service.

Although ITE's use of these dynamically updated variables improves the model's performance, ITE's current method of updating the α and β values can be improved since it focuses on using the Rate of Change (RoC) of the direct feedback values that are obtained by the trustee from all transactions with trustors at a given point in time, where the direct feedback is the trustor satisfaction which is also referred to as SAT, against the SAT from all transactions with trustors from the previous point in time at which this aggregated SAT value has been computed. The full details on this approach will be thoroughly presented later in the thesis. Although this is a good representation of a trustor's current needs that has been proven to work well in simulations, when working within one or more MASs there will likely be many trustors,

potentially from different MASs, that will display similar behaviours to one another at one or more periods of time. Furthermore, each trustor may shift their individual behaviours differently depending on the trust evaluation model that they are using. This can result in the single RoC value not being representative of the changes to trustor behaviours at different time steps.

Since the trustors in the environment may be using different trust evaluation models, it is important to identify similarities between trustor behaviours at each time step to better model their needs and attitudes when updating the α and β values. By identifying these similarities within a trust establishment model which is using dynamic variables, such as ITE, the model will be able to use these groupings to update the dynamic variable values more accurately than performing global updates. This idea is incorporated through a cluster-based methodology and improves the general performance of ITE when utilized. The process is also generalizable to discrete-based trust establishment models to help improve the performance of existing and future trust establishment models that use the robust method of dynamically updating improvement and disimprovement rate variables that directly affect how a trustee's behaviour is to be updated.

With trust establishment models becoming increasingly complex, this can cause issues with the models not being modular enough to easily adapt to new research findings. This is because there is no formal structure in place that is generalizable enough for trust establishment models of varying designs to define their components within. This is also an issue with the practical use of the models since the implementation of complex trust establishment models can require a domain expert capable of correctly programming each of the model's components and of optimizing the potentially unmanageable amount of hyperparameters that are used to tune the model for a specific environment or trustee. Furthermore, when models are designed without a generalized architecture, the designs of the different models may differ drastically. This can cause certain models to only function within specific domains or environments, rather than being generalizable to various domains.

With a formal, generalizable architecture, existing and new models can be more formally structured to organize their inner components. Many, but not all, of the currently proposed trust establishment models build upon previously proposed models and utilize increasingly complex logic to improve their performance. Due to this, current trust establishment models share many similarities that can be categorized into modules of a generalized architecture. Thus, a robust

generalized trust establishment model architecture will be presented based on these found similarities which will address the above concerns while providing a framework for improving the performance of existing and future trust establishment models. This architecture will present two central modules with subcomponents that allow for future research concepts, such as the social relationships of agents and dealing with trustor attacks, to be more easily integrated with the existing subcomponents utilized by current trust establishment models.

Overall, these changes can provide real-world benefits as trust establishment models become more widely adopted in areas such as smart grids and e-commerce. As popular e-commerce environments become more easily accessible to intelligent agents via API and Web Service calls, the ability for agents to collaborate with other agents autonomously will be increasingly realizable in practice. Given the importance of trust evaluation in these environments to help determine who to purchase from, when businesses begin creating agents to help sell items in these marketplaces, the agents will need to use robust learning methods for improving and maintaining trust in the environment. An example to illustrate the importance of trust establishment models in these scenarios is that a model may learn that agents prefer fast shipping and will pay more for a product which includes faster shipping times. Thus, improved agent satisfaction and trust can be achieved by providing faster shipping while allowing for increased product costs. Due to the diverse nature of agents in specific environments, the research proposed within this thesis, which is evaluated within diverse simulation environments, will provide performance benefits that will be useful when working with diverse sets of agents. Therefore, the need for these improvements will become increasingly evident as trust establishment models become more adopted within real-world environments.

Trust evaluation presently is more prevalent in agent-based applications, but as mentioned, the need for trust establishment models will be increasing over time as it becomes more realizable for these environments to function in differing domains. When businesses rely on their trustee agent to help bring in additional profits and when smart grids must be able to reliably trust other devices which are represented as agents, establishing trust will help improve the state of transactions in these environments. This can result in additional profits over time and result in more reliable communication within an environment. Since trust evaluation is already used in e-commerce environments, using trust establishment models is the next step that is

needed to move towards more autonomous e-commerce platforms. Thus, in addition to being able to work well diverse MASs, trust establishment models must also be conformable to varying assumptions, such as when a physical product will take time to ship, resulting in a transaction not being complete for an undetermined period of time. As it will be demonstrated, this proposed research can help models become more modular and perform better, in general, than existing solutions. This can lead to additional profits and better transactions within the environment.

1.3 Problem Statement

Within the existing trust establishment model research there are many intelligent ideas that have been presented which allow trustees to become more trustworthy to trustors in an environment. Models such as ITE have proposed robust methods of more accurately making changes to a trustee's behaviour towards specific trustors with improvement and disimprovement variables [10]. When a model updates improvement and disimprovement rate variables dynamically in preparation for updating a trustee's behaviours for specific trustors, the model should update these variables independently for each trustor model contained by the trustee by using the data obtained from a grouping of most similar trustors. Through a simulated comparison with ITE, it is seen that updating independent improvement and disimprovement variables by using groups of similar trustors can help improve the performance of models which perform these updates.

As trust establishment model research progresses, there will be many ideas that can be used to improve existing models or that can be incorporated into new models. Existing models lack a formal architecture that is generalizable to encapsulate the design process of trust establishment models. By organizing the various components of a trust establishment model and by understanding how the future research ideas can be utilized in these models, a proposed generalized trust establishment model architecture will help with the design of trust establishment models and the ability to merge past and future ideas together. This architecture can be used to program simulation environments as a standard when performing research on the topic and can be used to enhance the performance of trust establishment models through the use a newly proposed preprocessing module. This thesis will outline both above approaches and will exhibit their performance in various tests with trustors of varied behaviours.

1.4 Research Contributions

This work presents the following contributions to trust establishment model research. Each of the contributions below will be explored in detail and the list will be revisited at the end of the thesis with additional details to support each contribution.

- Cluster-based Improvement and Disimprovement Rate Variable Updates
    - A cluster-based method for updating improvement and disimprovement rate variables within trust establishment models is presented; where ITE is used as the model to exhibit the performance benefits of using the technique.
    - The cluster-based method that is implemented within a modified version of ITE is presented to be generalizable to discrete-based trust establishment models that are designed differently than ITE.
- A Generalized Trust Establishment Model Architecture
    - A generalized trust establishment model architecture is proposed which allows new and existing models to be described within a more formal structure. This permits new research ideas to be more easily integrated into existing models, provides a structure to use when designing new models, and formalizes the design process to allow generalizable programming libraries to be developed for models and for simulation purposes.
    - The generalized architecture allows for both existing and future trust establishment models to improve their performance by using the new transaction-level preprocessing module that is presented within the architecture. This concept complements the existing ideas utilized by models, which are defined within a trustor-level postprocessing module, to allow the models to see improved performance without requiring a model to be completely redesigned.

1.5 Publications

The work seen in this thesis has been submitted to and accepted by a peer-reviewed workshop and a peer-reviewed conference. Journal papers of the work are also presently under consideration from two peer-reviewed journals. Each of these papers are listed below.

- A Cluster-based Integrated Trust Establishment Model for Intelligent Agents (accepted and to be published in the Proceedings of the 22nd International Workshop on Trust in Agent Societies)
- Improved Trust Establishment Performance with Transaction-based Preprocessing (accepted and to be published in the Proceedings of the 34th Canadian Conference on Artificial Intelligence)
- Cluster-based Improvement Rates for Trust Establishment Models in Single or Distributed Multi-Agent Systems (under review for publication in Wiley's Computational Intelligence journal)
- A Generalized Trust Establishment Model Architecture for Designing Robust Trust Establishment Models (under review for publication in Springer's Autonomous Agents and Multi-Agent Systems journal)

1.6 Thesis Organization

This thesis will organize itself into chapters which describe the research contributions in detail along with all information relevant to understanding the research that is performed. Following this chapter, Chapter 2 will discuss the background information relevant to trust establishment model research, will provide a literature review of papers relevant to trust establishment models, and will present concepts that are related to the research contributions. Each component of Chapter 2 will be referenced throughout the remaining chapters. Next, Chapter 3 will propose a cluster-based method of updating the improvement and disimprovement rate variables that are used by models such as ITE. Chapter 4 then presents a generalized trust establishment model architecture that can be used to design robust trust establishment models. This includes details on

the newly added transaction-level preprocessing module which is showcased within a newly proposed trust establishment model. Both Chapters 3 and 4 provide self-contained simulated comparisons and discussions to present the effectiveness of the contributions. Finally, Chapter 5 will conclude the work that has been performed and will provide details regarding the next steps to be taken in trust establishment model research.

Chapter 2

Background Information, Literature Review, and Related Work

2.1 Introduction

Before moving onto the chapters containing the research contributions that are made within this thesis, this chapter will provide all relevant information that is needed to understand the research contributions and the techniques that will be used. This begins with a description of background information related to the techniques that are being used by trust establishment models and related to the expected agent and environment architecture that is utilized throughout the thesis. This provides details on the topics which are necessary to understand when reading this thesis. Following this will be a literature review that provides more insight into literature surrounding trust and trust establishment to supplement the critical topics discussed in the background information section. The literature review will focus on papers relevant to this emerging topic but not those which propose trust establishment models. These trust establishment models will be discussed in the related works section since they are established models which directly relate to the work seen in this thesis and will provide context to the general designs and contributions made by each of the core decentralized trust establishment models that exist. This information will provide context on the works from Chapters 3 and 4 and will be expanded upon accordingly.

2.2 Background Information

In this section, the fundamental concepts that are used within this thesis and the design of the environment that is used within the thesis will be presented. This will begin with a description of the design of the environment and agents which will be used in future chapters. Following this will be information regarding Reinforcement Learning, Unsupervised Machine Learning, Online Machine Learning, and concept drift detection since these techniques are all discussed as part of the proposed research concepts. Finally, details on the evaluation methodology that is used by existing trust establishment model research and the evaluation methodology that is utilized in this thesis will be described.

2.2.1 Agent and Environment Definition

In trust establishment model research, the MASs that are discussed follow similar definitions between the presented models. The environment can be a singular or distributed MAS consisting of intelligent agents, where agents in multiple MASs are capable of interacting with each other and of performing transactions. These intelligent agents will have varying needs and their behaviours will vary between each agent. Although each agent in the simulations that will be described in future chapters take on the role of either a trustor or a trustee, the agents are capable of being a trustor and a trustee at any time step t. Within the scope of this thesis, a time step in an environment is defined as the time taken for each trustor to select zero or more interaction partners and for each trustee to propose and provide the UG values for the criteria of the specific task which a trustor requests during the time step. Within the same time step trustees also receive the direct feedback, or predict it if no feedback is received, from the trustors who performed a transaction with the trustee. This is because the communication between agents and the time taken to perform the digital tasks are instantaneous within the scope of this work.

Although the environment can take on different properties, the environment that will be referred to throughout this thesis is expected to be open, dynamic, and decentralized. An open MAS allows agents to act in a self-interested, diverse, and deceptive manner; whereas a dynamic

MAS allows agents to join or leave the environment [11]. The decentralized attribute means that the MAS will not contain centralized entities that handle the tasks performed by agents. Each agent will need to make their own decisions without the aide of a central entity by using the tools that are available within the agent's trust management module (such as the trust establishment model).

Despite this concept not being thoroughly explored throughout the thesis, the environment should also allow agents to be deceptive to one another through means such as having trustees lie about the proposed UG and having trustors lie or not provide any direct feedback to trustees [12]. A variety of trust-related attacks, such as when a trustor propagates false information about a trustee in the environment to decrease the trustee's overall trust, are discussed in [13]. These decentralized, open, and dynamic MASs allow for a diverse set of agents to reside within them and interact with each other.

In these environments, a trustor is defined by x_i, where x_i is the i^{th} trustor, and a trustee is defined by y_i, where y_i is the i^{th} trustee. Each trustee can perform zero or more tasks, denoted by s_i, where s_i is a single task out of all tasks that can be performed in the environment, defined by $S = \{s_1, s_2, s_3, \ldots, s_n\}$. Depending on the environment, each task can contain one or more criterion values that represent the different aspects of the task. Each of the p total criteria of a specific task s, where s is an arbitrary task from the set S, are represented as s_{c_i}. An example of a criterion for a task is the time it takes to perform the task, which can vary from being instantaneous for digital distribution to days for the physical distribution of goods.

Existing decentralized trust establishment models are either designed for tasks consisting of a single criterion or are designed for tasks consisting of multiple criteria. Thus, depending on a model's design there will be limits on the environments that they can effectively work within. Focusing on a single criterion is a more straightforward approach since there is less complexity when managing a trustee's behaviours for one criterion. Working with multiple criteria introduces a more complex task for trustees since the trustee needs to actively adjust to the needs of individual requirements that can vastly differ between the trustors who choose to interact with the trustee. Newer trust establishment models tend to be more robust by being designed with multiple criteria in mind to be able to handle tasks with both single and multiple criteria.

Since agents collaborate with one another in MASs, the collaboration between a trustor and trustee is referred to as a transaction. Before starting a transaction, a trustor x requests a service from the trustee y. Different trustees have different capabilities, thus some trustees may be able to serve unlimited trustors at a single time step or may only be able to handle a small number of trustors at a time. When trustee y receives a request from trustor x, the trustee responds to the request with a proposed UG to be provided for the requested task. This request then becomes an interaction between x and y. When trustor x receives the proposed UG, they may use their trust evaluation model to determine how trustworthy trustee y is, if that trust value is not already computed. This trustworthiness may be used to help determine whether the trustee will provide the full UG that is being proposed, or if the trustor should be expecting less. If the trustor is satisfied with the proposed UG, they respond to the trustee to agree to the transaction. The trustee can then provide the same UG that has been proposed, or a different UG. Following the transaction, the trustee will either receive the trustor's satisfaction with the transaction or will compute an estimate of the trustor's satisfaction (since the satisfaction may not be provided or may not be trusted). Alternatively, if the trustor is not satisfied with the proposed UG from an interaction the trustor may reject the interaction, resulting in no transaction being performed.

Following a series of transactions, a trustee uses its trust establishment model to update its behaviour towards individual trustors in the environment to help improve or maintain the trustee's trust within the environment. This trust establishment model is provided within the trust management module which each agent will be provided with. This trust management module will also include the trust evaluation module and the means to communicate with agents from the same or different environments. Since the trust establishment model that is used by the agent may require some historical data, agents should be capable of storing some amount of transaction history for each trustor. This data is stored within models of the trustors which the trust establishment model may use to adjust a trustee's behaviour in a specific manner to a specific trustor. If every transaction cannot be stored in memory, then at least the transactions from the last H active time steps should be stored. Here, an active time step refers to a time step in which a transaction is performed with at least one other agent. Trustees are assumed to have differing capabilities regarding their computation power and their available memory. Thus, a trustee can be equipped with a trust establishment model that best suits their capabilities.

2.2.2 Reinforcement Learning

When working with agents in MASs, Reinforcement Learning is a commonly used method to help agents learn in an environment. There are a variety of surveys regarding Multi-Agent Reinforcement Learning, such as [14], but the primary technique that is used by existing trust establishment models is Q-Learning. There are several different approaches that are used for Reinforcement Learning techniques, but the Q-Learning algorithm that is used by trust establishment models is a Temporal Difference (TD) approach. TD algorithms are useful in the open and dynamic MASs which the agents may reside within since the algorithms allow an agent to learn from their experiences without knowing the outcome from their time within the MAS (where a complete run of an agent's time within the MAS is referred to as an episode) [15].

Using Q-Learning, an agent can adjust its behaviour after each time step without completing the episode. Q-Learning works by updating a state-action value $Q(s, a)$ with the reward r that is obtained by executing the action a in state s, which leads the agent into state s' [16]. This results in the agent incrementally learning from experience within the environment without requiring the full episode to conclude. Existing trust establishment models, such as ITE, use this TD approach to help with a model's ability to learn an agent's behaviours over the long-term. This is useful in helping a model incrementally learn how to please individual trustors over a long period of time. This long-term learning can be complemented by other techniques to help handle random behaviours or to provide better short-term behaviour learning, as discussed within section 2.3 of this chapter. Thus, since the performance benefits of the TD approach have been exhibited by existing trust research, it will be used within the models discussed in this thesis. More details of Reinforcement Learning algorithms, their approaches, and the theory behind them can be found in a variety of Machine Learning literature, including [16].

2.2.3 Unsupervised Machine Learning

Despite Reinforcement Learning algorithms being the primary algorithm choice that is used to help agents learn in an environment, Unsupervised Machine Learning algorithms are important

tools that can be used to allow agents and their models learn specific information about an environment. Using Unsupervised Machine Learning algorithms, it becomes possible to perform groupings within an environment based on available information. These algorithms are called clustering algorithms and are useful for clustering data into groupings which consist of the most similar data. Despite not being commonly used when discussing trust establishment models, these algorithms will play a key role when discussing the framework in Chapter 3 of this thesis.

There are many clustering algorithms that have been proposed, each with their own pros and cons. One of the most well-known Unsupervised Machine Learning algorithms is the K-means algorithm. K-means is a distance-based clustering algorithm which uses the concept of cluster centroids to iteratively update each of the clusters [17]. Unlike K-means, which tends to be more useful for environments that are less densely populated and whose cluster shapes are more spherical, DBSCAN is a classic density-based clustering algorithm which finds clusters based on the density of the data [18]. When deciding which clustering algorithm to use for a specific problem or within a specific environment it is important to understand which algorithms are available and how they make decisions when performing the clustering. Thus, consulting surveys such as [19] and [20] can provide useful insight into which algorithms will be best suited for a specific environment or task due to their detailed comparisons between a variety of clustering algorithms.

Within Chapter 3 of this thesis, the K-means algorithm will be used when demonstrating the proposed cluster-based update approach. Although many other Unsupervised Machine Learning algorithms can be used, which can provide better overall performance, this simplistic model is selected to keep the focus on the cluster-based approach that is presented and to showcase that using a simplistic clustering approach can still result in performance benefits to a trust establishment model. It is seen that this algorithm is sufficient within the simulation environment due to the number of agents available and due to the trustor behaviours being static, however in practice it is important to understand that varying algorithms exist, with new algorithms consistently being proposed. Using these more sophisticated approaches can provide improved clustering accuracy and potentially dynamic hyperparameter tuning. This can result in better overall performance when used over a simplistic algorithm.

2.2.4 Online Machine Learning

Following the discussions regarding Reinforcement Learning and Unsupervised Machine Learning algorithms, there is another important Machine Learning algorithm type that will be used in Chapter 4 of this thesis. Online Machine Learning algorithms, which are also called Incremental Machine Learning algorithms, are Machine Learning algorithms which operate on potentially infinite streams of data and are trained as the data arrives. These algorithms play an important role when performing supervised tasks in environments where data is continuously being streamed to and from agents. Typically, these algorithms will process data streams from individual data entries or from batches of data entries which will then be used to adjust itself for improved performance or to perform some task on the data. A simplistic Online Machine Learning algorithm is the Hoeffding Tree algorithm which is a tree-based Machine Learning algorithm that can learn how to perform a classification task from an endless data stream [21]. Surveys such as [22] provide a more complete description on the various types of these Online Machine Learning algorithms and help to identify how they can be used.

The Hoeffding Tree algorithm will be used within the trust establishment model which is proposed within Chapter 4 of this thesis. There are other Online Machine Learning algorithms which can be used over Hoeffding Trees, some of which are more sophisticated and are known to provide better overall classification results. Although these can be applied to the model as a replacement to the selected Hoeffding Tree approach, the Hoeffding Tree approach is chosen due to its relative simplicity to understand and use. The purpose of the simulations being performed in Chapter 4 are to showcase the results obtained when using the fully proposed generalized trust establishment model architecture, rather than on the performance of individual Machine Learning algorithms which can be used. Thus, this simplistic algorithm is selected to help keep the trust establishment model simplistic within the scope of the thesis while still providing good overall performance. In practice other Machine Learning approaches can be used for better classification results, including improvements to Hoeffding Tress, such as Hoeffding Adaptive Trees. These algorithms can be selected based on the available tasks and agent behaviours within the environment. Many of these algorithms are also widely accessible via programming libraries.

2.2.5 Concept Drift Detection

When working with agents in MASs, there may be behavioural shifts within the environments at various points in time. This can occur for individual agents or for groups of agents and can vary in terms of the frequency at which the behavioural shifts occur. Thus, for scenarios in which behavioural changes can happen, it is important to be able to identify when a behavioural shift occurs and to ensure that any Machine Learning models are able to adapt accordingly to the shift. Concept drift detection can be performed to help identify when these shifts occur [23]. Concept drift is the issue in which existing Machine Learning models perform poorly due to behaviour shifts in the observed data [24]. Four types of concept drifts that can occur in an environment are sudden drifts, gradual shifts, incremental drifts, and reoccurring drifts; each of which are discussed in [24]. Understanding which of the concept drift types are occurring and having an appropriate method of handling these shifts is both an important and difficult task. Papers such as [25] are thus useful references for understanding how to formalize concept drift terminologies.

Two examples of concept drift detection algorithms are the Drift Detection Method (DDM) [26] and the Early Drift Detection Method (EDDM) [27]. EDDM improves upon DDM's ability to handle the detection of gradual concept drifts, which can be important when tasks may slowly see a rise in importance to specific criteria. DDM is less sensitive than EDDM and detects concept drifts later, which may be useful depending on whether a task will have known gradual shifts which do not want to be addressed by the agent, perhaps due to the agent's limitations on the amounts that can be provided for specific criteria, or depending on whether the data being used is known to contain noise. As Chapter 4 discusses, these techniques can be used to help with the trust establishment process when used correctly. However, these approaches remain unused within the scope of this thesis since the simulation environments that are used focus on diverse, but static, trustor behaviours. Since trustor behaviours are static, there will not be any concept drifts within the simulations, but in open and dynamic MASs there likely will be concept drifts which can be identified with algorithms such as DDM and EDDM.

2.2.6 Trust Establishment Model Evaluation Methodology

To understand the evaluation methodology that will be used within Chapter 3 and Chapter 4 of this thesis, this general evaluation methodology will be discussed. The specifics regarding the evaluations that are performed will be thoroughly described in their respective chapters, with this description supplementing them by providing a better understanding on why the chosen approach is selected. Furthermore, this subsection will explain the evaluation approaches which are currently used in this field and why the evaluation is performed through these methodologies.

Presently, there is no standard that is used for testing trust establishment models. In the existing models which will be discussed in section 2.4 of this chapter, the evaluations are performed through simulated environments. These simulation environments are used due to a lack of datasets which allow the general performance of a trust establishment model to be captured. However, existing research follows a general testing methodology which allows for a comparison of a trust establishment model's general performance. Within each simulation, the new model will be clearly defined alongside the details of another approach which is used as a baseline for evaluation. There may also be several other models involved to highlight the performance benefits of the newly proposed approach. The simulations are designed to allow for varying trustor behaviours, but each model may use different metrics to evaluate the models.

In terms of the evaluation metrics which are typically used, the average trust received is universally deemed to be important to track. This allows a model to be evaluated based on its ability to improve and maintain trust in the environment. However, this data insufficiently describes a model's performance on its own and is often coupled with one or more complementary metrics. The average UG being spent in transactions may also be tracked to determine how much must be spent to achieve higher trust. Comparing this value with the trust often gives a good indication of how well a model balances the trade-off between the amount spent and the trust received. Other metrics, such as the rate of meeting trustor desires, can be tracked to determine how well a model can meet a trustor's needs. By analyzing the results from each of these metrics, the research can conclude how well the model works, in general, for specific simulation environments. Not every model or research idea will need to use the same set of metrics, but the set of metrics should always include a combination of the trust received and

the amount spent to help determine how well a model works. Other metrics should be selected to help highlight what a research topic or model is aiming to improve.

Within the scope of this thesis, the evaluation process will be similar to those which are used for robust models such as ITE but will focus on understanding how well a model will work in diverse trustor environments [10]. Analysis on the general performance of a model in diverse environments is underrepresented in current research, thus the evaluations being performed within this thesis will provide comparisons which highlight a model's general performance in a diverse setting. This will help display how previous research can be improved in areas which are not well captured by their evaluation methodologies. The general methodology that this thesis will use begins by clearly defining a baseline approach alongside the research topic to compare with the baseline. The simulations will be defined to capture a model's performance in environments of varying levels of diversity, with a set of hyperparameters being used to control the settings of the simulations. Depending on the complexity of a model there may be many hyperparameters to help control both the simulation and model, but these hyperparameters are fully presented to ensure a complete understanding of how the tests are being performed. These values also allow a variety of unique trustor behaviours to be defined within each test while allowing each test to be reproducible. This is important since a model's performance in diverse environments is evaluated within simulation environments that are more and less densely populated with diverse trustor behaviours via varying trustor to trustee ratios.

These simulations will be run while tracking a specified set of evaluation metrics which effectively track the performance of the models for comparison in a diverse setting. The results will then be analyzed to understand the trade-offs between the models. Since each trustor will have differing needs and act in different ways, this allows a model's general performance in a diverse environment to be evaluated. Ideally a model will be able to effectively improve and maintain trust while minimizing the cost of doing so and will be able to quickly learn the needs of trustors to ensure that the trustee can quickly adjust itself when needed. The obtained simulation results will then be analyzed and discussed to determine the specific improvements of a model when working in open and dynamic MASs. Overall, this methodology provides fair and accurate comparisons when evaluating the presented research topics in diverse environments.

2.3 Literature Review

Before discussing the trust establishment models which directly relate to the content of this thesis, this section will provide insight into relevant literature that is related to trust establishment. Previous sections have highlighted the importance of trust establishment within a proposed trust management module, the expected agent and environment definitions, and the relevant literature regarding trust in Computer Science. Thus, this section will focus on topics that relate to trust establishment but are not directly related to the work that will be presented in this thesis.

Following up from the expected environment definition in section 2.2.1, there are other attributes of MASs that can apply to trust-related scenarios. These do not directly apply to the work that is presented within this thesis, but there are some MAS attributes that are interesting to highlight. This provides a better understanding of the types of environments which an agent may enter when the environment is open. Surveys regarding the attributes of various trust models in open MASs, such as [28], present thorough descriptions of each attribute along with a variety of trust models which present these attributes. Outside of a model being centralized or decentralized within an open MAS, models and MASs must also have clear expectations on the expected visibility of trust information for an agent, on the information sources that are to be available to models within the environment, on the possibility of cheating within the environment, and much more. Each of these concepts play a relevant role in the design and effectiveness of trust models, making it important to understand how a trust model can be generalizable enough to effectively manage as many of these attributes as possible.

As previously discussed, trust plays an important role in MASs. Trust is seen in many different forms within MASs, with each type of trust playing a crucial role in specific environments. Two fundamental categories of trust are individual-level trust and system-level trust, each of which categorize several trust categories [29]. Within the scope of this thesis, an agent tries to become more trustworthy in an environment of other agents. This is categorized as a form of individual-level trust since the other agents have their own trust values to represent their trustworthiness of the trustees. This differs from system-level trust in which the system itself forces the agents to follow specific rules to ensure that they are trustworthy. Despite the

importance of system-level trust, the trust establishment model research that has been done focuses purely on improving a trustee's individual trust with trustors in the environment.

Since trustees want improved trust in an environment, it is important to briefly discuss the trust evaluation models which compute the trust values that are to be assigned by a trustor to specific trustees. There are a variety of trust evaluation models that have been proposed, each using various ideas to compute the trust values more accurately. One example of a trust evaluation model is the Hybrid Trust Model (HTM) for Intelligent Agents [30]. HTM tackles the trust evaluation problem by using a combination of Q-Learning and Fuzzy Logic Systems (FLSs). This allows HTM to help deal with both random and short-term behaviours, in addition to learning long-term behaviours with the TD Q-Learning approach which is used by trust establishment models for the same reasons as HTM. Although FLSs are currently not used by many trust establishment models, they have been found to be effective at improving the performance of some trust models, including HTM. The FLS structure that is used by these trust models, alongside other areas in Artificial Intelligence, is described in [31].

An important observation from HTM is that using a combination of FLSs and Q-Learning helps the model more effectively manage many of the attack types that can be directed to trust evaluation systems, such as the attack in which an agent switches its behaviour back-and-forth from being malicious or honest [32]. As future research is performed on attack detection or prevention in trust establishment models, this technique may be useful. Another technique which HTM uses that directly affects how trust is computed for trustees is the concept of using both direct and indirect trust. Rather than solely relying on the trustor's view of a trustee, HTM uses witness testimonies from other trustors to calculate an indirect trust value that is combined with the direct trust. This is important to note when designing a trust establishment model since a trustee should focus on being trustworthy in the environment rather than simply with a small number of trustors to ensure that both the direct and indirect trust values that are computed by some trust evaluation models remain high. Since different trustors will use trust evaluation models that compute trust in different ways, the trust establishment models that are being designed must be capable of working well in varying scenarios.

2.4 Related Work

Despite being a relatively unexplored area of research, there are several trust establishment models that have been proposed. Higher level details on how each of these models work and their contributions will be discussed within this section. Each model either uses a unique approach to performing the trust establishment or directly expands off previous models. Understanding the concepts that are used by existing models will help highlight what this thesis aims to improve in this research field and will provide context on the techniques that are currently being used for trust establishment. Each of the models that are discussed in this section are decentralized models which aim to improve and maintain a trustee's trust with trustors in an environment. This allows agents to be more autonomous in their actions, without requiring a source to rely on for decision-making.

2.4.1 Trust Decision Making with Reputational Incentives

An early trust establishment model that is presented by [33] uses the concept of reputational incentives to allow trustees to adjust their behaviours towards trustors to increase their trust in the environment. This model has trustees determine the potential losses that can occur if a transaction with a trustor ends up being unsatisfactory and models the corresponding increase or decrease to the trustee's reputation. Based on these calculations, the trustee can adjust the UG to be provided for a task such that the trustee can actively manage its reputation within the environment. The paper presents this model to work with single criterion tasks, rather than with multi criteria tasks.

2.4.2 Establishing Trust in Multi-Agent Environments

Another single criterion trust establishment model that has been proposed is designed around the idea of having trustees predict the type of behaviour that is exhibited by a trustor and to adjust

the UG that is provided to that trustor based on which trustor type is predicted [34]. This model will be referred to as the ETME trust establishment model throughout this thesis due to it Establishing Trust in Multi-Agent Environments. ETME is designed with differing assumptions of how the environment is set up than what is presented in section 2.2.1 of this thesis, but the process that is taken is fundamentally the same. Each trustor still uses trust evaluation to determine which trustees should be interacted with in the environment at a given time step. Trustees using ETME sell items which each contain a price value and a quality value that are used when computing the UG that is provided when selling a product to a trustor. Here, the quality value represents all criteria of the item within a single value, similar to a single criterion task, and the price value represents the price of the product. Each trustor desires to obtain a UG value that is greater than or equal to a self-selected amount to be satisfied with the transaction and each trustee desires to offer UG values that allow the trustee to make a profit while still maintaining trust in the environment.

When determining how to provide a satisfiable UG value to a trustor while making enough profit, the trustees predict whether the trustor is price-sensitive (primarily desires a low-price value), balanced (primarily desires balanced price and quality values), or quality-sensitive (primarily desires a high-quality value). Depending on the predicted trustor type, the trustee uses a specific UG equation to compute the UG value to provide to the trustor for items of specific price and quality values. This results in trustees changing their behaviours towards a trustor if one classification method results in the trustor not being satisfied. An issue with this approach is that it has difficulties natively handling shifting trustors behaviours. Thus, this model serves as an important example as to how a model can benefit from applying new techniques to help combat existing weaknesses.

2.4.3 Trust Establishment Model in Multi-Agent Systems

A different type of trust establishment model is later presented in [35] which aims to assign weights to a set of criteria such that the trustee will know how to adjust its behaviours for each criterion. First, the importance of a criterion is captured by using the direct feedback that is obtained from trustors. This provides some fundamental information that can be used to

understand which criteria need to be enhanced by the trustee. By using this information, the model determines the demand of a specific criterion from trustors and calculates the average weight of the criterion. From this point, the model classifies each criterion into one of three distinct criterion sets such that the trustee will know exactly how to improve for those criteria and hence improve its performance and trust within the environment. This is a simplistic model which solely uses direct feedback to update a trustee's behaviour towards multiple criteria. This is problematic if a trustee does not receive direct feedback or if the provided feedback is false.

2.4.4 Reinforcement Learning Based Trust Establishment Model

The Reinforcement Learning Based Trust Establishment (RLTE) model is an important trust establishment model that is expanded upon by future models due to its strong architectural design and due to the ideas which the model presents [36]. RLTE uses Reinforcement Learning to dynamically adjust the behaviour of trustees to match the predicted demands of trustors by updating the UG that is provided by the trustee for a task. One of the important contributions that is made by RLTE is its use of trustor retention to help a trustee compute the UG that should be provided to each trustor to improve the trustee's trust with the trustors. The concept of trustor retention that is used by RLTE and future models that build off RLTE's design refers to how interested a trustee believes a trustor is to interact with the trustee for a task. The model uses these retention values to update the trustee's behaviours more accurately towards each trustor. Alongside trustor retention, RLTE also proposes the idea of using implicit trustor feedback when explicit feedback is unavailable, is not provided, or is not trusted. Using implicit trustor feedback helps improve a model's performance and is used by future models.

2.4.5 Fuzzy Logic Based Trust Establishment Model

Despite most trust establishment models using specific equations, potentially combined with dynamically updated hyperparameters, to update a trustee's behaviours towards specific trustors for specific tasks, another method that has been applied to trust establishment models is the use

of FLSs. The Fuzzy Logic Based Trust Establishment Model for Intelligent Agents (FTE) incorporates ideas from RLTE, such as the use of trustor retention, but also uses a FLS to better understand the behaviours of trustors in the environment [37]. Despite FLSs not being a major component of other models, this model is an important example of how trust establishment models can use a variety of techniques to help a trustee improve and maintain its trust in an environment. Since trust evaluation models such as HTM have benefitted from the hybrid approach of Reinforcement Learning and FLSs, FTE is an example of how FLSs can be used for trust establishment. This work may be helpful to future models as more trust establishment model research topics are investigated.

2.4.6 Acting as a Trustee Using Implicit Feedback

The Acting as a Trustee Using Implicit Feedback (ATeIF) trust establishment model focuses on using implicit feedback from trustors to improve the trust of trustees in the environment for tasks with multiple criteria [38]. This model builds off the structure that is presented by RLTE and uses specific equations with static hyperparameters to update the behaviours of trustees based on the implicit feedback that is obtained for specific trustors. In a multi-criterion environment, each trustor will assign different weight values towards each criterion of a task to represent the UG that they desire to receive for each criterion. Since this model handles tasks with multiple criteria, the model must constantly update what the trustee considers to be the criteria which individual trustors value the most and the criteria which individual trustors value the least. By having trustees assign predicted weight values to each criterion of each task, for each trustor, a trustee can better understand how much UG to provide to specific trustors for each criterion of each task over time. Since the weights are continuously updated, the trustees are also able to adapt to shifting trustor behaviours in the environment.

2.4.7 Integrated Trust Establishment Model

The final existing trust establishment model that will be discussed is ITE [10]. ITE's architecture will be discussed rigorously in Chapter 3 of this thesis, thus this description is an overview of the model. ITE is one of the current state-of-the-art trust establishment models which improves upon both RLTE and ATeIF. One of the key design decisions that is made by ITE is that it aims to optimize the balance between the UG that is spent to achieve higher trust and the trust itself. ITE uses both the direct feedback from trustors via obtained or predicted transaction satisfaction values and the indirect feedback values which are continuously updated to help with calculating the amount of UG to provide to a trustor. The model uses static hyperparameters, dynamic hyperparameters, direct and indirect trustor feedback, and trustor retention alongside complex equations to more accurately predict the weights that should be assigned to the criteria of a task, for each trustor. The retention values that are used are important for highlighting more specific ways to manage the trustee's trust with individual trustors.

The two dynamic hyperparameters that ITE uses to make updates more accurately play an important role in allowing ITE to manage the trade-off between the obtained trust and the provided UG. These parameters and their update process will be exhibited in Chapter 3 to highlight how ITE manages the rate at which it performs updates to a trustee's behaviour. Despite exhibiting good results in simulations, ITE is a complex model with many hyperparameters that are difficult to appropriately tune in practice. This highlights the trade-off which different trust establishment models take when balancing the model's ability to perform well and the model's accessibility to developers in practice.

2.5 Summary

This chapter has explored background information that is relevant to trust establishment models and to the contents of this thesis, has presented a literature review on papers that present information that supplement the work that is performed in this thesis, and has described a variety of trust establishment models that have been proposed. The background information that has been described provides insight into the various techniques that will be used and described

within the context of trust establishment. This allows a better understanding on the techniques performed and how they work. The literature that has been reviewed helps to provide additional context regarding the attributes of MASs and the type of work that has been done for trust evaluation. When discussing the purpose and design decisions of the work in Chapters 3 and 4, this will help with providing additional descriptions on the purpose and effectiveness of the research. Finally, the various trust establishment models that have been described will be referenced throughout the thesis since they exhibit the various design decisions that can be made when performing trust establishment. This helps to extrapolate information on what improvements should be made to help improve trust establishment research.

Chapter 3

Cluster-based Improvement Rates for Trust Establishment Models in Single or Distributed Multi-Agent Systems

3.1 Introduction

Within the open and dynamic MASs in which trustees use a trust establishment model to help improve and maintain their trust with trustors, there can be many different types of trustors that appear within an environment. These trustors may have drastically differing desires and may shift their behaviours over time. Thus, trust establishment models must be capable of sufficiently handling diverse sets of trustors in an environment. As explored in section 2.4 of this thesis, there are various trust establishment models which use differing techniques to help with trust establishment. Of these models, a common design choice among the more robust multi criteria models, such as ATeIF [38] and ITE [10], is to use specific variables when determining the rates at which to perform updates to a trustee's behaviour towards trustors. These two models derive this approach from the work that has been presented by RLTE [36]. This technique has achieved strong results in simulations and helps with more accurately performing updates to the trustee's behaviours. Of these models, ITE has converted two of these update rate variables into variables that are dynamically updated during runtime to ensure that updates are more specifically performed and that the updates reflect the general satisfaction within the environment.

Although ITE provides state-of-the-art results, the methodology that the model uses to update the two dynamic variables, which will be referred to as the dynamic improvement and disimprovement rate variables α and β, insufficiently handles diverse sets of trustors. Since ITE works with open and dynamic MASs which may be distributed, it is important that the approach better suits diverse groups of trustors. ITE's current methodology uses the RoC of trustor satisfaction at different periods of time to determine how the dynamic improvement and disimprovement rate variables should be updated. As discussed in section 1.2 of this thesis, this approach can result in RoC values that do not accurately represent the behaviours of specific trustors. To improve the performance of any trust establishment model which uses dynamic variable improvement components, whether with continuous or discrete dynamic improvement and disimprovement rate variables, this chapter will propose a new update mechanism that will utilize Unsupervised Machine Learning algorithms to perform the updates for groups of trustors.

By using Unsupervised Machine Learning algorithms for clustering, a trust establishment model can allow trustees to group trustors at a given time step based on a set of parameters that are available to the trust establishment model that is being used. By grouping similar trustors, a trustee can derive an improved representation of how the needs of similar trustors are or are not being satisfied at a given point in time. From these groupings, the dynamic improvement and disimprovement rate variables can accurately be updated for specific trustors based on the average needs from a group of similar trustors. This process helps to identify trustors with similar behaviours at a given time step and ensures that a trustee more accurately updates itself for the individual trustor groups. Although there have been frameworks and systems proposed for clustering in MASs, such as the Multi-Agent Data Mining framework that is proposed in [39], the concept that is presented in this chapter specifically presents a generalizable framework that will improve trust establishment model performance. Proper implementation of this approach is important in improving the overall capabilities of existing and future models.

The cluster-based approach that is presented in this chapter will provide a more robust method of performing the dynamic computations that are used to update the key learning rate variables of a trust establishment model by using Unsupervised Machine Learning. With this approach, the improvement and disimprovement rate variables are independently stored and updated for each trustor model, rather than being stored globally at the trust establishment level.

The core idea of the approach is to adjust a dynamic variable's value by the same amount for each trustor in a group of similar trustors, independently from dissimilar trustors, to provide a better approximation to the trust establishment model on how the trustee's behaviour should be adjusted at a given time step. Any updates that are made to the dynamic variables of trustors in a group based on the collective value that has been obtained for that specific group are only made for the trustors within that corresponding group. Thus, as groupings shift, trustees will be able to independently update their local improvement and disimprovement rate variables accurately at each time step by calculating the RoC of SAT for each trustor grouping. This concept also allows the dynamic variables of trustors which cannot be clustered to be updated based on the mean of the RoC values that are computed for each established trustor group. A high-level overview of how the cluster-based approach compares to ITE's approach is displayed by Figure 3.1 below.

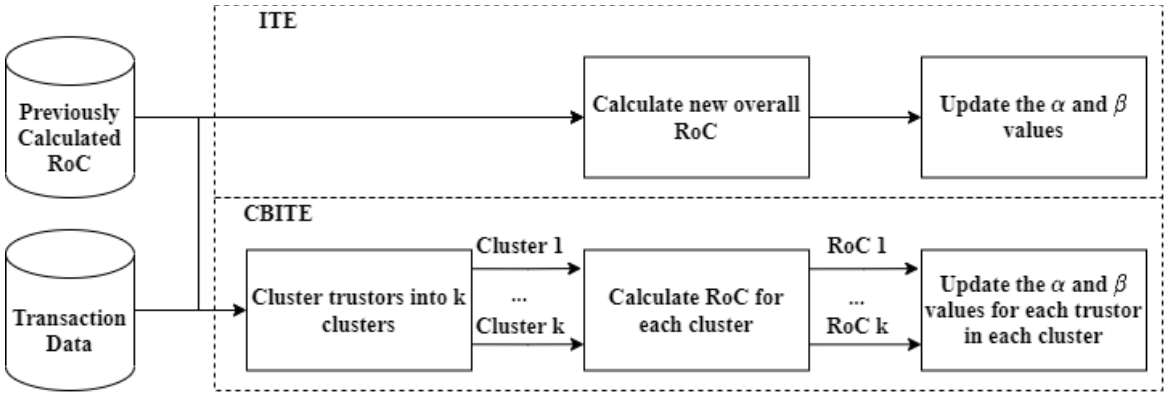


Figure 3.1 - A high-level comparison of ITE's approach and the cluster-based approach

This chapter will propose the cluster-based approach to dynamic improvement and disimprovement rate variable updates after providing a full description of ITE and outlining the agent and MAS architecture that is expected to be used for this cluster-based approach. The architecture of the full approach will be presented and applied to ITE to showcase how it can be used by models which use continuous valued dynamic variables and will then be generalized to explain how it can be used for models which use discrete valued dynamic variables. The results from simulated tests that are akin to those used by ITE are then presented and discussed in detail.

3.2 System Architecture

The trust establishment model architecture that is used to display the effectiveness of the cluster-based improvement methodology follows the general architecture that is presented by ITE [10], with changes made to some components that are not explicitly defined within ITE's paper. This system is a strong candidate to help display the importance of grouping trustors when updating the improvement and disimprovement rates for individual trustors due to its robust design. Thus, this section will present the cluster-based approach and how it can be used for continuous valued variables and for discrete valued variables. ITE's architecture will be thoroughly described after providing specific details on the expected agent and MAS architecture. This will highlight the motivations behind the cluster-based approach by analyzing the formulas that are currently being used by ITE. Following a look at ITE's improvement mechanism, the cluster-based approach will be described and applied to ITE. This will define the Cluster-Based ITE (CBITE) model which will be compared to ITE in the next section of this chapter. Afterwards, an example and description of how the cluster-based approach can be adapted for use by discrete-based trust establishment models will be presented to showcase how the concept can be applied to different scenarios.

3.2.1 Expected Agent and Environment Architecture

Before discussing ITE's architecture and the cluster-based approach, this subsection will discuss any details regarding the expected agent and environment architecture that have not already been explained in section 2.2.1 of this thesis. Since the cluster-based approach that is proposed throughout this chapter relies on the idea of trustor groupings, any trustees using a model which implements this technique must be able to run some Unsupervised Machine Learning algorithm to perform the clustering of trustors. Although the type of Machine Learning algorithm used can vary depending on the target environment or the capabilities of the agent, some algorithm that performs a sufficiently robust level of clustering must be usable.

Since ITE is a decentralized model, the agents which use the model, or the cluster-based variation of the model, should be able to make decisions without the help of other agents. This is important as it allows the intelligent agent to be autonomous and allows the agent to act in self-interested ways. Although an agent can have limits on the number of transactions that they can process or request at a given time step, these limitations will not apply throughout this thesis. Since the transactions are based around digital distribution, the trustees have no issues with servicing as many trustors as possible at a given time step. For these trustees, they also must be capable of storing models of each trustor in the environment(s). Since the MAS is open and dynamic, there may be trustors that come and go, causing certain design decisions to be made. Although a trustee can discard the models of trustors which leave the environment, it is beneficial to store that data to ensure that it can be used later if the trustor returns or if similar trustors appear in the environment(s).

Within an environment, a trust establishment model which will use the cluster-based technique should be able to approximate trustor feedback when it is not obtained. Since trustor feedback can be omitted or falsely provided in the target MASs, there should be methods of ensuring that high quality transactional data is available to the trustee. This is because the clustering algorithm(s) that are used will require data to use for the clustering and therefore should never be missing inputs or be relying on untrusted data. Furthermore, since the cluster-based approach is applied to dynamic variables that are used for dynamically updating a trustee's behaviour towards individual trustors, the trust establishment model that is being used must have some variables to make use of the approach. Although these dynamic variables may be used for different purposes, such as discrete values that are dynamically assigned to trustors following a transaction, this chapter focuses on the case in which they represent the improvement and disimprovement rates that a trustee uses to adjust its behaviours towards trustors. These dynamic variables should be stored within the trustor models which are contained by each trustee such that the trustees can use the variables to better understand how to adjust itself for independent trustors. Traditional Q-Learning based trust establishment models which implement dynamic variable updating mechanisms can more easily be adjusted to use the proposed cluster-based approach than other models.

Within the environments that will be used in the simulations that are performed throughout this thesis, trustors will be given different attributes to help define their behaviours within the environments. Since trustors will vary based on their demands and based on their level of activity within the system, these behaviours will be assigned to agents within the MASs that are used for simulation. This helps to replicate a diverse set of trustors within the environments which will interact with the trustees. This idea, which is also used by ITE [10], will be explained in more detail when discussing the simulation program that is used to test the cluster-based approach's performance when compared to ITE.

3.2.2 ITE's Architecture

To understand how the cluster-based approach can be used to improve the performance of trust establishment models, the approach will be applied to ITE. Thus, before discussing the changes to ITE's architecture and the specifics related to ITE's dynamic variable updating process for improvement and disimprovement rate variables, it is important to highlight the specific functionalities and equations of the trust establishment model that will be discussed throughout this chapter. This will provide additional context when discussing terminologies related to trust establishment, such as the use of relative weights, SAT, and more. The details regarding ITE are derived from ITE's paper and will be referred to in this chapter [10].

ITE is a multi-criterion, Reinforcement Learning-based trust establishment model which has results that are similar to, or higher than, other state-of-the-art trust establishment models in a variety of scenarios. One interesting concept that ITE uses is trustor retention. Trustor retention refers to the frequency of when trustee y performs a transaction with trustor x. This retention is calculated for each trustor to determine whether a trustor is engaged with a trustee, is unengaged with a trustee, or is neither engaged nor unengaged with a trustee at a given time step t. The core idea behind the use of trustor retention is to provide more control over how a trustee updates its behaviours for a trustor. This is done by checking whether the retention is above or below certain thresholds at a given time step (as seen in later equations). The general trustor retention value between trustor x and trustee y is computed with the equation below.

$$R_x^y(t) = \sum_{j=1}^{N_x} e^{(-(\rho)\Delta t_{xj})} \quad (3.1)$$

where,

- $R_x^y(t)$ is the retention between trustor x and trustee y at time step t
- N_x is the total number of interactions that have occurred between trustor x and trustee y within the last H time steps (where $H \in \mathbb{Z}^+$)
- ρ is the decay rate (where $0 \leq \rho \leq 1$) used to modify the Δt_{xj} values
- Δt_{xj} is the total number of time steps that have passed since interaction j occurred with trustor x
- t is the current time step

Using the retention value from equation (3.1), ITE can calculate how likely it is for a trustor x to interact with trustee y at time step t for task s. This is represented by $ret_x^y(s,t)$, which uses all the trustor retention values that are available to the trustee to reflect the retention of x more accurately.

$$ret_x^y(s,t) = \begin{cases} \frac{R_x^y}{\max\limits_{i \in X'}(R_i^y)} + \frac{R_x^y}{\operatorname{avg}\limits_{i \in X'}(R_i^y)}, & R_x^y > 0 \\ \operatorname{avg}\limits_{i \in X'}(R_i^y), & \text{Otherwise} \end{cases} \quad (3.2)$$

where,

- R_x^y is the retention value computed by trustee y for trustor x (see equation (3.1))
- X' represents the set of trustors that have interacted with y in the last H time steps
- $\max\limits_{i \in X'}(R_i^y)$ is the maximum retention value among all trustors i in X'
- $\operatorname{avg}\limits_{i \in X'}(R_i^y)$ is the average retention value among all trustors i in X'

When calculating $ret_x^y(s,t)$, the average and maximum trustor retentions from *X'* are computed. Since low retention values often correlate with a lack of trust and high retention values often correlate with higher trust, the use of the maximum and average retentions in equation (3.2) allows the trustee y to better model the likelihood of trustor x interacting with y at a time step t (due to potential trends in the environment showcasing a general shift in retention values). This retention value is used to determine whether a trustor is engaged or unengaged with the trustee. A trustor x is engaged with trustee y (denoted by $x \in X_e^y$) if $ret_x^y(s,t)$ is greater than or equal to the trustee's arbitrary engagement threshold. If $ret_x^y(s,t)$ is less than or equal to the trustee's arbitrary unengagement threshold, then the trustor x is unengaged with trustee y (denoted by $x \in X_{ue}^y$). If neither of the cases above hold, then the trustor is neither engaged nor unengaged with the trustee.

In ITE, each trustor contains a relative weight (rw) value for each criterion of each task. This rw value represents the trustor's desire of how much is expected from a specific criterion of a specific task (represented by s_{c_i}).

$$rw_x(s_{c_i}) = \frac{w_x(s_{c_i})}{d_x(s_{c_i})} \tag{3.3}$$

where,

- $w_x(s_{c_i})$ is the actual weight of the criterion c_i for a specific task s
- $d_x(s_{c_i})$ is the total demand that trustor x has for the criterion c_i for a specific task s

ITE's goal is to predict the rw values for each criterion of each task, for each trustor. By doing so, the trust establishment model can determine how to adjust the trustee's behaviour for the trustor such that the trustee's average trust is high, and the average provided UG is minimized. Thus, ITE stores a predicted rw value for each criterion of each task, for each trustor. This predicted rw is continuously updated to attempt to predict the true rw value that is displayed in equation (3.3). The predicted rw value is denoted by rw_x^y and is updated via Q-Learning to

continuously improve the accuracy of the prediction. The equation for calculating rw_x^y is presented below.

$$rw_x^y(s_{c_i}, req) = \begin{cases} (1+\alpha) * rw_x^y(s_{c_i}, tra'), & SAT_x^y(s, tra') < \Omega \\ & \text{and } rw_x^y(s_{c_i}, tra') \geq \phi \\ (1+\beta) * rw_x^y(s_{c_i}, tra'), & SAT_x^y(s, tra') \geq \Theta \\ & \text{and } rw_x^y(s_{c_i}, tra') < \phi \\ (1+\gamma * \alpha) * rw_x^y(s_{c_i}, tra'), & SAT_x^y(s, tra') < \Omega \\ & \text{and } rw_x^y(s_{c_i}, tra') < \phi \\ (1+\zeta * \beta) * rw_x^y(s_{c_i}, tra'), & SAT_x^y(s, tra') \geq \Theta \\ & \text{and } rw_x^y(s_{c_i}, tra') \geq \phi \\ (1+\lambda * \alpha) * rw_x^y(s_{c_i}, tra'), & \text{Otherwise} \end{cases} \quad (3.4)$$

where,

- $rw_x^y(s_{c_i}, req)$ is the predicted rw value for a specific criterion c_i of a specific task s, for trustor x by trustee y following the previous transaction *tra'* for the next interaction *req*
- $rw_x^y(s_{c_i}, tra')$ is the previously predicted rw value between x and y from transaction *tra'*
- γ (the attraction-factor-scaling parameter) and ζ (the profit-making-factor-scaling) are scaling factors that are used to adjust the predicted rw more accurately, where $0 \leq \gamma \leq 1$ and $0 \leq \zeta \leq 1$
- λ is a parameter that is used to attract trustors that are neither satisfied nor unsatisfied, where $0 \leq \lambda \leq 1$
- Ω is the trustee's threshold for considering a trustor to be considered engaged with the trustee and Θ represents the unengagement threshold (based on the SAT value rather than the retention), where the recommended values are $0.5 < \Omega$ and $0 \leq \Theta \leq 0.5$
- ϕ is called the low weight threshold and is used to define $rw_x^y(s_{c_i})$ as important if the rw is greater than its value

- α and β are the improvement and disimprovement variables which are dynamically updated by ITE, where $0 \leq \alpha \leq 1$ and $-1 \leq \beta \leq 0$
- $SAT_x^y(s, tra')$ is the trustor's satisfaction with the trustee who performed the task s for the transaction *tra'* (described in equations (3.10) and (3.11) below)

As seen in equation (3.4), ITE uses a robust set of conditions to calculate a good predicted rw value for each interaction with a trustor. This considers the trustor's engagement based on the provided SAT value and the current value of the predicted rw. The additional scaling values provide more flexible updates which results in a more accurate predicted rw value. However, since the purpose of this chapter is to focus on the improvement and disimprovement variables, the key takeaway from this equation is that it relies on the additional parameters that are used to determine how it should be updated. Having the correct α and β values will ensure that each rw_x^y value is being updated to accurately reflect the actual rw value of the trustor.

When the trustee determines how to propose an appropriate UG value for a trustor, the trustee computes this UG value based on an improvement formula that is calculated from both the minimum UG that can be provided by the trustee and the trustee's explicit and implicit feedback that have been obtained for the trustor in question. The improvement formula takes into consideration the explicit feedback from a trustor by using the predicted rw value (equation (3.4)) to compute the explicit improvement for the UG to be provided to a specific criterion of task s. Below is the equation that is used for calculating the explicit improvement for a single criterion of task s.

$$E\_Imp_x^y(s_{c_i}, req) = rw_x^y(s_{c_i}) * MaxImp^y(s_{c_i}) + MinUG^y(s_{c_i}) \quad (3.5)$$

where,

- $E\_Imp_x^y(s_{c_i}, req)$ is the explicit improvement to the UG to be provided for a criterion of a specific task s in the interaction *req* for trustor x, where $0 \leq E\_Imp_x^y(s_{c_i}, req) \leq MaxUG^y(s_{c_i}) - MinUG^y(s_{c_i})$

- $rw_x^y(s_{c_i})$ is the trustee's predicted rw value of trustor x's rw for a specific criteria of task s during the current interaction *req* (which is updated based on explicit feedback from x)
- $MinUG^y(s_{c_i})$ is the minimum UG that can be offered for a criterion in task s by trustee y
- $MaxImp^y(s_{c_i})$ is the maximum improvement that can be provided for a specific criterion of task s (this equation is defined below as equation (3.6))

$$MaxImp^y(s_{c_i}) = MaxUG^y(s_{c_i}) - MinUG^y(s_{c_i}) \tag{3.6}$$

  - Where $MaxUG^y(s_{c_i})$ is the maximum UG that y can offer for a criterion in task s

Contrary to the explicit improvement that is calculated in equation (3.5), the implicit improvement calculation is dynamically adjusted with the dynamic improvement and disimprovement variables (which are referred to as the engagement factors in ITE's paper). These implicit improvement values represent the implicit feedback that have been obtained by the trustee for each criterion of a task and are updated alongside the predicted rw values after a transaction. Below is the implicit improvement equation that is used by ITE which serves as an example of how these dynamic variables can be used to help update how the UG is calculated when proposing a UG value to a trustor.

$$I\_Imp_x^y(s_{c_i}, req) = \begin{cases} (1+\alpha) * I\_Imp_x^y(s_{c_i}, tra'), & x \in X_{ue}^y \text{ and } rw_x^y(s_{c_i}) > \phi \\ (1+\beta) * I\_Imp_x^y(s_{c_i}, tra'), & x \in X_e^y \text{ and } rw_x^y(s_{c_i}) < \psi \\ I\_Imp_x^y(s_{c_i}, tra'), & \text{Otherwise} \end{cases} \tag{3.7}$$

where,

- $I\_Imp_x^y(s_{c_i}, tra')$ is the implicit feedback value from the previous transaction with the trustor x, defined by *tra'*, and $I\_Imp_x^y(s_{c_i}, req)$ is the updated implicit feedback value to be used for the next interaction *req*

- ψ is called the high weight threshold which defines $rw_x^y(s_{c_i})$ as important if the rw is less than its value
- X_{ue}^y represents the set of trustors that are unengaged with the trustee
- X_e^y represents the set of trustors that are engaged with the trustee

Since this equation (3.7) and the predicted rw equation (3.4) rely on the dynamic improvement and disimprovement variables to better predict how to adjust a trustee's behaviour at a given time step, this displays that it is important to accurately update these variables at each time step. Through continuous updates, these values will better represent how to optimize the trade-off between the average trust from trustors and the average UG that is provided to trustors.

Using equations (3.5) and (3.7), the improvement function that is used to compute the proposed UG for a criterion from the task s during interaction *req*, can be defined as followed.

$$Improvement_x^y(s_{c_i}, req) = \omega * E\_Imp_x^y(s_{c_i}, req) + (1-\omega) * I\_Imp_x^y(s_{c_i}, req) \quad (3.8)$$

where,

- ω represents how the explicit and implicit improvement values are weighted in the equation
- This value must be inclusively between 0 and $MaxImp^y(s_{c_i})$

With the improvement equation (3.8) defined, below is the computation that is performed by the trustee to determine the total UG to be proposed to a trustor that interacts with the trustee.

$$UG_x^y(s, req) = \sum_{i=1}^{p} Improvement_x^y(s_{c_i}, req) + MinUG^y(s_{c_i}) \quad (3.9)$$

where,

- $UG_x^y(s, req)$ is the total UG to be provided to trustor x by trustee y for task s in the interaction *req*
- p represents the total number of criteria within task s

This UG value is proposed to a trustor who chooses to interact with the trustee. It is the summation of $ug_x^y(s_{c_i}, req)$ values, which represent the addition between the proposed UG improvement for a criterion of a task (equation (3.8)) and the minimum UG that can be provided for that criterion. When received, the trustor will decide whether to proceed with the interaction by selecting the trustee as a transaction partner or to decline the interaction. If the interaction becomes a transaction, the trustee will perform the service by providing the trustor with the corresponding UG (in this chapter, the provided UG is always calculated via equation (3.9)). The trustor then either provides explicit feedback to the trustor based on their satisfaction with the transaction (equation (3.10)) or the trustee ignores (or does not receive) this satisfaction value and predicts the trustor's satisfaction with the transaction (equation (3.11)).

The SAT for a transaction *tra* may be provided directly from the trustor x following a transaction with a trustee y with the equation below.

$$SAT_x^y(s, tra) = \sum_{i=1}^{p} \frac{w_x(s_{c_i}) * ug_x^y(s_{c_i}, tra)}{d_x(s_{c_i})} \quad (3.10)$$

where,

- $ug_x^y(s_{c_i}, tra)$ represents the UG that is provided by trustee y to trustor x for a specific criterion of a task s during the transaction *tra*
- $w_x(s_{c_i})$ is the actual weight of the criterion c_i for a specific task s
- $d_x(s_{c_i})$ is the trustor x's demand for the criterion c_i for a specific task s

Alternatively, the SAT may not be provided by a trustor or may not be trusted by the trustee, thus the following equation can be substituted for equation (3.10) to allow the trustee to predict the SAT value for the transaction *tra*.

$$SAT_x^y(s, tra) = \sum_{i=1}^{p} rw_x^y(s_{c_i}) * ug_x^y(s_{c_i}, tra) \quad (3.11)$$

With the SAT functions defined, each core component of ITE's general architecture have been summarized. These display the importance of the dynamic variables in adjusting the behaviours of trustees and provide context to the terminologies that will be used through the remainder of the chapter. In section 3.2.4, the specifics regarding ITE's dynamic variable update mechanism will be fully detailed to complete the description of ITE's architecture. In the simulations performed later, the trust establishment model that will be used as the baseline and the trust establishment model that will be used to exhibit the cluster-based approach follow the above architecture, with more details and changes presented in the upcoming subsections.

3.2.3 Changes from ITE's Architecture

Although the overall structure of the trust establishment model that is used to provide a comparison between this cluster-based dynamic variable updating approach and the current RoC approach is mostly the same as ITE's, there are some key differences. ITE's paper omits details on specific functionalities that are required when implementing a trust establishment model, thus the trust establishment model being used will contain the following components that provide the following functionalities that may differ from ITE.

The system starts by allowing trustors to select which trustees to interact with. The trustees are then notified about the selection process and provide a proposed UG value as detailed in equation (3.9), along with the UG values for each criterion of the task, to the trustors who have interacted with the trustee. If the trustor decides to pursue the transaction, they alert the trustee and do not reject the interaction. The trustee provides a UG to each trustor who has

interacted with the trustee, but who did not reject the interaction. This provided UG may be different from the proposed UG in practice but remains the same within the scope of this chapter. When the service is performed, the trustor will potentially return a SAT value to represent how satisfied they are with the transaction (as seen in equation (3.10)). This SAT value can be predicted by the trustee if not given or not trusted (as seen in equation (3.11)). The implicit improvement values (equation (3.7)) and predicted rw values (equation (3.4)) are then updated accordingly.

A trustee can receive any number of requests to be interacted with and will interact with all these requests by proposing a UG for each criterion of each task from the requests. These calculations are performed independently on a trustor-by-trustor basis, where the calculations are made based on the estimations made about the importance of each criterion for the requested task based on the trustor's profile which is contained by the trustee.

The MAS environment that is used in this chapter is set up quite differently than ITE's environments. The weights of the importance of a task's criteria are randomly set based on a Dirichlet Distribution, which defines the criteria weights to sum to one, for each trustor. The demands of each trustor, defined by a specified demand level, are randomly generated based on pre-defined ranges of values that are contained for each possible demand level that can be assigned to the trustor. This allows variety in the types of trustors in the environment since the demands vary for each trustor but are within a finite set of ranges to allow similarities to be found. These weights and demands are then used to compute the trustor's rw values (seen in equation (3.3)).

Although ITE's paper does not specify in detail how trustors interact with each other, this environment selects a trivial approach that allows the robustness of the newly presented technique to be seen from the tests. At each time step, every trustor requests to interact with every trustee so long as the trustor is active at that time step t. This is performed during the cluster-based approach simulations, rather than random sampling or through the selection of the most trustworthy trustee(s), as the trustor to trustee ratio is small in these trust simulations. Thus, through this approach, it is easier to view a complete comparison that is more representative of the larger, yet more diverse, traffic that trustees should generally expect in an open and dynamic MAS. This also provides a better representation of how the cluster-based approach of dynamic

variable updating will have a positive effect on the overall results when compared to the general RoC approach. The interactions are then accepted and the proposed UG is provided to the trustor to ensure that the test revolves around honest trustees. The transactions will be completed, and a specific SAT value (calculated in equation (3.10)) will always be sent to trustees rather than needing to be predicted for the simulations performed. Although the environment allows interaction rejection, interaction selection, and direct feedback prediction functionality, the architecture is set by default to run without these features within the scope of this chapter. This will better exhibit the effects that this proposed solution has on trust establishment models when working in open and dynamic MASs.

3.2.4 ITE's Dynamic Variable Updating Process

To complete the description of ITE's architecture, the process which ITE uses to update its dynamic improvement and disimprovement rate variables will now be discussed. This process serves as the baseline approach which will be compared to the cluster-based approach in the tests that are performed within this chapter. Thus, it is important to clearly highlight how the baseline approach works. Similar to the architecture descriptions in section 3.2.2, the details in this subsection are derived from ITE's paper [10].

The baseline variables that have inspired this work are the α and β values that are used by ITE. These are used to update the predicted relative weights of trustors and the implicit improvement values used by the trustee, which are both used to calculate the total UG to propose to a trustor (as seen in equations (3.4) and (3.7)). These α and β variables are dynamically updated based on the general satisfaction of trustors in the environment. This general satisfaction function, called g^y, returns the current RoC of trustor SAT divided by the previous RoC of trustor SAT. In the original paper, the equation is underspecified such that it does not specify whether the RoC is set for all transactions that have been performed or only for transactions with a specific trustor. As the newly proposed approach is relative to the RoC for all transactions, this chapter will compare the cluster-based approach with a baseline approach that follows this same assumption (the RoC is based on all transactions from all trustors). The general satisfaction value

can be computed after each transaction, after each time step, or after some other time interval. This is used to update the α and β values that are shared among trustors since they are globally shared by the trust establishment model. The satisfaction rate for a specific task is calculated by the following equation.

$$\overline{SAT_x^y(s)} = \frac{\sum_{j=1}^{N_{tr}} SAT_x^y(s,j)}{N_{tr}} \tag{3.12}$$

where,

- N_{tr} is the total number of transactions that the trustee has performed for a specific task s. In the original ITE paper, this is also denoted by N_{tr}^y [10]
- $SAT_x^y(s,j)$ is the satisfaction between trustee y and trustor x for the transaction j of task s

The general satisfaction rate which determines how the general rate of SAT in the environment has changed since that general rate has last been computed is then calculated via the equation below.

$$g^y = \frac{\widehat{SAT_x^y}}{\overline{SAT_x^y}} \tag{3.13}$$

- Where $\widehat{SAT_x^y}$ is the satisfaction rate (equation (3.12)) that is computed any time after the previous satisfaction rate $\overline{SAT_x^y}$ is computed

With the general satisfaction rate calculated, the trust establishment model will update the α and β values that are used to update the predicted relative weights that are assigned to trustors and the implicit improvement values that are used to help define how much UG the trustee should propose during future interactions with each trustor. In the comparative approach that is performed, this baseline approach will be compared to the cluster-based approach such that the

general satisfaction rate that is used will be updated after every transaction in a time step has been performed. This method is used, rather than after each independent transaction, since the changes made can be too drastic on the α and β values if done after each transaction. Doing the calculations in small batches may also lead to better performance, just as how Mini-Batch Stochastic Gradient Descent (SGD) can outperform traditional SGD in Deep Learning models, in part due to its increased stability [40]. However, this is not performed in this chapter.

The α and β values are updated as followed, based on the continuously updated general satisfaction function.

$$\hat{\alpha} = \alpha - \ln\,(g^{y}) \tag{3.14}$$

$$\hat{\beta} = \beta + \ln\,(g^{y}) \tag{3.15}$$

This updates the values according to the RoC that is computed for the state of the environment at different periods in time. The update functions seen above, along with the general satisfaction rate; will both be translated over to the newly proposed cluster-based approach.

3.2.5 Issues with ITE's Dynamic Variable Updating Process in Open Environments

From section 3.2.4, there are several prevalent issues that may arise depending on the environment(s) and the trustors in those environment(s). The first is that the α and β values are globally updated within the trust establishment model, rather than locally updated for models of each trustor. Setting the improvement and disimprovement rate variables to be defined globally can restrict the effectiveness of the updates that are performed. For example, if one interaction results in a very high satisfaction while another results in a very low satisfaction, the general satisfaction rate will be somewhere in the middle. Thus, the α and β values will not be updated to a satisfactory standard for that pair of trustors.

This leads into the second potential issue. Even if updates are performed to individual α and β values that are contained within a trustor model based on the standalone RoC of the trustor's SAT between two time periods, this can result in changes that are too drastic if a trustor's behaviours or attitudes shift away from what is expected for only a single time period. Although this issue is not completely removed by grouping trustors, an aggregation of trustor behaviours will result in more specific updates that better reflect the behaviours of trustors. This helps to ensure that any updates that are being made are less intense when a sudden shift in a trustor's behaviour is detected (stabilizing the adjustments).

3.2.6 Cluster-based Dynamic Variable Updating Process

Opposed to the baseline approach, this cluster-based approach is set such that each model of a trustor, defined by $x \in X$, where X is the set of all trustors in the environment(s), contains a variable α_x^y and a variable β_x^y, where y is the trustee who x interacts with. These are initialized according to the value that has been found to be optimal for the trust establishment model that is being used (since different models may use different variables with varied starting values). This approach is also able to compute the general satisfaction function g^y, similarly to equation (3.13). Furthermore, this approach will be configured with an Unsupervised Machine Learning algorithm, defined by Φ_{model}, which is configured to perform k groupings at each time the variables α and β are requested to be updated. Since the trust establishment model is decentralized, the Unsupervised Machine Learning algorithm will be fit with the appropriate data before the groupings are performed. The following process is done individually for each task performed. To simplify the notation, the task s is omitted from the following equations. However, note that the approach below is performed individually for each task $s \in S$, where S is the set of all possible tasks.

The Unsupervised Machine Learning algorithm that is used will have a large impact on the performance of this module, as different approaches should be used for different environments. Information regarding Unsupervised Machine Learning algorithms, along with specific examples of different types of these algorithms is presented in section 2.2.3 of this

thesis. As an example of where one type of algorithm is better than another in a specific MAS, in environments that are densely populated with trustors, a density-based clustering algorithm such as DBSCAN can be used to better group similar trustors than a distance-based algorithm such as K-means. With sufficient knowledge on the environment, the trustors in the environment, and the available Unsupervised Machine Learning algorithms, a trust establishment model can be outfitted with several clustering algorithms where a detection submodule can determine which to use at each time step. This is a problem that is not discussed in this thesis, but it is important to highlight the different possibilities that are available with this cluster-based approach.

After performing transactions with m different trustors for a specific task, defined by $X_{interacted} \subseteq X$, a trustee's trust establishment model will begin clustering every trustor $x \in X_{interacted}$ by using Φ_{model}. The trustors will be clustered on the SAT values which have been provided to the trustee, or which the trustee has estimated for the completed transactions with those trustors, along with the UG values which are provided by the trustee to those trustors, for the transactions. These are defined by SAT_x^y, which highlights the SAT values that have been provided by the trustors, and $providedUG_x^y$. These parameters can change depending on which parameters a developer determines to be most useful to use within a specific environment, but these are the parameters that will be used for the cluster-based approach in this chapter. Thus, the k clusters C are found by the Unsupervised Machine Learning model as followed.

$$C = \Phi_{model}(SAT_x^y, providedUG_x^y) \tag{3.16}$$

where,

- Φ_{model} is the Unsupervised Machine Learning algorithm that clusters trustors in $X_{interacted}$ for a specific task s
- SAT_x^y is the satisfaction of trustor x for the service provided by trustee y for a task s at a given time step t (see equation (3.10))
- $providedUG_x^y$ is the UG that is provided to trustor x for a requested service by trustee y for a task s at a given time step t (see equation (3.9))

For every cluster, the average trustor SAT that has been computed from the previous time step, which is defined as the average over all transactions which the trustee has completed and has stored in memory, is used to calculate the cluster average trustor satisfaction, defined by $\overline{CBSAT_{c_\iota}^{y}}$, where c_i is the i[th] cluster group in C. This equation extends equation (3.12) to work for clusters of trustors for a specific task.

$$\overline{CBSAT_{c_\iota}^{y}} = \frac{\sum_{q}^{|TR|} SAT_{TR_q}^{y} + \sum_{j}^{|c_i|} SAT_{x_j}^{y}}{|TR| + |c_i|} \tag{3.17}$$

where,

- $|c_i|$ is the number of trustors in cluster i (out of the k clusters), which contains trustors defined by x_j
- $SAT_{x_j}^{y}$ is the SAT that is provided by trustor x_j to trustee y for the performed task that has been completed during the current time step t
- |TR| is the total number of transactions for a specific task that has been completed by the trustee in any time step before the current time step
- TR_q is the q[th] transaction for a specific task that the trustee has completed in any time step before the current time step

Once $\overline{CBSAT_{c_\iota}^{y}}$ has been calculated for each of the k clusters to obtain the cluster-based equivalent of equation (3.12), the following is computed for each cluster.

$$\overline{CBg_{c_\iota}^{y}} = \frac{\overline{CBSAT_{c_\iota}^{y}}}{\overline{SAT_{x}^{y}}} \tag{3.18}$$

- Where $\overline{SAT_{x}^{y}}$ is the satisfaction rate for a specific task, as described in equation (3.12)

Equation (3.18) represents the RoC of trustor SAT based on a group of the most similar trustors for a given task at a time step t against the overall RoC from the previous time step t - 1. This process is an extension to the version that is defined by equation (3.13). This is calculated for each of the k clusters in C, and the corresponding α_x^y and β_x^y variables that are contained by each clustered trustor are updated through the following equations.

$$\widehat{\alpha_x^y} = \alpha_x^y - \ln(\overline{CBg_{c_i}^y}) \tag{3.19}$$

$$\widehat{\beta_x^y} = \beta_x^y + \ln(\overline{CBg_{c_i}^y}) \tag{3.20}$$

- Where these values are updated for trustor x only if x is in cluster c_i

To understand the general satisfaction trend in the entire environment, the general satisfaction rate for a task at time step t is calculated by the following equation.

$$\overline{avgCBg_c^y} = \frac{\sum_{i=1}^{k} \overline{CBg_{c_i}^y}}{k} \tag{3.21}$$

Equation (3.21) represents the averaged satisfaction rate amongst trustors who have completed a transaction with the trustee at time step t. This allows trustors who have not performed a transaction with the trustee at time step t to still be updated according to the trend of the environment at the given time step t. The trustors that did not complete a transaction with the trustee during a given round of transactions can still have their improvement and disimprovement rate variables be updated through the following two update functions.

$$\widehat{\alpha_x^y} = \alpha_x^y - \ln(\overline{avgCBg_c^y}) \tag{3.22}$$

$$\widehat{\beta_x^y} = \beta_x^y + \ln(\overline{avgCBg_c^y}) \tag{3.23}$$

- Only if trustor x did not complete a transaction at this time step with the trustee y

Thus, the above has demonstrated how to group trustors together with Unsupervised Machine Learning algorithms to be able to dynamically update their individual dynamic variables for a specific task. This is a generalized approach that can be adjusted in many ways. For example, the above can be done in mini-batches rather than at the end of each time step. This may improve performance when the trustors are more active in the environment, and when the number of trustors in the environment is sufficient. This, like most Machine Learning problems, is tunable according to the environment that the trustee is working within. There are also specific techniques that can be used to help perform this tuning process for specific algorithms. For example, there are techniques such as the depth difference (DeD) algorithm that can help estimate the optimal k value to be used when using specific clustering algorithms by analyzing the data depth [41] (need to select an appropriate technique for the model (Φ_{model}) that is selected). Techniques such as DeD can be run each time the clusters C (see equation (3.16)) need to be generated to ensure that the k value is optimal at each time step. Within the scope of this chapter, the value of k will simply be selected based on manual testing of various k values.

In the future, as trust establishment models become more robust and consider more features, this technique can be applied to allow models to better estimate dynamic variable values related to individual trustors. For example, when trust establishment models begin approximating the relationships between other agents, this approach can be used to better understand these relationships. If the model obtains some predicted relationship data or explicitly provided relationship data, the model can cluster the agents to detect patterns amongst these agents that can be used to predict their relationships more accurately over time.

3.2.7 Applying Cluster-based Updates to Discrete-based Trust Establishment Models

In section 3.2.6, the general process of the cluster-based dynamic variable updating process is presented and explained within the scope of being used within ITE, a trust establishment model that uses explicit improvement and disimprovement variables that contain continuous values. To provide more insight into how this process can be adapted to different trust establishment model types, an example will be provided when the improvement and disimprovement rate variables directly correspond with one or more discrete values. As a running example, assume that a trust establishment model updates a trustee's behaviour towards individual trustors by assigning a discrete value to each trustor that interacts with the trustee. In this example, there will be two discrete classes that can be assigned to a trustor. A trustor can be considered of the class overcompensated or of the class undercompensated. If a trustee classifies a trustor as overcompensated, an arbitrary amount will be deducted from the UG that is provided to that trustor in the next interaction between the trustor and trustee. Alternatively, when the trustee classifies a trustor as undercompensated, the trustee will add an arbitrary amount to the UG that will be provided in the next interaction with the trustor.

In this scenario, the improvement and disimprovement rate variables correspond with the discrete classes that are assigned to trustors since they directly determine by how much the trustee should adjust its behaviour towards the trustors. By identifying the improvement and disimprovement rate variables, many methods of applying the cluster-based updates that are explained in section 3.2.6 become possible despite the difference in design.

A simple method is to first perform the clustering as exhibited in equation (3.16) on trustors that have interacted with the trustee at a given time step and assign every trustor in a cluster the same discrete class based on any desired approach. An example approach to determine which discrete class to assign to all trustors in a cluster is to first use the selected trust establishment model's default method of assigning discrete classes to assign each clustered trustor their own discrete class. Then, for each cluster of trustors, the trust establishment model can determine the majority class within a cluster and assign that majority class to all trustors in the cluster. For example, if a cluster contains three trustors with the assigned discrete classes

(overcompensated, undercompensated, overcompensated), each trustor in that cluster will be assigned the overcompensated class if the majority class approach is used. If the trust establishment model desires to have trustors that have not interacted with the trustee at a specific time step to be updated alongside the trustors that have interacted with the trustee, this can be done by assigning the non-clustered trustors the majority class based on the assignments to each cluster. This process is similar to the process described in section 3.2.6, specifically equations (3.21), (3.22), and (3.23), with the primary difference being that the discrete classes themselves do not need to be updated afterwards.

In a discrete setting, when there are more than two possible discrete values, there becomes multiple improvement and disimprovement rate variables. The overall process remains the same as described above, however it becomes important to select a robust method of assigning discrete classes within clusters. Without careful consideration of this assignment method, inaccurate assignments can be made that result in updates that are too drastic for specific trustors. An example of a more robust method is to select the weighted majority class by assigning a weight alongside each discrete class assignment. A potential method to derive this weight is by calculating the percentage of times which the trustor has been classified as that discrete value in the past.

3.3 Evaluation

To compare the cluster-based implementation to the general RoC implementation, a variety of scenarios are run between agents in a simulated environment. All details regarding the architecture are presented in section 3.2 of this chapter. The cluster-based implementation that is presented will be referred to as CBITE since the architecture for the trust establishment model is related to ITE's design. The environment has been set up similarly to the environment that has been used by ITE but will perform differently since ITE does not explicitly specify the test scenarios used, nor the values for specific parameters used in the system. Furthermore, unlike ITE, the scenarios are not completed pre-generated and will thus perform differently than as presented in the original ITE paper. However, these scenarios are controlled through parameters

such that the results will remain static for a given set of parameters and the only difference between the comparisons of ITE to CBITE will be the calculations made for updating the α and β values. This is done by setting the random seeds that are used before defining the environment and agents or before running the simulation. Certain concepts from this section will also return in Chapter 4 of this thesis.

3.3.1 Simulation Setup

Since the code for the original ITE implementation, which is done in Java through the MASON library [42], is unavailable, this evaluation is set up in Python with the use of the Mesa Multi-Agent Modelling library [43]. Although the simulation environment is different, the program is set to run in the same manner as MASON simulations due to Mesa's BaseScheduler being used. This scheduler will schedule the agents in the environment in the same manner as MASON's scheduler. The environment is set up with the same parameters that are used from ITE's paper, with those not specified being manually selected for this simulation. It is important to note that the simulation only includes one type of trust evaluation model and one type of trust establishment model in a single run of the simulation. Thus, although ITE's experiments describe the use of several MASs to replicate the distributed nature of the Internet of Agents, this has no effect on the actual testing on the performance of the trust establishment model in a distributed environment.

The simulation is set such that there is only a single task that an agent can select from or perform, but a trustor is not required to request interactions with trustees at each time step (they can choose to not participate in interactions for any time step). Just as with ITE, each task is provided a set of criteria for which trustors have independent relative weight values for (see equation (3.3)). Thus, trustees will need to make use of the trust establishment model to best predict these relative weights over time, as seen in equation (3.4). CBITE uses the K-means algorithm for clustering over other clustering techniques due to its overall simplicity and due to how the tests are defined. Although K-means has flaws, such as its inability to effectively handle outliers or form distinct cluster shapes [44]; when working with a limited, but representative,

sample size in a simulated environment it is a fine option. However, if the K-means algorithm is desired, variants exist that optimize the algorithm and remove weaknesses found when using it. One example is the PM-Kmeans method which uses particle swarm optimization as a method of improving the original K-means algorithm [45]. As mentioned previously, dense environments benefit from density-based algorithms such as DBSCAN. Like K-means, there are also variations of these density-based algorithms, such as PODCC, which use particle swarm optimization to help fix several of DBSCAN's weaknesses [46].

To allow trustors to be able to select zero or more trustees to interact with at each time step t, this chapter uses the concept of activity levels which is also used in ITE [10]. Thus, at each time step t a trustor only interacts with trustees if they are deemed to be active. In the simulation program that is used for testing, trustors are each assigned one of three available types of activity levels. Highly active trustors will request service(s) with a probability of at least 65%, trustors with a low activity level will interact with trustees if the probability is at most 35%, and regularly active trustors are active when the calculated probability is between 35% and 65%. Systems may or may not use the concept of activity levels, but the concept is used in this simulation to help facilitate a more diverse set of trustors in the environment.

In the simulation, once a trustor receives the proposed UG values from the trustee for an interaction, this becomes a transaction, and the transaction is considered to be good or bad depending on whether the provided UG meets the specified demands for each criterion. High demanding trustors require that at least 65% of the maximum UG will be provided by the trustee for each criterion of a task. Low demanding trustors require at least 35% of the maximum UG to be provided by the trustee for each criterion of a task. Finally, regular demanding trustors require between 35% and 65% of the maximum UG to be provided by the trustee for each criterion of a task (the average of the high and low demand requirements). These demand levels are also used when instantiating trustors, such that the demand values that are used for the computation of the relative weights of each criterion are randomly generated within a demand level's pre-defined demand ranges. This helps increase the different behaviours that can be found in the MAS to better represent the variety found in open and dynamic MASs.

The trust evaluation model that is used by the trustors is the same probabilistic model used by ITE's simulation. This model computes a trust value between [min_trust, max_trust] to

represent each trustee's trustworthiness. The model uses the trustor's direct trust, which is the average percentage of SAT that the trustor computes for each transaction with a trustee. It also uses the indirect trust, which is the average of the direct trust values from all trustors, other than the requesting trustor, who have interacted with the target trustee. A trustor receives this indirect trust value by communicating with other trustors to request their direct trust values for a trustee. In this simulation, each trustor will accurately provide this value when it is requested by other trustors. The trustor's total trust value is then calculated with the equation below.

$$trust_x^y = direct\_trust * 0.5 + indirect\_trust * 0.5 \quad (3.24)$$

In this simulation, for the purposes of comparing the results that are obtained to ITE's original test results, trustees act honestly with all trustors by providing the UG amount that is proposed. However, trustors and trustees can misbehave in practice, which will add a variety of behaviours that can be modelled by the clustering algorithm which groups similar trustors together. With respect to trust evaluation, the system defines the ability for trustors to compute the trust of trustees, but for the purposes of testing, the trustors will not reject interactions with trustees. That said, trustors still provide direct feedback through a calculated SAT value to alert trustees about their performance. Although trustees can approximate the SAT, trustors will always provide it in this system and trustees will always trust the SAT that is received.

All values used during the simulation are presented in Table 3.1 on the following page. This table includes lists of values when each index of the provided list is used for that corresponding test ($test_0$, $test_1$, and $test_2$). Individual values within the table are used for all tests. Although parameter tuning can be further performed, these values are sufficient in providing an accurate comparison between each test when working with an ITE-based trust establishment model.

Parameter	Assigned Value	Parameter	Assigned Value
Number of agents (N)	100	Θ	0.75
Number of trustors	[12, 24, 36]	λ	0.1
Number of trustees	[88, 76, 64]	ζ	1
High demanding trustors	[4, 8, 12]	γ	1
High demanding value	65%	Ω	0.5
Low demanding trustors	[4, 8, 12]	ϕ	0.25
Low demanding value	35%	ψ	0.5
Regular demanding trustors	[4, 8, 12]	ω	0.5
High activity level trustors	[4, 8, 12]	H (transaction sliding window)	1000
High activity level value	65%	ρ (decay factor for retention)	0.5
Low activity level trustors	[4, 8, 12]	Minimum UG	0.1
Low activity level value	35%	Maximum UG	1
Regular activity level trustors	[4, 8, 12]	Engagement threshold	0.75
Number of steps	1000	Unengagement threshold	0.25
Steps to bootstrap	5	Minimum trust (min_trust)	0
Number of tasks	1	Maximum trust (max_trust)	1
α (initial value)	0.02	Number of criteria per task	10
β (initial value)	-0.01	Number of clusters	[-1, 5]

Table 3.1 - Parameter setup for the ITE and CBITE simulation program

3.3.2 Tracked Evaluation Metrics

As the simulation progresses, the program tracks a set of metrics that represent the results from the simulation at each time step. Since this comparison focuses on comparing the cluster-based approach against the RoC baseline, the metrics that are tracked are the same as what is being tracked in ITE's testing [10]. This ensures that the results and the corresponding discussion directly relate to the baseline results without discussing factors that do not reflect important changes between the approaches.

The *Average Direct Trust* over time is tracked. This represents the average direct trust of all trustors in the system at the end of each time step t. This will help to illustrate the effects that both approaches will have on the overall trust of trustees by trustors. Here, a higher value is better, however a higher value is considered better only if the number of good transactions (seen below) are high. A balance between this and the average delivered UG (discussed directly below) is important since the trustee can trivially give the maximum UG each time to achieve the maximum trust.

The *Average Delivered Utility Gain* is the average UG that has been provided to trustors by trustees, for all transactions that have performed in the environment since a specific time step. This is tracked to determine which approach is better at managing the trade-off between gathering trust and providing a higher UG. Here, a lower value is better if the average trust remains at a satisfiable level. Since trustees want to save resources to ensure that they do not operate at a loss, this data is crucial to analyze alongside the corresponding average trust.

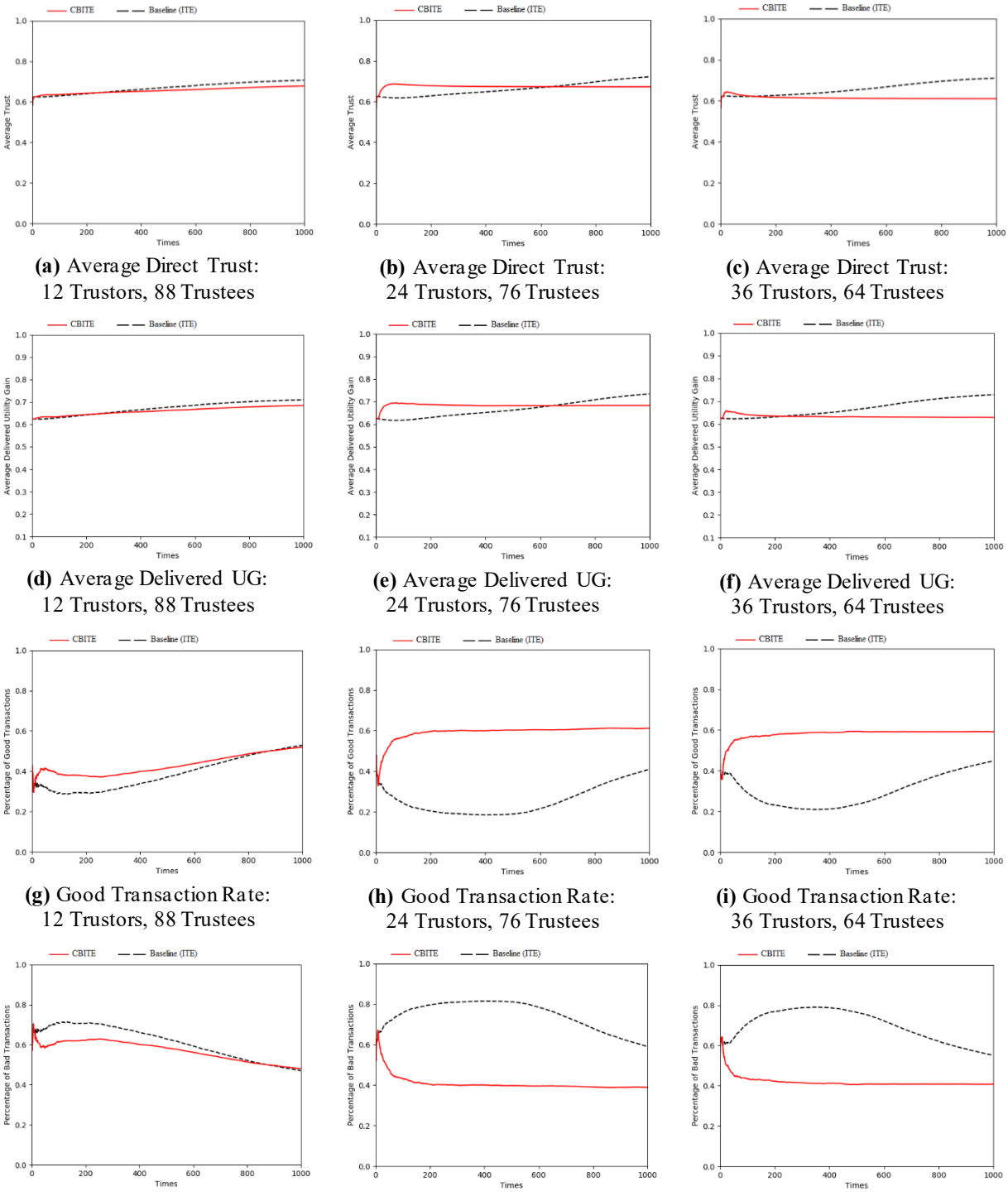
The total *Good Transaction Rate* and *Bad Transaction Rate* is tracked to represent whether a trustor's needs for each criterion of a task are completely being met by the trustees. These are computed as the percentage of good and bad transactions from all transactions that have been performed. Here, a good transaction is defined as one in which the provided UG values for each criteria of a task match or exceed the demand threshold of the trustor for those criteria values. Bad transactions represent any transactions which do not completely satisfy the trustor's demand for each criterion. The demand is defined by the maximum possible UG for a criterion multiplied by the rate associated to each demand level that is assigned to a trustor. The

more good transactions and less bad transactions, the better the trust establishment model is performing at adapting to match the overall needs of the trustors.

3.3.3 Results Obtained

Now that the simulations and metrics have been defined, the results that have been obtained from simulating the baseline approach and the cluster-based approach will be compared and analyzed. It is important to emphasize that the scenarios which are generated in ITE's paper change the base configuration of the system for each run, while these tests are obtained from different scenarios using the same initial configuration. This means that the results will appear to be different due to the variations from ITE, as mentioned in section 3.2.3 from this chapter.

Since this chapter proposes a more robust method of updating the dynamic improvement and disimprovement rate variables that are used to update a trustee's behaviours, three different tests have been prepared for the ITE-based baseline model and CBITE. As previously stated, these tests run the same in every way except for the dynamic variable updating module. This delivers accurate, replicable tests for comparison. Each test changes the trustor to trustee ratio within the environment to allow for a better representation of how the models handle a larger set of diverse trustors. In the first test, there are 12 trustors and 88 trustees in the environment. This test provides insight into a model's effectiveness when faced with a less diverse environment. In the second test, the same test is performed, except with 24 trustors and 76 trustees to better test how an incremental increase in diverse trustors will affect the models. This non-drastic increase to the sample size of trustors will provide insight on the difference in performance when slightly more varied behaviours exist in the environment. In the final test, the environment contains 36 trustors and 64 trustees. This test provides a more drastic change in the trustor to trustee ratio that allows clear comparisons to be made with respect to the effectiveness of managing varied behaviours. Each test also results in drastic increases to the number of interactions and transactions that occur due to the design of this testing environment. This helps understand how each model copes with more interactions that require varying UG values to be computed for different criteria of a specific task.



(a) Average Direct Trust: 12 Trustors, 88 Trustees

(b) Average Direct Trust: 24 Trustors, 76 Trustees

(c) Average Direct Trust: 36 Trustors, 64 Trustees

(d) Average Delivered UG: 12 Trustors, 88 Trustees

(e) Average Delivered UG: 24 Trustors, 76 Trustees

(f) Average Delivered UG: 36 Trustors, 64 Trustees

(g) Good Transaction Rate: 12 Trustors, 88 Trustees

(h) Good Transaction Rate: 24 Trustors, 76 Trustees

(i) Good Transaction Rate: 36 Trustors, 64 Trustees

(j) Bad Transaction Rate: 12 Trustors, 88 Trustees

(k) Bad Transaction Rate: 24 Trustors, 76 Trustees

(l) Bad Transaction Rate: 36 Trustors, 64 Trustees

Figure 3.2 - Comparative results between CBITE (red) and the ITE baseline (black)

As seen in Figure 3.2, CBITE results in a minor decrease to the overall direct trust of trustors, but also decreases the average UG that is provided for tasks. Since the trust difference between the two models is minor, this decrease to the provided UG is more beneficial to the trustee. Out of all the transactions, CBITE has a much higher good transaction rate than the baseline and a much lower bad transaction rate than the baseline. This means that CBITE is better respecting the needs of trustors at varying demand levels for an overall lower UG than the baseline. This will save the trustee some resources throughout their involvement in the MAS, while still allowing the overall trust among trustors to be high and adjustable (as seen in plots (a), (b), and (c)).

To further display that CBITE more effectively manages trustor needs, in the first test that is performed (exhibited by plots (a), (d), (g), and (j)), although CBITE begins providing less average UG to trustors after roughly 250 time steps, the good transaction rate consistently increases and stays above the baseline's rate until roughly the end of the simulation. Although this difference is minor, there is a more prominent difference with respect to the average direct trust and average delivered UG as the number of trustors in the environment increases (displayed by plots (b), (e), (c), and (f)) and a drastic difference when compared to the remaining plots ((h), (k), (i), and (l)). These exhibit that CBITE can decrease the average UG that is provided to trustors while drastically improving the good transaction rate. This, compared to the baseline system's good transaction rate which slowly decreases then slowly increases, remains high and stable throughout the remainder of the simulation. Furthermore, since these second and third tests introduce more varied trustors, this displays that CBITE is better at dealing with larger numbers of trustors with unique behaviours than the baseline approach.

Plots (a), (b), (c), (d), (e), and (f) display that the larger and more diverse the trustor population, the bigger the difference between the two approaches is with respect to the trade-off between the average direct trust and the average provided UG. However, it can also be seen in those same plots that CBITE drops the average direct trust below the baseline around the same point where the respective average provided UG decreases. This behaviour is expected given the test architecture's design choices, as it displays that the trust establishment model is determining that the UG can be lowered at a slow rate while still maintaining a high enough trust. Since there is a bigger difference with respect to the results between the models as there are more varied

trustors in the environment, this indicates that locally updating the improvement and disimprovement rates for the trustors with clustering algorithms does have a noticeable impact on the results. Although these plots exhibit the behaviour of a single simulation environment in a single trust establishment model architecture, these results clearly display the effectiveness that the cluster-based approach provides in a general setting when compared to an architecture akin to a state-of-the-art trust establishment model.

When there are less trustors, the difference between the models is smaller. This is because it becomes harder to gain meaningful information from smaller groups. Thus, since the compared results deviate more when the trustor size grows, this displays that the clustering approach that CBITE performs can form meaningful groupings. These groupings are meaningful, and the cluster-based process is better than the baseline, because the results exhibit that a consistently higher percentage of trustors are having their needs fully fulfilled while providing less overall UG. Furthermore, since there is a more stable change between plots (h) and (i) when compared to the drastic difference between plots (g) and (h), when comparing more varied environments the obtained results will be more stable when given the same parameters. This stabilization can be configured through the parameters given to the cluster-based module. In this test, a k value of five is selected as the result from testing which k value works best for each of the three tests (five total clusters). However, when increasing the diversity in the environment, different k values should allow for more variety to be captured and dealt with. For example, if seven clusters were to be used when performing the test in plot (i), the increased variance can be better captured at certain time steps to allow a bigger difference in the results when compared to plot (h). This, along with the option of selecting various clustering algorithms according to the heuristics of the environment(s) (perhaps automatically detected by the agent) allow for this cluster-based approach to be specifically tuned to the needs of the trust establishment model that is being used and to the environment(s) that are being operated in.

Thus, these results strengthen this chapter's hypothesis and exhibit that the cluster-based approach that is presented by this chapter results in stronger trust establishment for existing models. Since the results in Figure 3.2 are all improvements over the baseline (except for the average direct trust which eventually dips below the baseline's results in the tests), these tests have displayed the importance of grouping trustors when adjusting the trust establishment

model's dynamic improvement and disimprovement rate variables that are used to modify the trustee's behaviours for each individual trustor.

3.4 Discussion

With the results obtained from section 3.3.3, the key takeaways from the results will be discussed to highlight how the cluster-based approach improves the performance of trust establishment models that dynamically update their improvement and disimprovement rate variables. As mentioned, the key focus of these tests has been to understand how grouping updates will affect a model's performance. Thus, the tests being based on the trustor to trustee ratio helps to create simulated environments with more trustor behaviours being present and with more trustors being similar to other trustors. The hypothesis behind the performed tests has been to showcase that when there is a more diverse set of trustors, that in open and dynamic MASs, updating a trustor's model individually based on the combined values from a similar group of trustors will more accurately reflect the needs of the trustor than without averaging based on a group of trustors. The results from these tests exhibit the validity behind this hypothesis as previously analyzed and as discussed below.

Due to CBITE's much higher good transaction rate when compared to the baseline, using groupings and storing the dynamic variables independently within the trustor models helps to better identify the needs of individual trustors. Combined with the lower average proposed UG, this implies that the average trustor who interacts with the trustee will be satisfied despite the total UG not being as high. Part of the reason why this occurs at the expense of some direct trust is because some trustees learn that a group of trustors may be too demanding and will decide to focus on improving trust with the groups of less demanding trustors for less UG, rather than providing excessive amounts of UG to trustors that are highly demanding. Due to the varying demand and activity levels that are assigned to each trustor, maintaining a stable set of good transactions indicates that the trustee will continue to receive interactions from trustors who logically consider previous transactions when searching for interaction partners. Thus, this

indicates that the trustee will achieve more transactions with trustors that use a logical trust evaluation model and selection algorithm.

Overall, the selected sample size and the tests performed are good representations of how the model can continuously improve the rate at which it updates the trustee's behaviours for each trustor. That said, performing further tests with this approach on larger, more skewed environments can further display the potency of the approach in different scenarios. Similarly, the cluster-based approach can be tested with real datasets which may contain larger sets of dissimilar trustors. This can help with determining the optimal clustering algorithms and corresponding hyperparameter values to use when working in scenarios with less similar trustors. Further testing with various trust evaluation models being used by the trustors will also help further prove the effectiveness of this approach in specific situations. Essentially, the more varied the environment is with respect to the components which are used by the trust management module that is contained by each agent, the clearer the distinction between clustering and not clustering becomes.

By tracking the results from Figure 3.2 with the data which can be collected based on the ranges of possible trustor behaviour, it becomes possible to track how effective the implemented clustering approach performs at each time step. Each clustering that is performed at a time step can be monitored and displayed following a simulation to allow for the manual selection of the best Unsupervised Machine Learning algorithm for the environment type. This can also be done dynamically by setting threshold values to determine which different algorithms should be used and when. Although the effectiveness of a variety of clustering algorithms has not been tested in this chapter, the idea of dynamically swapping algorithms is possible and can lead to better results if correctly implemented in practice.

Here, the cluster-based approach uses the provided UG and the provided SAT to cluster the trustors with the selected clustering algorithm. This is one of many potential approaches, where data such as the predicted relative weights (see equation (3.4)) can also be used. However, the tests exhibit that clustering with the provided UG and SAT generates positive results and can thus be used by default and later tuned to benefit from the specific trust establishment model that is being used. Since this research is generalizable to many trust models, the proposed solution can be adjusted for a specific model and can help to achieve the best results possible with that

model. Even when SAT is not specifically used in an environment, so long as there is a metric for feedback (direct or indirect) that is present, this approach can be used.

The biggest challenge with this cluster-based research has been the creation of a simulator that can accurately reflect MASs that allow trustors to act in different ways. Without a standard, the overall testing that is performed is based on independent criteria that reflect the effectiveness of the model but does not directly compare to the testing that has been performed by older trust establishment models. When a standard for testing is created, then results obtained from these tests can be extended to display how it outperforms other approaches when performing the same tests. Thus, a balance between following existing tests and defining new tests is difficult when comparing trust establishment models or when comparing extensions to existing architectures.

3.5 Summary

This chapter has presented a new method for dynamically updating the improvement and disimprovement rate variables that can be used by trust establishment models to better understand how a trustee should update its behaviours towards specific trustors. Each trustor model that is contained by a trustee should contain these dynamically updatable variables, rather than having the variables be held globally by the trust establishment model. The proposed cluster-based methodology involves clustering trustors into groups such that each trustor in a group of similar trustors has their dynamic rate variables updated in the same way at a given time step. When compared to the similar architecture of a state-of-the-art trust establishment model, this approach yields slightly lower average direct trust, but produces a consistently higher good transaction rate with a lower average delivered UG. The effects of this approach become more prevalent as the environment becomes larger and more diverse. Thus, this exhibits that the cluster-based approach improves trust establishment model performance and can be implemented in robust trust establishment models.

Chapter 4

A Generalized Trust Establishment Model Architecture for Designing Robust Trust Establishment Models

4.1 Introduction

Despite trust establishment model research being relatively undeveloped, the models that have thus far been proposed provide a variety of ideas as to how trust establishment can be performed. Each trust establishment model aims to increase the overall trustworthiness of a trustee in an environment. This is important since logical trustors will desire to only interact with trustees which an equipped trust evaluation model deems to be trustworthy. Since each trustor may use different methods to compute the trust of trustees in the environment, trust establishment models must be capable of working well in a variety of scenarios. Each current trust establishment model offers unique design decisions, as discussed in Chapter 2, and have their own pros and cons. Chapter 2 exhibits that there are a variety of models which each make various assumptions regarding the environments that they are used in and the agents which use them. The models also use a variety of techniques, such as the FLSs that are used by FTE [37] or the trustor classification mechanism that is used by ETME [34], to provide strong results in an environment.

As the trust establishment models improve in performance in varying scenarios, their designs continue to increase in complexity. Although these complex designs work well in

simulations, the models can become increasingly difficult to develop and can be difficult to optimize for a target environment. ITE is an example of a model that provides strong simulated results but does so at the cost of using many hyperparameters that will be difficult to appropriately tune in practice [10]. Thus, it is important to analyze the various components and design decisions that are being used by current trust establishment models to identify similarities between them. This can allow frameworks to be designed that allow models to be more easily developed and improved. Specifically, if these trust establishment models can be defined with respect to a generalized architecture, it will become easier to adjust models to work with varying domains and to improve the performance of existing models by allowing new components to be inserted into their designs.

By proposing a robust generalized architecture, it becomes possible to provide guidelines for existing models on how to update as research progresses. Since current models focus on updating a trustee's behaviour towards trustors after transactions occur, it is possible to introduce different modules that can be used to improve the overall performance of trust establishment models. These new modules can be included in older and newer models to improve performance in areas which the models are lacking in. Specifically, Machine Learning algorithms can be used to help fine-tune the UG that a trustee will provide to a trustor in addition to the postprocessing that is being performed.

This fine-tuning can occur before a transaction is performed as a form of preprocessing at the transaction-level. Online Machine Learning algorithms, such as Hoeffding Trees, are perfect candidates to help in this scenario since the input data can be represented as either batches or data streams. This chapter will first present the generalized trust establishment model architecture which is compatible with both existing and future trust establishment models. This architecture introduces a new preprocessing module which allows a trust establishment model to achieve better performance when used in addition to the established postprocessing approaches. Using this architecture, a simple, yet robust, trust establishment model named the Generalized Trust Establishment Model (GTEM) is presented to showcase how the architecture can be used to develop a new trust establishment model. This model is designed to be simplistic to showcase how a simple model can be designed to compete with complex models. Finally, the effectiveness of this architecture is then exhibited and discussed by performing simulated tests to compare

GTEM's performance when using the new preprocessing model against GTEM's performance when performing no preprocessing and against two existing trust establishment models.

4.2 Defining a Generalized Architecture

This section presents a generalized trust establishment model architecture that can be used to produce robust trust establishment models for various domains and environments. The architecture includes a postprocessing module which is derived based on designs from existing research and a new preprocessing module which can help to improve the performance of both new and existing models. Before presenting the architecture, a high-level overview of the components that are required by standard and robust trust establishment models will be detailed. This overview provides insight into the trustor-based updates that are performed by the existing trust establishment models that have been described in Chapter 2 of this thesis. From this overview, the full generalized trust establishment model architecture, including the new preprocessing module, will be presented in detail. To showcase that this architecture can encapsulate the designs of existing trust establishment models, a final discussion will provide brief descriptions on how each existing model can be defined within the scope of the architecture.

4.2.1 A Standard Trust Establishment Model Architecture

At the top level of abstraction, a trust establishment model allows a trustee to improve its trustworthiness towards individual trustors in an environment. This process involves some method of passing input data into a learning algorithm which determines how the trustee should adjust their behaviours towards individual trustors. The learning algorithm's functionality can vary in terms of implementation, but the general process that is performed is shared between existing trust establishment models. Figure 4.1 exhibits the fundamental process of a trust establishment model in its simplest form.

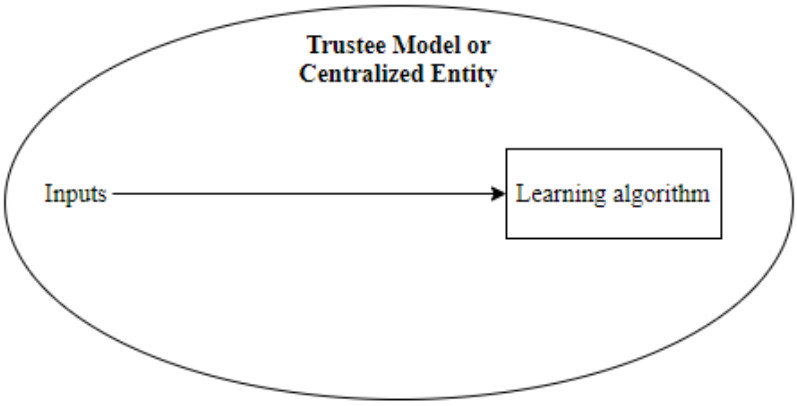


Figure 4.1 - A top-level overview of the trust establishment process

When considering current trust establishment models, both components that are seen in Figure 4.1 can vary between the trust establishment models. For example, the inputs may involve direct and indirect feedback values, or only one of the two. At its most basic level, a trust establishment model should use some form of feedback that is either directly provided by the trustor or that is predicted by the trustee, and some information on current and/or past transactions with that trustor. These are either the direct inputs that are utilized by the learning algorithm or are inputs that are passed into the learning algorithm to compute the necessary values to learn from.

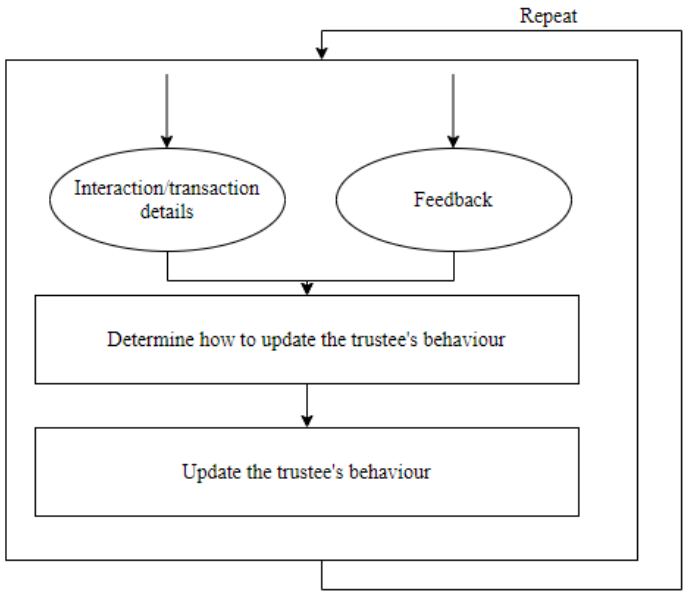


Figure 4.2 - A high-level overview of the standard trust establishment process

From these inputs, a learning algorithm will determine how to update the trustee's behaviour for individual trustors. As mentioned in section 2.2.2 of this thesis, Q-learning is the prominent technique that is utilized in MAS-related trust models. However, the learning algorithm may use additional approaches so long as it allows the trustee to learn how to establish and maintain trust with trustors in an environment. For standard trust establishment models, the learning algorithm is the process of using the inputs to determine how the trustee's behaviour should be updated in future transactions with a given trustor to improve the trustee's trust in the environment. When this update decision is made, the trustee will be able to update its behaviour towards a trustor by using an update module (assuming that it is feasible to do so). Figure 4.2 showcases a slightly lower-leveled look at the trust establishment process in its most basic form to visualize what has just been discussed. As it can be seen, the process is continuous and is repeated by the trustees based on how the update process is defined (can be done in batches or after each individual transaction). This framework follows the existing designs that are in use by established trust establishment model research, as discussed in section 2.4 of this thesis.

4.2.2 A Robust Trust Establishment Model Architecture

With a clear understanding of the general process that is taken by standard trust establishment models, the next step is to consider more robust models which consider current and future research ideas. This involves a lower-level look into the core components that more robust trust establishment models will need as research in the field progresses. Regarding the inputs, both direct and indirect feedback should be used to help improve the model's overall performance. ITE is an example of how trust establishment models can obtain better results when using the direct and indirect feedback to propose more accurate UG values for specific trustors [10]. Along with these inputs and the data regarding how the transactions are performed (such as the UG that is provided for each criterion), there may be other data to consider using. These concepts will not be explored as rigorously since they are ongoing research topics within the scope of trust establishment model research and are established research topics in the field of trust. However, these topics will be described along with their importance.

First is the relationship information between agents in the MAS(s). Although currently unused by the state-of-the-art trust establishment models, data regarding the social relationships between agents may prove to be important in more accurately adjusting a trustee's behaviours towards each trustor. This data may also be important in detecting trustor attacks, which is another topic that a robust trust establishment model will have a module for (either within or outside of the learning algorithm). Trustor attack detection is a module that should state whether a trustor has acted maliciously towards the trustee at a given time step. This may involve the trustor providing false direct feedback or by the trustor collaborating with other trustors to maximize the UG that the trustee needs to spend on the trustor. More information of specific trustor attacks is presented in Chapter 2. Regardless, a robust trust establishment model may require a built-in attack detection module, or it may require the output from an external attack detection module to be used within its learning algorithm. This module can be used to help improve how the trustee decides to adjust its behaviour towards dishonest trustors, since it may be best to penalize a dishonest trustor based on their actions or to reward honest trustors with a close relationship to a dishonest trustor to try and reduce the risk of being targeted for an attack.

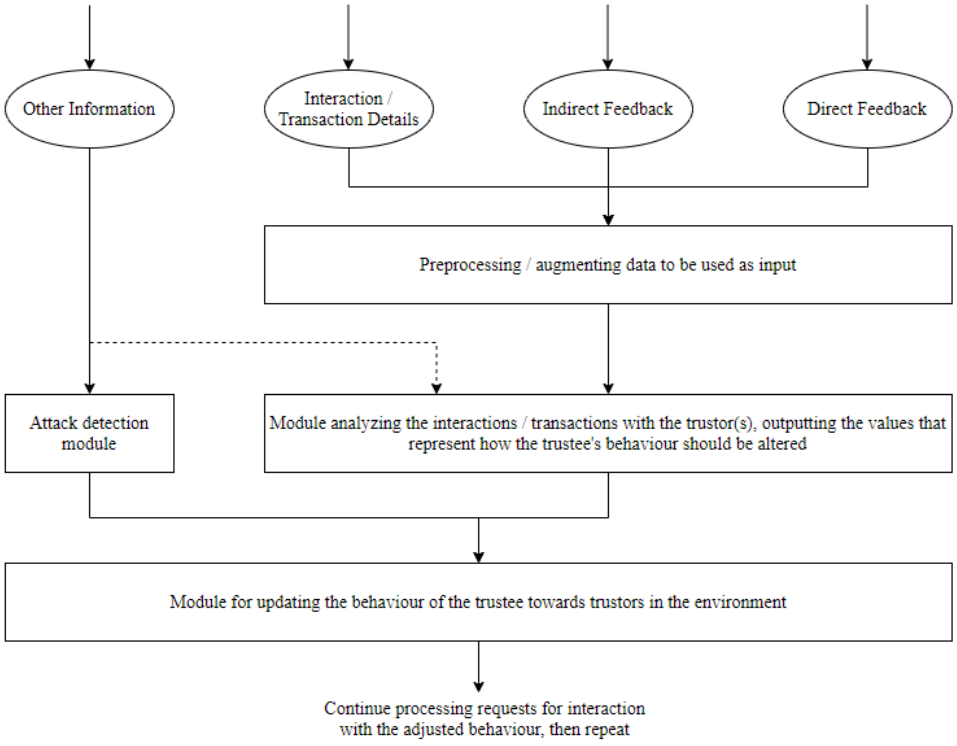


Figure 4.3 - A high-level overview of a robust trust establishment model

Keeping this in mind, it is possible to incorporate the additional modules and data into the high-level structure displayed by Figure 4.2. By allowing the robust modules and the various types of input data to be optional, the architectures that are used by all current trust establishment models can be defined with the same general structure as a robust trust establishment model. This is beneficial since it allows the modules to be added and removed based on the requirements of the trustee or the environment. Figure 4.3 showcases this integration of ideas. With the ability to use different types of input data and with the additional attack detection module, an existing architecture can more easily be updated to include more robust features over time. Although the logic may require certain updates to other submodules, the general structure of the model will not need to change. This is especially important when implementing the model. When a model is not defined with respect to a formally defined structure, the architecture may be designed in a non-

modular form. This lack of modularity will cause delays in implementing updated models and can cost the agent valuable resources that can be gained or saved with the new architecture.

4.2.3 A Generalized Trust Establishment Model Architecture

Thus far the general structure of current trust establishment models and a structure allowing the current structures that are used by trust establishment models to be compatible with future research concepts have been presented in the two previous subsections. The modules that are presented in these structures update a trustee's behaviour towards specific trustors by updating the models of the trustors that are stored by the trustee. This update process utilizes feedback from transactions and thus must take place after one or more transactions have occurred. Thus, this process of establishing trust between a trustee and specific trustors after one or more transactions will be defined as trustor-level postprocessing. Current trust establishment models establish trust using trustor-level postprocessing, which is a formal name to describe the techniques presently being performed.

Although trustor-level postprocessing has provided strong results in simulations, there are other ideas that models can use to improve their effectiveness. When considering that the current process is a form of postprocessing for each trustor, this brings forth the question as to the effectiveness of a form of preprocessing. Since postprocessing uses feedback to directly update the models of trustors, the question is then based on how preprocessing can effectively be applied in conjunction with postprocessing. The solution to this question is that the preprocessing can fine-tune the UG that a trustee proposes or provides to a trustor within a specific transaction based on input data that is related to the task being performed and based on information from previous transactions with the trustor.

This preprocessing will focus on learning how to fine-tune the UG that is proposed or provided for each transaction of a specific task before the transaction is performed. This is a stark contrast to the postprocessing which focuses on learning the desires of a trustor, rather than the general behaviours of trustors in transactions for specific tasks. Thus, this new preprocessing component can be referred to as transaction-level preprocessing. The preprocessing module fine-

tunes the UG that will be proposed or provided to a trustor for a specific transaction to provide a more robust form of improving and maintaining trust within an environment. For example, the preprocessing module may detect trends over time and will learn how to add or deduct UG from specific criteria for specific tasks to match those trends. Despite being performed for tasks that differ from trust establishment, there are many successes that have been achieved via fine-tuning, such as SenseBERT [47] which is a BERT-based architecture [48] that uses offline fine-tuning methods to improve the performance of the model for word sense disambiguation [49]. Section 4.4 of this chapter will exhibit how fine-tuning also benefits trust establishment. The following image represents the generalized trust establishment model architecture and provides a high-level look into each component of a trust establishment model, where the trustor-level postprocessing module relates to the design that is displayed in Figure 4.3.

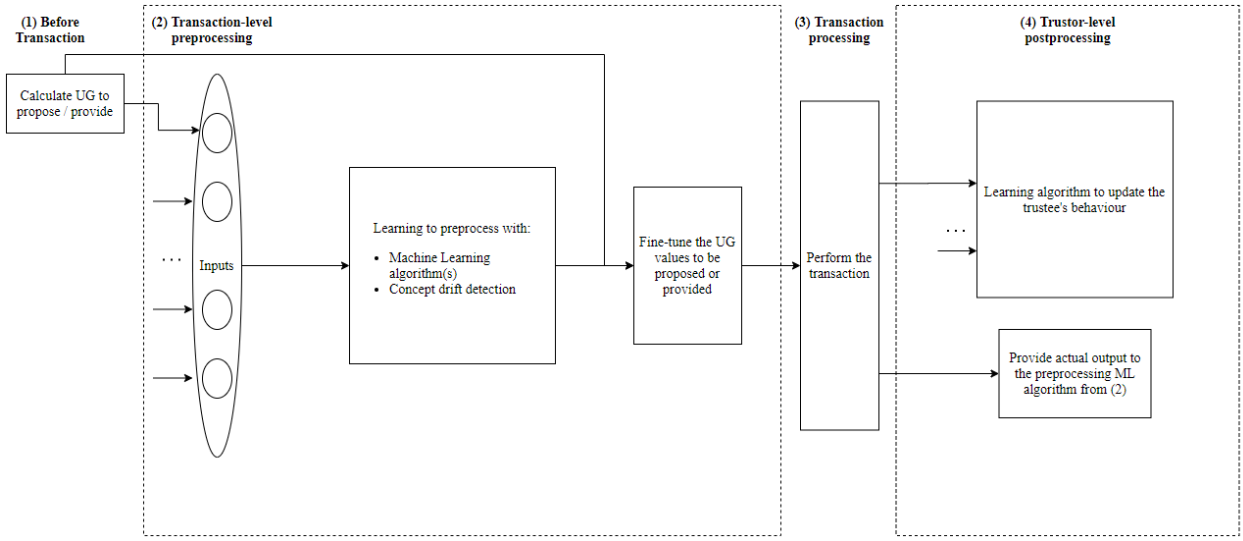


Figure 4.4 - The generalized trust establishment model architecture

As seen in Figure 4.4, a trust establishment model will first compute the UG that the trustee should propose or provide to a trustor. This calculation process can be designed for trustees that have unlimited resources or for trustees with limited resources for single- or multi-criteria tasks in an environment. When the UG is computed, a trust establishment model may or

may not utilize the transaction-level preprocessing to fine-tune the UG for the task in question. This preprocessing should use a learning algorithm, where Figure 4.4 explicitly states the use of a Machine Learning algorithm, that can learn how to predict values that can later be derived from the output of the transactions.

The Machine Learning algorithm that is selected can be a categorical model or a regression model depending on the design of the module which updates the UG. For example, a discrete classifier's prediction can be used to simply state whether a transaction will satisfy all the needs of a trustor excessively, will satisfy all the trustor's needs accurately, or will not satisfy all the trustor's needs. The preprocessing module can then fine-tune the UG based on the prediction from the Machine Learning classifier. Alternatively, a regression model can output a numerical value that can be used as a variable to more precisely fine-tune the UG that will be provided for a specific task. This predicted numerical value can be the prediction of the direct feedback that will be provided by the trustor or be the prediction of some other metric that will be calculated in the postprocessing module.

Although there are many types of Machine Learning algorithms, when considering which to use for the preprocessing component it is important to consider the method of which the data will need to be processed by. In the case of open and dynamic MASs, there is no upper bound on the number of transactions that can occur for a specific trustee at a given time step. Furthermore, the algorithm must use this data as it becomes available to continuously improve its overall performance. Due to these factors, Online Machine Learning algorithms, which have been discussed in section 2.2.4 of this thesis, are the optimal algorithm choice when performing this preprocessing. Although there are many different types of Online Machine Learning algorithms, one algorithm that will be used in the trust establishment model that is presented in the next section of this chapter is the Hoeffding Tree algorithm. This is a simplistic algorithm which will work well for the simplistic trust establishment model that is presented.

Another benefit of this preprocessing component is that it can provide concept drift detection for transactions of specific tasks. In trust establishment, dealing with shifting behaviours can be tricky on both the trustor-level and the transaction-level. By using these concept drift detection algorithms, which are discussed in section 2.2.5 of this thesis, it is possible for trustees to detect shifts in the general expectations of trustors for various tasks and to

ensure that any Machine Learning model being used can maintain accurate predictions despite the new behaviours that are introduced. Although unused in the models which are discussed in this chapter, it is important to note that these algorithms can be integrated into the preprocessing component of the architecture to better prepare for behaviour shifts by trustors or for changes with general trends in the environment at a given time step for a specific task.

The concept drift detection, Online Machine Learning algorithm(s), and fine-tuning process all lead to a final UG value that will be proposed or provided to the trustor (depending on when the model decides to perform the preprocessing). Following the preprocessing, the transaction between the trustee and the trustor will occur and the trustee will then proceed to perform trustor-level postprocessing, as expected by a trust establishment model. This trustor-level postprocessing can occur in many ways but will always result in the trustee updating its behaviours towards specific trustors. Whether through just Q-Learning or through a combination of Q-Learning and another set of modules (see Figure 4.3), this will result in the trustee establishing trust in the environment while attempting to balance the average UG that is spent to maintain a satisfiable level of trust.

These modules cumulate into an architecture that both existing and new trust establishment models can use to generalize the design process and to improve general performance via the preprocessing module. It also becomes possible to design different versions of trust establishment models for trustees of differing capabilities. For example, if the preprocessing module is too computationally intensive for a specific trustee while being manageable for other trustees, a developer can design a 'lite' version of the trust establishment model that provides all the same functionality without the preprocessing components. Similarly, it becomes easier to plug in additional modules that may be needed to optimize a trustee's performance as future research is performed in the field. Furthermore, a generalized architecture can allow modular modules to be available for different trust establishment models to use. Perhaps each model will handle different assumptions in the same way and can simply reuse existing modules from other models. This can save development time while providing the option to use open-source modules that are thoroughly tested by members of the research community (as they become more readily available over time).

4.2.4 Existing Model Designs Within the Generalized Architecture

To showcase that the generalized trust establishment model architecture can encapsulate the designs of existing trust establishment models, high-level descriptions will be provided which outline how each of the trust establishment models that have been described in section 2.3 of this thesis can be defined by the architecture's postprocessing module. The individual trust establishment models will not be discussed in detail; however, the brief discussions will outline how a model's definition fits within this architecture. This will help to showcase that the architecture can define the existing models, which, in turn, highlights that a preprocessing module is capable of being designed to fit the postprocessing being performed.

First, the trust establishment model that is presented in [33] will be discussed. This model focuses on using reputational incentives as a method to determine how a trustee should adjust its behaviours towards a specific trustor. By modeling the reputational incentives to be gained and lost from a transaction, the trustee becomes able to determine how they should act towards specific trustors during future transactions. These values can be computed following one or more transactions to accurately derive these reputational incentives. Thus, by using trustor-based data following transactions that have occurred, this model can adjust the trustee's behaviours to balance the trustee's trust with the UG provided. This clearly fits into the postprocessing module since the module is defined to use this data for specific trustors to update the trustee's behaviours for future transactions with those trustors.

ETME is another model that can clearly fit into the postprocessing module [34]. This model uses the satisfaction that is received following a transaction to determine whether to reclassify a trustor with a different classification label. Since each trustor is assigned a classification label which the trustee uses to determine the UG to provide to the trustor, this update which uses a trustor's direct feedback following a transaction directly fits within the postprocessing module. The update always occurs after a completed transaction by using direct trustor feedback, where the resulting classification value is then used to adjust the trustee's behaviours towards specific trustors during future transactions.

Following ETME, the trust establishment model that is proposed in [35] will be analyzed. This is a model which many of the future models expand upon. As discussed in section 2.4.3,

this trust establishment model uses the direct feedback that is obtained by trustors to help adjust the trustee's behaviours towards those trustors. Although not a robust model, this design is exactly what the postprocessing module encapsulates. By requiring direct feedback after a transaction occurs to make the updates to the trustee's behaviours, this model fits into the postprocessing module's design.

Each of the remaining models follow the same general logic as the model previously discussed. RLTE, which is the model that the remaining models to be discussed are based on, follows a similar design in which trustor feedback regarding specific transactions with trustors is collected and used to update the trustee's behaviours towards those specific trustors [36]. FTE [37], ATeIF [38], and ITE [10] each follow the same structure but use different processing techniques and potentially different types of feedback (direct and/or indirect feedback). Thus, each of these models follow the design presented by the postprocessing module and can hence be defined by that module within the generalized trust establishment model architecture.

From these high-level descriptions, it has been displayed that each of the models discussed within Chapter 2 are capable of being defined by the trustor-level postprocessing module within Figure 4.4. Since none of these models use research ideas such as trustor attack detection or the social information of agents in the environment, the postprocessing module's design is set to still allow the models to incorporate these ideas in the future. Since each model is defined by the postprocessing module, a preprocessing module can also be designed for each model to help enhance the model's performance. This results in the architecture robustly covering existing designs and allowing for performance benefits to be added via preprocessing. Even if the models are working under differing assumptions, such as non-instantaneous transaction completion, the architecture can still cover a model's general designs. The models can then operate normally under these varying scenarios by applying their postprocessing techniques as the transactions finalize and as the trustee receives the satisfaction values from trustors (or this can be predicted if not received). Similarly, if a trustee has limitations on the amount of UG that can be provided for each criterion of a task, the trust establishment model can provide the trustee with the suggested UG to provide to a trustor, where the trustee can then adjust these suggestions based on their limitations. Thus, these varying ideas and designs are all still supported by the postprocessing module.

4.3 A Generalized Trust Establishment Model

Following the proposal of a generalized trust establishment model architecture, a new trust establishment model will be defined by using the proposed architecture. Since the model uses this generalized architecture, it is named the Generalized Trust Establishment Model. GTEM is designed to be a simple, Reinforcement Learning-based trust establishment model that utilizes fundamental ideas from pre-existing trust establishment models in its postprocessing module and introduces functionality for the optional preprocessing module that will improve its performance when used. This section will explore GTEM's architecture and will provide all the details for its implementation of the postprocessing and preprocessing modules.

4.3.1 GTEM's Postprocessing Module Design

GTEM's postprocessing module is designed to be simplistic and easy to modify. The general structure and fundamental formulas are inspired by ITE's design [10]. Thus, some of the equations that are specified in this section will be what has been discussed in section 3.2.2 of this thesis. The fundamental difference between any shared equations exhibited in this section is that they will be using GTEM's hyperparameters, which may use differing values. To be simpler than ITE, which contains many hyperparameters, GTEM uses a classification-based system to determine how trustees should update their behaviours towards specific trustors. The classification system involves assigning each trustor that has interacted with the trustee for a specific task a label, defined by $t_x^y(s)$. This label represents how the trustee views a trustor's behaviour, similarly to how ETME has trustees classify the type of each trustor [34]. The equation to calculate $t_x^y(s)$ will be defined later in this section, however it is important to note that GTEM considers trustors to be either overcompensated (represented by 0) or undercompensated (represented by 1) at a given time step. An overcompensated trustor is considered to be receiving too much UG for a specific task and an undercompensated trustor is considered to be receiving too little UG for a specific task. Although more labels can be used to perform more specific updates, these two allow trustees to continuously balance the UG values that are provided to trustors.

Similar to some previous trust establishment models, such as ITE, GTEM assumes that each trustor contains values which represent their expectations on the UG amount to be provided for each criterion of each task. This is the rw value[1] and is what GTEM attempts to predict to ensure that a provided UG is enough to satisfy a trustor's needs for each criterion of a task s_{c_i}.

$$rw_x(s_{c_i}) = \frac{w_x(s_{c_i})}{d_x(s_{c_i})} \tag{4.1}$$

where,

- $w_x(s_{c_i})$ is the weight value assigned to criterion c_i of task s by trustor x
- $d_x(s_{c_i})$ is trustor x's demand for the criterion c_i of task s

Since the $rw_x(s_{c_i})$ values represent trustor x's expectations from transactions for the corresponding criteria of each task, GTEM will need to predict these values to accurately provide UG values that can increase or maintain trust with the trustor while not providing excessive UG values. To predict these values, GTEM stores predicted rw values within models of each trustor x, stored by trustee y, for each criterion of each task. These predicted values, denoted by rw_x^y, are updated with the following equation after being initialized at random.

$$rw_x^y(s_{c_i}, req) = \begin{cases} (1-\alpha) * rw_x^y(s_{c_i}, tra'), & \text{if } t_x^y(s) = 0 \\ (1+\alpha) * rw_x^y(s_{c_i}, tra'), & \text{if } t_x^y(s) = 1 \end{cases} \tag{4.2}$$

where,

- $rw_x^y(s_{c_i}, req)$ is the predicted rw value by trustee y for trustor x for the criterion c_i of task s for interaction *req*, which is the interaction following the previous transaction *tra'*
- $rw_x^y(s_{c_i}, tra')$ is the predicted rw value from the previous transaction *tra'*

[1] The relative weight value displayed by equation 4.1 is also presented in Chapter 3's equation 3.3

- α is the update rate that is used by various functions within GTEM, where $0 \leq \alpha \leq 1$
- $t_x^y(s)$ is the classification label assigned to trustor x by trustee y for task s at the current time step, where $t_x^y(s) = 0$ when the trustor is considered to be overcompensated and $t_x^y(s) = 1$ when the trustor is considered to be undercompensated.

This update function for the predicted rw values showcases the simplicity of GTEM's functionality. By using a single variable to define the learning rate (which is also used by equation (4.5)), GTEM can reasonably have α's value be optimized based on the expected behaviours to be found in the target environment. Furthermore, this update process follows a logical assumption, that is, when a trustor is considered to be overcompensated (which implies that the trustor is satisfied), the rw_x^y values for the criteria of a specific task are reduced by some percentage. Similarly, if the trustor is considered to be undercompensated, some percentage is added to the rw_x^y values for the criteria of a specific task.

Using the same fundamental process as ITE, with a different update function for the implicit improvement values, GTEM calculates the UG values to propose for each criterion of a task, for a specific trustor, by using an improvement function which incorporates the explicit and implicit feedback which are obtained from a trustor [10]. Thus, this improvement calculation is computed via the weighted combination of an explicit improvement formula which utilizes the rw_x^y values that represent the corresponding criteria weights and an implicit improvement equation which is updated after each transaction with a trustor. To begin, the explicit improvement function[2] is defined below.

$$E\_Imp_x^y(s_{c_i}, req) = rw_x^y(s_{c_i}, req) * MaxImp^y + MinUG^y(s_{c_i}) \quad (4.3)$$

where,

[2] The explicit improvement function displayed by equation 4.3 is also presented in Chapter 3's equation 3.5

- $E\_Imp_x^y(s_{c_i}, req)$ is the explicit improvement value to be used for computing the UG value of a specific criterion of a task s_{c_i} by trustee y for trustor x for the specific interaction req, where

$$0 \leq E\_Imp_x^y(s_{c_i}, req) \leq MaxImp^y(s_{c_i})$$

- $rw_x^y(s_{c_i}, req)$ is the predicted rw value that y has set for x for s_{c_i} at the time step of the current interaction req

- $MinUG^y(s_{c_i})$ is the minimum UG value that y can offer for s_{c_i}

- $MaxImp^y(s_{c_i})$ is the maximum improvement value[3] that can be applied to the UG proposed by y to x for s_{c_i}, where

$$MaxImp^y(s_{c_i}) = MaxUG^y(s_{c_i}) - MinUG^y(s_{c_i}) \quad (4.4)$$

  - Where $MaxUG^y(s_{c_i})$ is the maximum UG value that y can offer for s_{c_i}

As seen above, the calculated explicit improvement value is dependant on the rw_x^y value for the specified criterion, as described in equation (4.2), and will be changed depending on how the corresponding rw_x^y value is updated with the results from a transaction with the trustor. With the explicit improvement defined, the implicit improvement function is defined below. This value is stored by a trustee for each criterion of each task, for each trustor. These implicit improvement values are updated after each transaction with the corresponding trustor for the specified task.

$$I\_Imp_x^y(s_{c_i}, req) = \begin{cases} I\_Imp_x^y(s_{c_i}, tra') - \alpha, & \text{if } t_x^y(s) = 0 \\ I\_Imp_x^y(s_{c_i}, tra') + \alpha, & \text{if } t_x^y(s) = 1 \end{cases} \quad (4.5)$$

where,

[3] The maximum improvement value in equation 4.4 is also used in Chapter 3 via equation 3.6

- $I\_Imp_x^y(s_{c_i}, req)$ is the implicit improvement to be provided by trustee y to trustor x for s_{c_i} during the next interaction req between x and y, where

$$0 \leq I\_Imp_x^y(s_{c_i}, req) \leq MaxImp^y(s_{c_i})$$

- $I\_Imp_x^y(s_{c_i}, tra')$ is the implicit improvement value that trustee y has calculated for trustor x for s_{c_i} in the previous transaction tra' between x and y

- α is the learning rate variable that is used throughout GTEM to represent how drastic changes should be made within the environment, where the recommended, but not required, value is $0 \leq \alpha \leq 0.2$ since increased values will result in changes that are too drastic within the program (see equation (4.2) for more details regarding α)

As exhibited by equations (4.2) and (4.5), GTEM utilizes a single parameter to represent the general rate of learning within the environment. Although each equation can easily contain specific reward and penalty variables, by replacing each α instance with a unique variable, GTEM utilizes a single variable to showcase that a trust establishment model can maintain strong results with a simplistic design. Compared to generalizing α, which may provide more optimal results if appropriately tuned, using a standalone α allows the model's hyperparameters to be tuned quicker. This also serves as an example of how the various modules and equations within a trust establishment model can be modified for various scenarios with relative ease when appropriately structured and defined.

With the implicit improvement defined to increment or decrement itself by some amount after each transaction with a trustor, depending on the type which the trustor is classified as by the trustee, the improvement function can be defined as a weighted combination of the explicit and implicit improvement functions. This improvement value[4] represents how much more UG the trustee should propose towards the trustor for a specific criterion of a specific task, beyond the minimum amount that can be provided for that criterion.

[4] The total improvement function in equation 4.6 is also used in Chapter 3's equation 3.8

$$Improvement_x^y(s_{c_i}, req) = \omega * E\_Imp_x^y(s_{c_i}, req) + (1 - \omega) * I\_Imp_x^y(s_{c_i}, req) \quad (4.6)$$

where,

- $Improvement_x^y(s_{c_i}, req)$ is the total UG improvement to be provided in addition to $MinUG^y(s_{c_i})$ for interaction req, where
$$0 \leq Improvement_x^y(s_{c_i}, req) \leq MaxImp^y(s_{c_i})$$
- ω represents the weight distribution between the explicit and implicit improvement values such that the maximum improvement value is $MaxImp^y(s_{c_i})$, where $0 \leq \omega \leq 1$

With the improvement formula defined, GTEM can use this value to calculate how much UG to provide a trustor for each criterion of each task, represented by ug_x^y. The total UG provided by the trustee is the summation of the ug_x^y values, for each criterion. Both of these equations are defined below[5] and are proposed to a trustor to see whether the trustor will end up rejecting the interaction with the trustee.

$$ug_x^y(s_{c_i}, req) = Improvement_x^y(s_{c_i}, req) + MinUG^y(s_{c_i}) \quad (4.7)$$

- Where $ug_x^y(s_{c_i}, req)$ is the UG value to be provided to trustor x by trustee y regarding s_{c_i} for the interaction req, where $MinUG^y(s_{c_i}) \leq ug_x^y(s_{c_i}, req) \leq MaxUG^y(s_{c_i})$

Each $ug_x^y(s_{c_i}, req)$ value is then used to compute the total UG proposed for task s via the equation below.

$$UG_x^y(s, req) = \sum_{i=1}^{p} ug_x^y(s_{c_i}, req) \quad (4.8)$$

where,

[5] Equations 4.7 and 4.8 are both related to the UG equation from Chapter 3 (equation 3.9)

- $UG_x^y(s, req)$ is the total UG proposed by trustor x to trustee y for task s during the interaction req
- p represents the total number of criteria within task s

These UG values are then used when a trustor requests to interact with the trustee. The trustee computes the values as seen in equations (4.7) and (4.8), then proposes these amounts to the trustor. If these are acceptable the trustor will not reject the interaction, which stops the transaction from occurring, and the trustee performs the task with either the same or different UG values. An honest trustee will use the same UG values that GTEM computes, but a dishonest trustee may trick a trustor by providing much less than what has been proposed. Throughout this chapter, trustees are honest and generally provide the same UG that they propose to the trustor to attempt to become trustworthy in the environment. The exception to this is that the preprocessing will slightly alter the UG values that are provided to the trustor after the initial amount is proposed. As detailed in the following subsection, these UG updates are minor and serve to better match the needs of the trustors.

After a transaction, the trustor may, or may not, provide direct feedback to the trustee regarding the UG that has been received from the completed transaction. If this value is not provided, it can be predicted by the trustee to ensure that direct feedback is available. This feedback is the trustor's satisfaction with the trustee and is the same type of feedback that has been discussed in Chapter 3. Since GTEM uses the same relative weight concept as ITE, the following SAT equations[6] (equations (4.9) and (4.11)) are the same used by ITE [10].

$$SAT_x^y(s, tra) = \sum_{i=1}^{p} \frac{w_x(s_{c_i}) * ug_x^y(s_{c_i}, tra)}{d_x(s_{c_i})} \tag{4.9}$$

where,

[6] The trustor satisfaction equations 4.9 and 4.11 are the same as those from Chapter 3's equations 3.10 and 3.11

- $SAT_x^y(s, tra)$ is the total satisfaction of trustor x regarding the transaction tra with trustee y
- The normalized SAT value between 0 and 1 is calculated by dividing $SAT_x^y(s, tra)$ by the trustor's maximum possible SAT value, denoted by $Max\_SAT_x(s, tra)$, where

$$Max\_SAT_x(s, tra) = \sum_{i=1}^{p} \frac{w_x(s_{c_i})}{d_x(s_{c_i})} \tag{4.10}$$

If the maximum possible ug_x^y values for each criterion are larger than one, which they are not in this chapter, the normalized SAT value will be set to one if its value is greater than one. Although unused within this chapter, the SAT value can also be predicted by the trustee. This is important since not every trustor will provide feedback to the trustee and some trustors may provide false feedback to a trustee to trick the trustee to provide more UG in the future. This predicted SAT value is defined below and can be computed by the trustee after each transaction tra.

$$SAT_x^y(s, tra) = \sum_{i=1}^{p} rw_x^y(s_{c_i}) * ug_x^y(s_{c_i}, tra) \tag{4.11}$$

The predicted SAT value can also be normalized by calculating $Max\_SAT_x^y(s, tra)$ with the following equation and dividing the SAT value by this maximum SAT value.

$$Max\_SAT_x^y(s, tra) = \sum_{i=1}^{p} rw_x^y(s_{c_i}) \tag{4.12}$$

These predicted SAT values utilize the predicted rw values since the actual weight and demand values that are used in equations (4.9) and (4.10) are not known by the trustee.

GTEM can now utilize the SAT values that are obtained by the trustor or that are predicted by the trustee to update the classification of the trustor. Recall that a trustor is

classified as overcompensated when $t_x^y(s) = 0$ and is classified as undercompensated when $t_x^y(s) = 1$. The trustor is classified according to the equation below for the corresponding task in the environment after a transaction is performed between the trustee and the trustor for that task. This equation utilizes the SAT value that has been obtained from the completed transaction and the total percentage of UG that has been provided for the transaction to determine how the trustor should be classified until their next completed transaction with the trustee.

$$t_x^y(s, req) = \begin{cases} 0, & \text{if } SAT_x^y(s, tra) \geq \gamma \text{ and } UG\_Percent_x^y(s, tra) \geq \lambda \\ 1, & \text{else} \end{cases} \quad (4.13)$$

where,

- $t_x^y(s, req)$ is the trustor classification assigned by y to x for task s regarding the next interaction req
- γ represents the threshold value for defining what SAT percentage should be considered as sufficient for a transaction, where $0 < \gamma \leq 1$
- λ represents the threshold value for defining the percentage of UG that should be considered as high, where $0.5 \leq \lambda \leq 1$
- $SAT_x^y(s, tra)$ represents the normalized SAT value obtained from the previous transaction between x and y for task s, denoted by tra
- $UG\_Percent_x^y(s, tra)$ is the percentage of UG provided by x to y for task s in the previous transaction tra, where

$$UG\_Percent_x^y(s, tra) = \frac{UG_x^y(s,tra)}{MaxUG^y(s)} \quad (4.14)$$

  - Where $\frac{MinUG^y(s)}{MaxUG^y(s)} \leq UG\_Percent_x^y(s, tra) \leq 1$

Thus, equation (4.13) defines a trustor as overcompensated if the trustor has provided the trustee with a sufficiently high SAT value for the previous transaction and if the trustee has

provided a large amount of UG towards the trustor for the previous transaction. Otherwise, the trustor is classified as undercompensated. Although simple, this method of classification follows the logical assumption that a trustor that is satisfied and that receives a larger amount of UG should receive less UG since the decrease should have little impact on the overall trust and that the trustor should otherwise receive a larger UG value to increase the trustor's trust towards the trustee. This classification process can be designed in a more robust manner with more trustor classification types, but this model will only use two to showcase how this simplistic classification approach can work in coordination with the preprocessing module to achieve better results than without the preprocessing module.

4.3.2 GTEM's Preprocessing Module Design

Following the complete definition of GTEM's trustor-level postprocessing module, the transaction-level preprocessing module used by GTEM will now be defined. This preprocessing module attempts to fine-tune the UG values for the criteria of a specific task that are provided to a trustor (see equation (4.7)). Thus, the preprocessing module learns the general trends for a specific task, rather than for individual trustors. Since GTEM uses trustor classification to determine how it should adjust a trustee's behaviour, as seen in equations (4.2) and (4.5), this preprocessing module will operate by utilizing an Online Machine Learning algorithm to predict what the trustor will end up classified as following a transaction, before the transaction occurs. This allows GTEM to appropriately fine-tune the UG that will be provided depending on what trustor type its Machine Learning model predicts the trustor to be. Since each task will use its own Machine Learning model, this also allows each of the Machine Learning models to learn which input values relating to transactions of specific tasks, such as the UG values that are being provided for each criterion (equation (4.7)), are generally more important at accurately predicting which trustor class will be assigned to a trustor after the transaction.

GTEM's preprocessing module uses the Hoeffding Tree Online Machine Learning algorithm that has been discussed in Chapter 2 and section 4.2.3 of this thesis. This algorithm is selected due to its simplicity, which is important for GTEM's goal of using a simple, yet effective design. Since each task s will contain its own Hoeffding Tree model, represented as

ϕ_s, each model will also accept a potentially varied set of input parameters. The input parameters that are used by ϕ_s are represented by the values within the set $Inputs_x^y(s, tra)$, where $Inputs_x^y(s, tra)$ contains the SAT values which have been obtained from the past two transactions between the trustor x who has accepted the transaction tra and trustee y. These SAT values are defined as $SAT_x^y(s, tra')$ for the previous transaction tra' between x and y and by $SAT_x^y(s, tra'')$ for the transaction between x and y directly preceding tra', named tra''. Although these two inputs are shared by all ϕ_s models, $Inputs_x^y(s, tra)$ also contains each $ug_x^y(s_{c_i}, tra)$ value that will be provided to the trustor by the trustee for the transaction tra. Since the number of criteria will vary per task, this results in a potentially different number of inputs for each ϕ_s model. Although other inputs can be used, the input choice defined above and displayed below (equation (4.16)) are selected since they are easy to retrieve and all directly relate the task in question (since ϕ_s must learn patterns regarding the specified task). Note that if there have not yet been two previous transactions to obtain the previous SAT values from, the preprocessing module is not used for that transaction. From the set of inputs, ϕ_s will return a prediction on the trustor type which will be assigned to the trustor following the transaction, as exhibited below (view equation (4.13) to see how the actual trustor type value is assigned).

$$p\_t_x^y(s, tra) = \phi_s(Inputs_x^y(s, tra)) \tag{4.15}$$

where,

- $p\_t_x^y(s, tra)$ is the predicted t_x^y value that is output by ϕ_s for the transaction tra between trustee y and trustor x for task s, where $p\_t_x^y(s, tra) = 0$ when the trustor x is predicted to be overcompensated and $p\_t_x^y(s, tra) = 1$ when the trustor x is predicted to be undercompensated
- $Inputs_x^y(s, tra)$ is the set of input values between y and x for the transaction tra, where

$$Inputs_x^y(s, tra) = \{ SAT_x^y(s, tra'), SAT_x^y(s, tra''), ug_x^y(s_{c_1}, tra), \ldots, ug_x^y(s_{c_p}, tra) \} \tag{4.16}$$

  - Where p represents the total number of criteria for task s

Once $p\_t_x^y(s, tra)$ is obtained, GTEM fine-tunes the UG values that will be provided to trustor x after those UG values have been proposed to the trustor (if the interaction has not been rejected). The UG values are updated after proposing the initial UG values to ensure that the trustor already trusts the trustee enough to accept the proposed amount and that the trustee will know whether making the specified small updates will have a significant impact on their trust from the trustor (from the obtained SAT value following the transaction). Rather than updating each criterion value based on the prediction, GTEM only updates k ug_x^y values depending on the predicted trustor classification value. Thus, two collections of k ug_x^y values are created and are defined as the lists $C\_L_x^y(s, tra)$ and $C\_H_x^y(s, tra)$ which are described below.

- Let $C\_L_x^y(s, tra)$ represent a collection of the k lowest UG values from the criteria in task s to be provided by trustee y to trustor x in the transaction tra (each calculated by equation (4.7))
- Let $C\_H_x^y(s, tra)$ represent a collection of the k largest UG values from the criteria in task s to be provided by trustee y to trustor x in the transaction tra (each calculated by equation (4.7))

Using these two sets of ug_x^y values, each $ug_x^y(s_{c_i}, tra)$ value that will be provided by the trustee to the trustor in transaction tra will be updated via the following formulas. Note that equation (4.17) increases the k lowest UG values when the trustor is predicted to be undercompensated and equation (4.18) decreases the k highest UG values when the trustor is predicted to be overcompensated.

$\forall ug_x^y(s_{c_i}, tra) \in C\_L_x^y(s, tra)$:

$$ug_x^y(s_{c_i}, tra) = \begin{cases} ug_x^y(s_{c_i}, tra) + \zeta, & \text{if } ug_x^y(s_{c_i}, tra) + \zeta \leq MaxUG^y(s_{c_i}) \\ & \text{and } p\_t_x^y(s, tra) = 1 \\ ug_x^y(s_{c_i}, tra), & \text{else} \end{cases} \quad (4.17)$$

$\forall ug_x^y(s_{c_i}, tra) \in C\_H_x^y(s, tra)$:

$$ug_x^y(s_{c_i}, tra) = \begin{cases} ug_x^y(s_{c_i}, tra) - \zeta, & \text{if } ug_x^y(s_{c_i}, tra) - \zeta \geq MinUG^y(s_{c_i}) \\ & \text{and } p\_t_x^y(s, tra) = 0 \\ ug_x^y(s_{c_i}, tra), & \text{else} \end{cases} \quad (4.18)$$

where,

- $p\_t_x^y(s, tra)$ is the predicted trustor classification type that is obtained directly before the two update equations above are executed
- ζ is an update variable which represents the amount of fine-tuning to be performed, either negatively or positively, to the UG values that are provided for a transaction, where the recommended range is $0 < \zeta \leq 0.15$ since the fine-tuning process should use smaller adjustments rather than larger adjustments

Equation (4.17) is set such that if a trustor is predicted to be classified as overcompensated based on a set of input values, the trustee should remove a small amount from the k largest UG values that will be provided. Similarly, equation (4.18) adds a small amount to the k smallest UG values that will be provided when a trustor is predicted to end up as undercompensated following the transaction, if given the current UG values. This ensures that the changes made are not too large and that the changes minimize the possibility of providing too much for specific criteria or too little for other criteria. This process represents the core purpose behind GTEM's preprocessing and exhibits how a preprocessing module can be designed for a specific trust establishment model by using the generalized architecture that is proposed in section 4.2.3 of this chapter.

The final step of the preprocessing module involves training ϕ_s with the actual trustor classification value that is obtained by the postprocessing module through equation (4.13) after the postprocessing module finalizes updating the states of the trustor models that are contained by the trustee. After a transaction is complete and the model calls the postprocessing module, ϕ_s is trained with the input data that has been used to predict the trustor's type during the preprocessing and the actual classification tag that is assigned to the corresponding trustor. This

will lead to the model more accurately predicting $p\_t_x^y$ values for the specified task (see equation (4.15)) and appropriately fine-tuning the UG values that are being provided to trustors.

Although GTEM can use concept drift detection algorithms to better support the Online Machine Learning algorithm being used when there are shifting trustor behaviours in an environment, these algorithms remain unused since the simulations that are run in the following section use diverse, but static trustor behaviours. Thus, although the addition of a concept drift detection algorithm to be used prior to obtaining the predicted trustor classification values can be useful in other environments, it will not provide any performance benefits within this environment. To use a concept drift detection algorithm in this scenario, it can be used directly before a prediction to determine whether the Machine Learning model that is being used should be updated to account for the detected shift in trustor behaviours for a specific task.

4.4 Architecture Evaluation

With the generalized trust establishment model architecture defined and explained in section 4.2 and with GTEM being defined with respect to the proposed architecture in section 4.3, this section will evaluate the generalized trust establishment model architecture by comparing GTEM to two other trust establishment models and by comparing GTEM's performance with and without the use of its preprocessing module. The results from these comparisons will be displayed after explaining the simulation environment that is used and the metrics that are tracked throughout the simulations.

4.4.1 Simulation Setup

The simulation definition which is used in this chapter for the comparative analysis of the selected trust establishment models follows the general structure that is described within section 2.2.1 and which is used within Chapter 3. Although these simulations can be set to emulate distributed MASs by splitting the agents into different MASs and running the simulations with an appropriate method of communication, these simulations run within a single MAS. Since the

trust establishment models can work in single or distributed MASs, given the correct trust management module components, this will have no effect on the overall results which are obtained from the tests.

For these simulations there will be four different trust establishment models that are used for comparison. The first model is GTEM with the postprocessing and preprocessing modules. This model showcases the potential of using the preprocessing module that has been presented in the generalized trust establishment model architecture alongside the traditional postprocessing module. To exhibit the difference made by including a preprocessing module to a model that is designed with a specific postprocessing module, a version of GTEM without the preprocessing module is used as the second model. This model will be referred to as GTEM-Lite and is the same as GTEM with the omission of the preprocessing module. GTEM-Lite is an example of how the generalized trust establishment model architecture allows a trust establishment model to include or exclude certain components with relative ease, allowing for simpler or more robust versions of a model to be available for agents to use. The third model that will be used is ITE, which uses the same hyperparameter setup that is provided within its paper[7], except with the unengagement threshold variable θ value set 0.5 and the engagement threshold variable Ω set to 0.75 to better match the bounds for the variables as they are defined in its paper [10]. The final model that will be simulated is ETME, which uses the same hyperparameter setup found within its paper [34].

The environment is defined to be the same for each of the four models, with some modifications made when running with ETME. Specifically, the environment has been set for all models other than ETME to run with one task that contains ten criteria whereas ETME is set to run with one task that contains one criterion to match the model's capabilities. Otherwise, the models will run in identical simulations, where each result can be reproduced via a selected random seed. This allows each simulation to run identically with the only difference being the trust establishment model that is used in the simulation. Each simulation is set such that each trustee uses the same type of trust establishment model, and each trustor uses the same type of trust evaluation model. The communication between agents and the completion of tasks is also instantaneous within the environment. Furthermore, the simulation environment is set to run

[7] ITE's general parameter setup is also available in Table 3.1 from Chapter 3

through ten different simulations for each of the three tests that are performed for each of the four models. Since the models are being compared on their general performance and have varying designs, this helps to provide a better understanding of how they will generally perform in an environment. These ten unique simulations are defined by changing the random seed selection for each run, with each model and test using the same ten seeds. The ten simulations then have their corresponding metrics averaged to display accurate results for each model and test.

Similar to Chapter 3, the environment, the agents, and each of the trust establishment models are programmed using Python's Mesa Multi-Agent Modelling library [43], rather than in Java with the MASON library [42]. The Hoeffding Tree Online Machine Learning algorithm that is used by GTEM's preprocessing module is implemented via Scikit-multiflow's Python implementation [50].

Following a similar design to the environment used in Chapter 3, each trustor is assigned an activity level and a demand level when initialized. The activity level represents the likelihood of a trustor deciding to interact with a one or more trustees at a given time step. A trustor with a low activity level has a low chance to interact with trustees at a specific time step, a trustor with a high activity level has a high chance to interact with trustees at a specific time step, and a trustor with a regular activity level has a chance to interact with trustees at a given time step based on the difference from the low and high activity level probabilities that are selected. Therefore, at each time step for these simulations a trustor chooses to interact with no trustees or with a random number of the trustees, between 1 and 30, that the trustor considers to be most trustworthy out of all trustees in the environment. The sole exception to this is for the first five time steps, during which the trustors interact with each trustee to begin to understand how trustworthy each trustee is.

The demand level that is assigned to each trustor represents how much UG a trustor expects to receive from trustees in interactions for specific tasks. When an agent is initialized as a trustor, the demand values that are used for computing the relative weight values for each criterion of a task, see equation (4.1), are set based on random selections from pre-defined ranges of values which each relate to a specific demand level. The three trustor demand types that are assigned to trustors are high demanding trustors which require a high percentage of UG in each

criterion of a task to be satisfied, low demanding trustors which require a low percentage of UG in each criterion of a task to be satisfied, and regular demanding trustors which are satisfied with a percentage of UG in each criterion of a task that is the average of the high and low demand percentage values. Note that the activity and demand levels are randomly distributed among the set of trustors to promote a diverse environment of differing trustor behaviours. Thus, each simulation that uses a different random seed will exhibit a different set of trustor behaviours.

In this simulation, each trustor will be honest and provide the correct SAT value to a trustee after a transaction between the two is performed (see equation (4.9)). Hence, the trustee will always use the direct feedback that is received, rather than predicting the value. Since the trust establishment models are not equipped with a standalone attack detection or prevention module, this ensures that the comparisons being made are based on the performance of the models themselves, without considering trustor attacks. Other than GTEM's small fine-tuning adjustments to the provided UG using its preprocessing module, trustees will always provide trustors the UG amounts that are proposed. All agents are also capable of storing the transaction information from all transactions performed in the simulation.

To calculate trust in the environment a basic trust evaluation model is used by each trustor. This trust evaluation model works by combining direct and indirect trust values into a final trust value. To provide more robust trust evaluation, the direct trust calculation will utilize both the trustor's satisfaction values and the states of the transactions with respect to the trustor's demand level. The concept of a good or bad transaction has been detailed in section 3.3.2 of Chapter 3 and will be applied to the direct trust computation that is displayed below. These concepts will be briefly detailed again in the following subsection to provide context as to how they are tracked in these simulations. The direct trust formula below is used by each trustor x to compute their direct trustworthiness of the trustee y.

$$direct\_trust_x^y = \begin{cases} 1.2 * (\frac{\sum_{i=1}^{n} SAT_x^y(i)}{n}), & \text{if the last five transactions were good} \\ 0.8 * (\frac{\sum_{i=1}^{n} SAT_x^y(i)}{n}), & \text{if the last five transactions were bad} \\ \frac{\sum_{i=1}^{n} SAT_x^y(i)}{n}, & \text{else} \end{cases} \quad (4.19)$$

- Where $SAT_x^y(i)$ is the i[th] SAT value provided by trustor x to trustee y for a transaction between the two agents out of the n total completed transactions between x and y

In the above formula, the direct trust is the average of the satisfaction values from all transactions between the trustor and the trustee. This value is modified by a positive or negative multiplier only if the last five transactions between x and y are all considered to be good or are all considered to be bad. These trust multipliers are applied since logical trustors may see trustees which provide consistent good transactions as more trustworthy and trustees which provide consistent bad transactions as less trustworthy. This helps to make the tests more comparable to when trustors use robust trust evaluation models which make logical assumptions when calculating the trust values. The trust multipliers are applied when the last five transactions are good or bad to help trustors quickly learn which trustees satisfy their demands and when to interact with new trustees if their demands are no longer being met. A positive and negative 20% multiplier is used since this value is large enough to allow trustors to shift their trust values to help remain frequently satisfied and is small enough to allow trustees who provide a mix of good and bad transactions to still compete as an interaction partner for trustors.

The indirect trust value is the average of the direct trust values between each trustor in the environment, other than trustor x, and the trustee y. For these simulations this can be calculated between all trustors with ease, but in distributed MASs the trustors will need to use a method of communication to retrieve the indirect trust from a select number of specific trustors in the environments. Thus, the weighted combination of the direct and indirect trust values represents the final trust value that is used by the trustor to model the trustworthiness of the trustee, as exhibited below.

$$trust_x^y = direct\_trust_x^y * 0.5 + indirect\_trust_x^y * 0.5 \quad (4.20)$$

where,

- $indirect\_trust_x^y$ is the indirect trust value that is obtained by trustor x for trustee y from other trustors in the environment

- $trust_x^y$ is the total trust value obtained from x for y, where $0 \leq trust_x^y \leq 1$ in this environment

This trust value is utilized by trustors to determine whether to reject a proposed UG from a trustee in addition to being utilized when selecting which trustees to interact with at a given time step. If the total UG value that is proposed by trustee y to trustor x in a request, see equation (4.8), multiplied by the trust value from equation (4.20) does not satisfy the demand percentage set for the trustor (as described earlier in this section), the trustor rejects the interaction with the trustee and does not continue to interact with that trustee during the current time step.

The above describes all the details regarding the simulations other than the tests performed and the evaluation metrics that are used. These are discussed in the following two subsections. Thus, the first table below presents the values of the hyperparameters used by GTEM and GTEM-Lite while the second table below displays the values utilized for the various parameters which are used in the environment by all models. Since there are three tests run for each model, the values in square brackets each represent the value used for a single test (ex: $[test_1, test_2, test_3]$). Standalone values are shared within each test unless otherwise stated.

Parameter	**Assigned Value**
α	0.02
ω	0.5
k	4
γ	0.75
λ	0.5
ζ	0.1

Table 4.1 - Parameter setup for GTEM and GTEM-Lite

Parameter	Assigned Value	Parameter	Assigned Value
Number of agents	100	Number of trustors	[12, 36, 88]
Number of time steps	1000	Number of trustees	[88, 64, 12]
Steps to bootstrap	5	Number of tasks	1
Number of criteria per task (other than for ETME)	10	Minimum UG	0.1
Minimum trust	0	Maximum UG	1
Maximum trust	1	High demanding trustors	[4, 12, 26]
High demanding value	65%	Low demanding trustors	[4, 12, 26]
Low demanding value	35%	Regular demanding trustors	[4, 12, 36]
High activity level trustors	[4, 12, 26]	High activity level value	65%
Low activity level trustors	[4, 12, 26]	Low activity level value	35%
Regular activity level trustors	[4, 12, 36]		

Table 4.2 - Parameter setup for the GTEM comparison simulation program

4.4.2 Tracked Evaluation Metrics

While running each simulation, there are four metrics that are tracked at each time step. These metrics will be used to evaluate the performance of the trust establishment models since they will provide insight into how each of the four models compare to one another regarding their general performance in an environment. The first three metrics that will be described are also used to help evaluate the cluster-based approach that is proposed in Chapter 3, but the final evaluation metric is introduced to help provide more meaningful comparisons of the general performance of each model.

The *Average Direct Trust* is tracked during each time step of a simulation. This value represents the average of the direct trust values that are calculated by each trustor for each trustee at the end of each time step (see equation (4.19) for more details). Since the goal of a trust establishment model is to increase and maintain the trust of a trustee in an environment, this metric will exhibit how well the models improve and maintain trust at each of the 1000 time steps during which agents collaborate with one another in the environment. A higher value for this metric is better when the corresponding average delivered UG value is not significantly higher or too large.

The *Average Delivered Utility Gain* value is tracked through each time step and represents the average UG that has been provided by all trustees to all trustors in all completed transactions with the trustors. This metric helps to evaluate the trust establishment models on how well they can minimize the UG values that are provided to the trustors, in general, over time. Since a trustee wants to achieve higher trust while providing less UG, comparing this metric with the average direct trust will provide insight into how well each model balances the trust vs delivered UG trade-off. Hence, a lower value is preferred for this metric if the corresponding average direct trust remains acceptably high.

The total *Good Transaction Rate* is tracked to provide insight into how well each model can fully satisfy the needs of trustors for each task. As with how it is defined in Chapter 3, a good transaction is defined as a transaction between a trustor and a trustee where the trustee provides UG values for each criterion of a task such that each of the provided UG values are greater than or equal to the demand percentage contained by the trustor (which is assigned based on the trustor's demand level). All transactions that are not good transactions are defined as bad transactions. Thus, the good transaction rate represents the percentage of transactions, out of all transactions, that are good transactions. This is important to track as it exhibits how well each model can learn the needs of the trustors without spending too much UG for each transaction. Furthermore, having more good transactions benefits trustees since the direct trust computation that is used by trustors utilizes the concept of a good transaction and bad transaction to potentially modify the trust value (see equation 4.19). As discussed, this modification to the trust value based on the classifications of prior transactions is performed since a logical trustor may assume that a trustee who consistently satisfies all the trustor's needs is more trustworthy than a

trustee who does not consistently satisfy all the trustor's needs. A higher good transaction rate is better so long as the average delivered UG is not considered to be too large.

Finally, the *Total Trustee Interactions* metric tracks the total number of interactions that each trustee receives during the entire simulation. Unlike the other metrics, this is visualized in a plot where the x-axis represents the trustee number, and the y-axis represents the total number of interactions that specific trustee has received throughout the entirety of the simulation. This allows a comparison to be made with respect to how well each model improves and maintains the trust of the trustees to allow each trustee to be a viable interaction partner within the environment. Without this information, it is possible that a model may cause some trustees to starve while only a few trustees receive most of the interactions. A more uniform distribution of the total interactions per trustee is better so long as each trustee appears to be receiving a good number of interactions. In practice, there will likely always be an uneven distribution total among trustee interactions, but this data will still provide some insight into how well the models allow each trustee to be viable in the environment.

4.4.3 Simulation Results

To evaluate the performance of each model, three different tests are performed. Each of these tests involve a different number of trustors and trustees, with each trustor being assigned varying combinations of activity and demand levels. The first test consists of 12 trustors and 88 trustees to exhibit how well trustees perform when there is significant competition with few trustors. The second test involves 36 trustors and 64 trustees to exhibit the performance of the models when there is less competition and a more diverse set of trustors. The final test is set with 88 trustors and 12 trustees to display how each model compares when there is little competition, but a significantly diverse set of trustors. These tests will allow each model to be evaluated based on their general performance in varied environments. Each test is run ten times for each model, with the random seeds being set as each integer from 0 to 9 for each of the ten runs. This ensures that, for each test, all models are evaluated based on the averaged metric values obtained from each of the ten runs. Figure 4.5 displays the averaged results obtained from each test for all models.

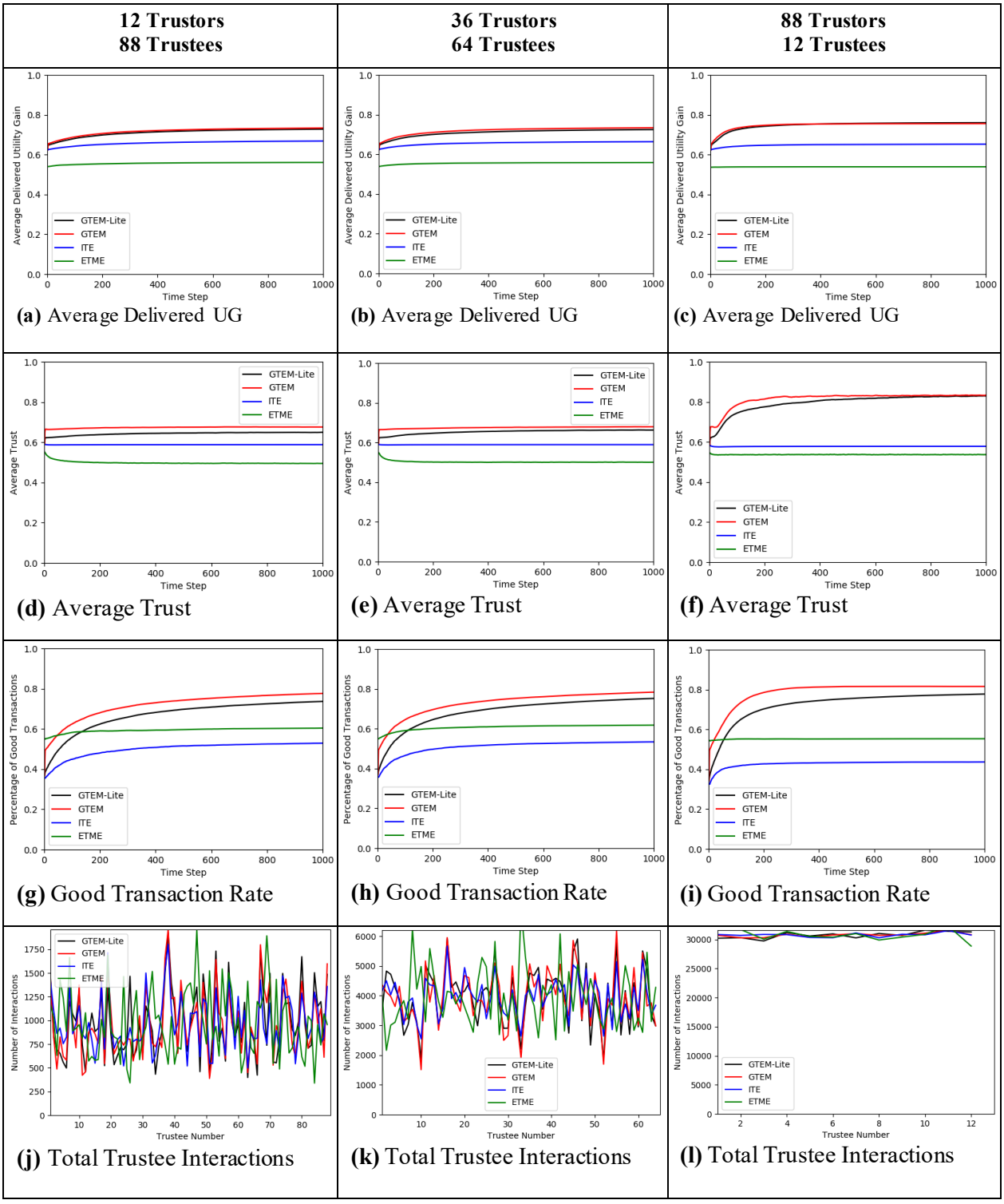


Figure 4.5 - Simulation results from comparing GTEM, GTEM-Lite, ITE, and ETME

From Figure 4.5, each test highlights important information regarding the performance of each model. In all three tests GTEM performs well, with GTEM and GTEM-Lite containing the highest direct trust values (see plots (d), (e), and (f)) and the highest good transaction rates (see plots (g), (h), and (i)). Although the average direct trust values of GTEM and GTEM-Lite are higher than the other models, with ITE and ETME producing noticeably lower trust values, the average delivered UG values from GTEM and GTEM-Lite are also increased by a noticeable amount. However, the reason for the increased average delivered UG in GTEM and GTEM-Lite is due to the two models receiving a significantly better good transaction rate than ITE and ETME. This showcases that although more UG is spent by GTEM and GTEM-Lite, the average amount that is spent by these two models is used to better meet the needs of each trustor that the trustees interact with. Furthermore, since the average delivered UG values do not increase drastically for any of the models, as displayed in plots (a), (b), and (c), this exhibits how GTEM and GTEM-Lite are able to learn the general needs of trustors more accurately in the environment for each criterion when compared to ITE and ETME. This is supported by the larger good transaction rate values obtained by GTEM and GTEM-Lite.

Despite the more drastic differences between the good transaction rates, the average delivered UGs, and the average direct trust values, each model has performed somewhat well with respect to the distribution of interactions between the trustees. In the first and second tests, where there are significantly more trustees than trustors, each model has produced results with higher variance. In these first two tests some trustees are also seen to receive many more interactions than other trustees (see plots (j) and (k)). However, this fluctuation in interaction distribution is expected when considering that there are many more trustees than trustors and that there are a limited number of interactions being initialized at each time step. Since each trustee using each of the models still receives a comparable number of interactions, the models are all working in increasing the viability of a trustor selecting a specific trustee as an interaction partner. Due to the tests containing less variance when there are more trustors and less trustees, this also exhibits that the models work best when there is less competition for a single task and when there are more trustors available with varying behaviours. Although the first two tests do not display a clear difference between the models regarding the distribution of interactions, the third test displayed in plot (l) exhibits that ITE and GTEM contain a slightly more uniform distribution of the interactions, on average, than GTEM-Lite and ETME. This small

improvement is important to note as it implies that GTEM and ITE will help trustees to be slightly more viable, on average, than GTEM-Lite and ETME.

In each test ITE has received lower trust values than GTEM and GTEM-Lite, but higher trust values than ETME. Despite this, ITE has received the lowest good transaction rate in all tests. In the third test ITE receives a drastically lower good transaction rate than what is seen from all the tests. This exhibits that although ITE can sufficiently manage the UG that is provided for each criterion to balance the average delivered UG with the average direct trust, it has trouble fully meeting the needs of diverse trustors when compared to the other models. This is also discussed in Chapter 3, which proposes a cluster-based approach to help improve ITE's performance when dealing with diverse trustors. Despite receiving lower average direct trust values and providing an average delivered UG value that is moderately higher than the corresponding average direct trust value, due to the good transaction multiplier from equation (4.19) not frequently being applied, ITE does otherwise balance the UG-to-trust trade-off by providing few good transactions but supplying enough UG to maintain moderately high direct trust values. Since the average direct trust values are not significantly lower in ITE's results, this also exhibits that although the good transaction rates in the tests are low, there are enough good transactions to avoid the trust penalty within equation (4.19) from significantly impacting ITE's overall trust in the environment.

ETME receives consistent results, with the average trust decreasing during the early time steps of the simulation. These results are understandable given ETME's poor performance when working with trustors of varied behaviours [34]. Furthermore, ETME's classification algorithm is set to predict the trustor type by alternating between trustor classifications when a classification results in an unsatisfactory transaction. This, combined with the fact that ETME uses hard-coded equations, results in specific trustors that cannot be satisfied by trustees using ETME (resulting in an early decrease in trust until the good transaction rate is stabilized).

Although GTEM and GTEM-Lite perform better than ITE and ETME in several evaluation metrics, on average, it is crucial to compare GTEM with GTEM-Lite to observe the difference in results when using the newly proposed transaction-level preprocessing module. Outside of the slight improvement with respect to the distribution of total interactions received by each trustee, GTEM sees a moderate increase to the good transaction rate in all tests when

compared to GTEM-Lite. This exhibits that the preprocessing module in GTEM is helping fine-tune the UG values that are provided to trustors to better meet all the needs of the trustors. Thus, the Hoeffding Tree classifier used by GTEM is accurately predicting whether an interaction will result in a trustor being classified as overcompensated or undercompensated by the postprocessing module and is performing the appropriate updates. Both GTEM and GTEM-Lite provide a nearly identical amount of average UG per time step. This demonstrates that the preprocessing module is balancing the fine-tuning process to maintain a similar average provided UG amount despite the average trust values generally being higher on GTEM when compared to GTEM-Lite for most, if not all, of the time steps in each test.

Plots (d), (e), and (f) display that GTEM obtains higher trust values than GTEM-Lite in earlier time steps and that these values remain higher in the first two tests before converging in the third test. GTEM-Lite's good transaction rate increases slower than GTEM's, which results in GTEM-Lite's trust slowly improving over time when compared to GTEM (due to the direct trust equation using the good and bad transactions when computing its value, see equation (4.19)). Part of the reason why the average trust values become closer over time is because GTEM's preprocessing module is actively increasing and decreasing UG values to fine-tune the amounts that are being provided. This results in GTEM's preprocessing module occasionally causing some bad transactions to occur when performing the fine-tuning, as exhibited by the early variance in average direct trust seen in plot (f) for GTEM. Despite this, GTEM still achieves better average direct trust values and better good transaction rates quicker than GTEM-Lite in all three tests. In the final test, despite the average direct trust and average delivered UG values ending similarly, GTEM-Lite provides slightly more UG than GTEM. This combined with the results from plot (i), which displays a stabilized good transaction rate for GTEM while GTEM-Lite's rate is still increasing, demonstrates that GTEM benefits by learning diverse trustor behaviours quickly while still attempting to balance the trustee's trust and provided UG values by performing fine-tuning.

4.5 Discussion

Following the presentation and analysis of the simulation results obtained in the previous section of this chapter, these results can be used to discuss the effectiveness of the generalized trust establishment model architecture that is proposed throughout section 4.2. By performing a comparative analysis with the plots displayed in Figure 4.5, the averaged results from ten simulations for each of the three tests display that the newly proposed preprocessing module in the generalized architecture can provide improved results to a model's overall performance at earlier time steps when compared to the same model without the preprocessing module. This is due to the average direct trust in GTEM ending up higher than in GTEM-Lite at earlier time steps and due to GTEM's higher good transaction rate over GTEM-Lite despite having a similar average delivered UG value obtained at each time step.

Furthermore, GTEM and GTEM-Lite perform better in several metrics when compared to ITE and ETME in the simulations. Specifically, GTEM and GTEM-Lite obtain increased average direct trust values and increased good transaction rate values. Despite GTEM and GTEM-Lite providing higher average delivered UG values (see plots (a), (b), and (c) of Figure 4.5), the difference between these provided UG values and the corresponding average direct trust values are similar to or better than the difference found between the average direct trust values and the corresponding average delivered UG values seen in ETME and ITE's results. Although not every trustee will desire to spend more UG to obtain increased trust, especially if the trustee can still maintain a decent number of total trustor interactions, GTEM and GTEM-Lite still balance the UG values to not increase beyond the point of fully satisfying the trustors. Thus, when the environment consists of trustees using different trust establishment models, trustees that use GTEM and GTEM-Lite will likely be selected more frequently for transactions by logical trustors due to the increase to the direct trust values in the environment.

This comparison is important since it showcases that a simplistic trust establishment model can perform better than a complex trust establishment model depending on the goals and resources of the trustee. GTEM and GTEM-Lite are designed to be simplistic and sacrifice certain complex components to allow the model to be easier to tune and to implement. Thus, these results demonstrate the power of using a generalized architecture to design trust

establishment models. In the future, GTEM and GTEM-Lite can integrate an attack detection module, incorporate the social relationships of agents into its equations, and modify the complexity of the preprocessing and postprocessing modules (such as adding concept drift detection as a component in the preprocessing process). Furthermore, the models can be designed with varying complexities to allow for agents of differing capabilities to run a more suitable version of the model.

Regarding the preprocessing module, although a trust establishment model can function well without it if the postprocessing module is well defined, in an environment with trustors that exhibit many different behaviours, the preprocessing module will help a trustee better understand the general behaviours of trustors with respect to specific tasks. When comparing plot (f) to plots (d) and (e) in Figure 4.5, the fine-tuning is helping improve the average direct trust more drastically in earlier time steps when there are a larger variety of trustors in the environment compared to when there is less overall trustor variety. Since the average direct trust in plot (f) begins converging between GTEM and GTEM-Lite at later time steps, this implies that the preprocessing module has little left to learn regarding the transactions for a specific task. If the trustors use more complex trust evaluation models that actively shift the behaviours of the trustors, the preprocessing module will continuously be learning the patterns for the various tasks to maintain average direct trust values that do not dip as much as the postprocessing module alone is expected to in that scenario (since the behaviour of the individual trustor may change too drastically to be adjusted for in a timely manner with just postprocessing). This highlights that the new preprocessing module can be used to help both existing and future models to work better in diverse environments, hence leading to an overall performance improvement to the models.

Designing the simulation program for this chapter has been one of the bigger challenges faced. Both ITE and ETME are not publicly available and there is a lack of standardization with respect to testing trust establishment models. Although more research will be needed to determine the optimal methods of evaluating a trust establishment model in varied scenarios, this chapter provides a possibility for preparing these methods. By utilizing a generalized trust establishment model architecture, tests can be derived to compare the individual submodules found within trust establishment models. In this chapter, the effectiveness of the postprocessing modules from three models (GTEM-Lite, ITE, and ETME) are compared with each other along

with a model that also utilizes a preprocessing module. The tracked metrics allow different aspects of the trust establishment models to be evaluated and discussed to better understand which trust establishment model will generally work best for each type of trustee and environment. By tracking the number of interactions among all trustees, trust establishment models can also be tested to ensure that each model is performing adequately for the environment(s) which the tests are being run in. This is primarily useful when testing trust establishment models with different general designs and assumptions. Thus, as research progresses in the field, standards can and should be established with respect to how various components of a trust establishment model are evaluated and compared.

4.6 Summary

Using the knowledge of existing trust establishment models and of the general environment and agent expectations, this chapter has derived a generalized trust establishment model architecture that can be used to define simplistic and robust trust establishment models. This is possible through the definition of a postprocessing module which allows existing trust establishment model designs, along with future research ideas, to be defined with respect to the generalized architecture. Through the definition of a preprocessing module which utilizes Online Machine Learning to fine-tune the UG values that are proposed or provided by trustees to trustors, a trust establishment model can also improve its performance and learning speed in an environment depending on how the preprocessing module is implemented. By partitioning the architecture into a module which performs postprocessing for each trustor and into a module which performs preprocessing for each transaction that is performed for a specific task, the architecture allows modular trust establishment models to be designed. The performance increase and design process which comes from using the proposed architecture is illustrated through the design of GTEM and the comparative analysis that is performed between GTEM, GTEM-Lite, ITE, and ETME. This comparison showcases that the preprocessing module can improve a trustee's performance regarding the time needed to achieve higher direct trust in an environment and the trustee's ability to better learn the overall desires of trustors for each criterion of a task.

Chapter 5

Conclusion and Future Works

5.1 Introduction

This chapter will conclude this thesis by first reiterating the specific research contributions that have been made within this thesis. These contributions will be more specific than what has been presented in section 1.4 since the full content of each contribution has been fully described. The concluding thoughts regarding the thesis content will then be outlined. Finally, the future work that should be done with respect to trust establishment model research will be discussed.

5.2 Research Contributions

With Chapters 3 and 4 outlining the proposed cluster-based approach and the generalized trust establishment model architecture, this concluding subsection will highlight the specific contributions that have been made in the list below. These will summarize the concepts that have been proposed and what their overall importance are to trust establishment model research.

- Cluster-based Dynamic Improvement and Disimprovement Rate Variable Updates
    - A cluster-based framework is proposed which uses Unsupervised Machine Learning algorithms to group similar trustors such that data regarding the similar trustors can be used to more accurately update the dynamic variables which may

be used by a trust establishment model to adjust a trustee's behaviours towards individual trustors. This approach has shown to improve the good transaction rate when used in open and dynamic MASs with a diverse set of trustors. Thus, trustees can better learn the needs of trustors in the environment with this framework.

  - This cluster-based approach is applied to ITE, where the newly proposed CBITE more accurately addresses the needs of trustors than ITE.
  - The proposed approach is generalized such that it can be used by models which use either numerical or discrete variables which are dynamically updated to manage the rates at which updates are performed to a trustee's behaviours towards trustors.

- A Generalized Trust Establishment Model Architecture
  - The general structures of existing trust establishment models are analyzed to understand commonly shared components in each model's architecture. After defining the general architectures of both basic and robust trust establishment models, a generalized trust establishment model architecture is defined. This architecture allows existing and future trust establishment models to be defined within its scope. The architecture is also set to allow for future research concepts to be defined within a model. Having this formalized architecture allows models to be designed more modularly and can allow models to share components. Overall, this will benefit the designs of future trust establishment models which are proposed and will help to make models more accessible if open-source frameworks are developed based on this generalized architecture. This can lead to real-world benefits as these models are integrated into various environments.
  - The generalized trust establishment model architecture is split into a trustor-level postprocessing module, which encompasses the general design decisions of all current trust establishment models, and into a newly proposed transaction-level preprocessing module which learns how to fine-tune the UG that is provided or proposed to trustors for specific transactions. This preprocessing module can improve the rate at which a model can learn the needs of trustors and can help to improve trust quicker with little to no additional UG cost.

- A simplistic trust establishment model named GTEM is presented which utilizes the proposed preprocessing module. This model outperforms ITE and ETME, on average, in simulated tests. When using the preprocessing module, GTEM achieves overall better results than when not using the preprocessing module. This exhibits that when applying a preprocessing module to existing or future trust establishment models, the model can expect to see an overall performance increase.

5.3 Conclusions

Operating in open and dynamic MASs allows trustees to interact with a variety of trustors. Without being trusted, trustees will have trouble with receiving interactions from trustors. Thus, advancements towards the improved performance and designs of trust establishment models are imperative in ensuring that trustees can become viable interaction partners to diverse sets of trustors. There have been a variety of models which have been proposed to help trustees improve their trust in environments. Each of these models use a variety of techniques, such as Reinforcement Learning, to operate efficiently and accurately. Chapter 2 of this thesis outlines these various models along with related work regarding trust models and different techniques that can be used by these models. Despite there not being too much trust establishment model research yet, trust itself is a well studied, open research topic which highlights some important concepts which trust establishment models will need to incorporate into their designs as research progresses, such as attack detection modules.

In Chapter 3, this thesis has presented a cluster-based approach to help trust establishment models which use dynamic improvement and disimprovement rate variables to update these dynamic variables more accurately for individual trustors in an environment. By clustering trustors into groups of most similar trustors with Unsupervised Machine Learning algorithms, a trust establishment model's dynamically updated variables can be accurately updated by considering the data from the trustors within each cluster. This allows more precise updates to be made, which helps in environments with a diverse set of trustors.

By replacing ITE's dynamic variable updating approach with the newly proposed cluster-based approach, where the cluster-based ITE is referred to as CBITE, the cluster-based approach provides improved performance over ITE at adjusting a trustee's behaviours towards specific trustors. This is exhibited through simulated tests which display that CBITE is much better at learning to fully satisfy the needs of trustors than ITE. CBITE achieves this while also reducing the amount of average delivered UG during the transactions and maintaining a comparable amount of average direct trust. Each of these results become more prevalent as the diversity of trustor behaviours in the environment increases. Thus, these simulations exhibit that the cluster-based approach can help a trust establishment model better learn the expectations of diverse trustors while spending less UG.

Using the knowledge gained from analyzing existing trust establishment models, a generalized trust establishment model architecture is proposed in Chapter 4 which presents a method of improving the performance of existing and future trust establishment models. By using a generalized architecture, models can become easier to implement by sharing common components between models and by providing a general structure that permits the expansion of existing models to future research. This architecture defines a trustor-level postprocessing module which encompasses the designs from existing models and considers how future research can be implemented alongside existing designs. This module has been developed by analyzing the designs that are presently used by proposed models. Since this module performs updates after transactions occur for individual trustors, a transaction-level preprocessing module is also defined to fine-tune the UG values which a trustee proposes or provides to trustors for specific tasks.

To exhibit how the generalized trust establishment model architecture can be used to design a well performing trust establishment model, a new trust establishment model named GTEM has been presented. GTEM is a simplistic trust establishment model which uses ideas from ITE's structure and ETME's unique design to effectively help improve and maintain a trustee's trust in an environment. GTEM highlights how a preprocessing module can be designed for a trust establishment model and exhibits the performance benefits which the fine-tuning brings. When comparing GTEM with and without the preprocessing module, GTEM achieves better results with the preprocessing module at a faster rate than without performing the fine-

tuning. Specifically, GTEM achieves a higher trust, a higher rate of fully satisfying trustors, and a slightly better interaction distribution when using the preprocessing module. These improvements come at little to no UG cost, making the fine-tuning greatly help when working in diverse environments. GTEM also performs better overall than both ITE and ETME in the tests, showcasing that a simplistic model can still perform well with diverse trustors. Within future trust establishment models, this observed behaviour demonstrates that using preprocessing can help a model to learn faster while maintaining better overall trust with a diverse set of trustors.

In conclusion, this thesis has focused on improving the overall state of trust establishment model research by providing techniques for improving the overall performance of models and by presenting an architecture which helps to both make models more modular and allow the models to perform better via transaction-level preprocessing. This will help provide a strong foundation for future trust establishment models and can help with reducing the overall complexity of models in practice. These concepts may also transfer well to other areas of trust research, leading to a variety of new implementations which may then be transferred to trust establishment models to further improve trust establishment model research.

5.4 Future Work

As trust establishment model research is still an active research area which has been explored less so than trust evaluation, there are many remaining topics which should be researched in the future. First, work should be done on standardizing tests for trust establishment models. Currently, most trust establishment models are tested on a simple scenario-by-scenario basis that typically involves agents using only a single trust evaluation model type and a single trust establishment model type at a time during a test. This presents a methodology for receiving individual results for comparison but is not always representative of the potentially vast scope of the Internet of Agents and MASs. Thus, more work into testing standards is a key task that should be investigated in the future.

Second, trust establishment models should investigate how they can use social relationship data to help improve their performance. Knowing whether certain agents have a

positive or negative relationship with other agents can help a model provide more specific updates to take advantage of this data. Since trust evaluation models may use witness testimony for calculating the total trust of a trustee, knowing which agents a trustor may contact for a testimony can help the trustee plan how to best allocate resources to ensure that the witnesses will trust the trustee when giving a testimony.

Similarly, attack detection modules should be investigated to help trust establishment models determine when a trustor is acting maliciously towards a trustee. This can help prevent wasting resources on a malicious trustor and can help the trustee ensure that they only interact with honest trustors. Overall, this research can help improve the quality of transactions that are performed in an environment since it can iteratively filter out interactions in which a trustor aims to take advantage of a trustee.

Finally, future research regarding trust establishment models should utilize the generalized trust establishment model architecture that has been described in Chapter 4 of this thesis to help improve the structure of any designed models and to allow implementations to be more modular. By analyzing the different components of trust establishment models through the generalized architecture, it is possible to design specific tests that allow individual components of trust establishment models to be analyzed and compared. This will help improve the overall testing standards for models and can help when beginning open-source implementations that can be used to help develop and test models.

References

[1] H. Yu, Z. Shen and C. Leung. "From Internet of Things to Internet of agents," in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, 2013, pp. 1054-1057. https://doi.org/10.1109/GreenCom-iThings-CPSCom.2013.179

[2] P. Pico-Valencia, J. A. Holgado-Terriza, D. Herrera-Sánchez and J. L. Sampietro. "Towards the internet of agents: an analysis of the internet of things from the intelligence and autonomy perspective." *Ingeniería e Investigación*, vol. 38, 2018, pp. 121-129. https://doi.org/10.15446/ing.investig.v38n1.65638

[3] C. Savaglio, M. Ganzha, M. Paprzycki, C. Bădică, M. Ivanović and G. Fortino. "Agent-based Internet of Things: State-of-the-art and research challenges." *Future Generation Computer Systems*, vol. 102, 2020, pp. 1038-1053. https://doi.org/10.1016/j.future.2019.09.016

[4] D. Artz and Y. Gil. "A survey of trust in computer science and the Semantic Web." *Journal of Web Semantics*, vol. 5, 2007, pp. 58-71. https://doi.org/10.1016/j.websem.2007.03.002

[5] H. Yu, Z. Shen, C. Leung, C. Miao and V. R. Lesser. "A Survey of Multi-Agent Trust Management Systems." *IEEE Access*, vol. 1, 2013, pp. 35-50. https://doi.org/10.1109/ACCESS.2013.2259892

[6] S. Sen. "A comprehensive approach to trust management," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013, pp. 797-800.

[7] F. Wang, S. J. Turner and L. Wang. "Agent Communication in Distributed Simulations." *Davidsson P., Logan B., Takadama K. (eds) Multi-Agent and Multi-Agent-Based Simulation. MABS 2004. Lecture Notes in Computer Science*, vol 3415, pp. 11-24. https://doi.org/10.1007/978-3-540-32243-6_2

[8] H. Zhang, Y. Wang and X. Zhang. "Transaction Similarity-Based Contextual Trust Evaluation in E-Commerce and E-Service Environments," in *2011 IEEE International Conference on Web Services*, 2011, pp. 500-507. https://doi.org/10.1109/ICWS.2011.62

[9] A. Aref and T. Tran. "Multi-criteria trust establishment for Internet of Agents in smart grids." *Multiagent and Grid Systems*, vol. 13, no. 3, pp. 287-309, Sep. 2017. https://doi.org/10.3233/MGS-170272

[10] A. Aref and T. Tran. "An integrated trust establishment model for the Internet of Agents." *Knowledge and Information Systems*, vol. 62, pp. 79-105, Mar. 2019. https://doi.org/10.1007/s10115-019-01348-z

[11] C. Burnett, T. J. Norman and K. Sycara. "Trust Decision-Making in Multi-Agent Systems," in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 2011, pp. 115-120. https://doi.org/10.5591/978-1-57735-516-8%2FIJCAI11-031

[12] J. Sabater and C. Sierra. "Review on computational trust and reputation models," in *Artif Intell Rev 24*, 2005, pp. 33-60. https://doi.org/10.1007/s10462-004-0041-5

[13] A. Altaf, H. Abbas, F. Iqbal and A. Derhab. "Trust models of internet of smart things: A survey, open issues, and future directions." *Journal of Network and Computer Applications*, vol. 137, pp. 93-111, July 2019. https://doi.org/10.1016/j.jnca.2019.02.024

[14] L. Busoniu, R. Babuska and B. De Schutter. "A Comprehensive Survey of Multiagent Reinforcement learning." *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156-172, Mar. 2008. https://doi.org/10.1109/TSMCC.2007.913919

[15] R. S. Sutton. "Learning to predict by the methods of temporal differences." *Machine Learning*, vol. 3, pp. 9-44, Aug. 1988. https://doi.org/10.1007/BF00115009

[16] R. S. Sutton and A. G. Barto. *Introduction to reinforcement learning*. Cambridge: MIT press, 1998.

[17] P. Flach. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge: Cambridge University Press, 2012. https://doi.org/10.1017/CBO9780511973000

[18] M. Ester, H. Kriegel, J. Sander and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *KDD*, 1996, pp. 226-231.

[19] M. Z. Rodriguez, C. H. Comin, D. Casanova, O. M. Bruno, D. R. Amancio, L. da F. Costa and F. A. Rodrigues. "Clustering algorithms: A comparative approach." *PLOS ONE*, vol. 14, 2019. https://doi.org/10.1371/journal.pone.0210236

[20] R. Xu and D. Wunsch. "Survey of clustering algorithms." *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645-678, May 2005. https://doi.org/10.1109/TNN.2005.845141

[21] P. Domingos and G. Hulten. "Mining High-Speed Data Streams," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000, pp. 71-80. https://doi.org/10.1145/347090.347107

[22] S. C.H. Hoi, D. Sahoo, J. Lu and P. Zhao, "Online Learning: A Comprehensive Survey," in arXiv:1802.02871 [cs.LG], Oct. 2018, [online] Available: https://arxiv.org/abs/1802.02871.

[23] J. Gama, I. Žliobaitėm A. Bifet, M. Pechenizkiy and A. Bouchachia. "A Survey on Concept Drift Detection." *ACM Computing Surveys*, vol. 46, no. 4, pp. 44:1-44:37, Mar. 2014. https://doi.org/10.1145/2523813

[24] V. Mittal and I. Kashyap. "Online Methods of Learning in Occurrence of Concept Drift." *International Journal of Computer Applications*, vol. 117, no. 13, pp. 18-22, May 2015. https://doi.org/10.5120/20614-3280

[25] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen and F. Petitjean. "Characterizing concept drift." *Data Mining and Knowledge Discovery*, vol. 30, pp. 964-994, Apr. 2016. https://doi.org/10.1007/s10618-015-0448-4

[26] J. Gama, P. Medas, G. Castillo and P. Rodrigues. "Learning with Drift Detection," in *Bazzan A.L.C., Labidi S. (eds) Advances in Artificial Intelligence – SBIA 2004. SBIA 2004. Lecture Notes in Computer Science*, 2004, pp. 286-295. https://doi.org/10.1007/978-3-540-28645-5_29

[27] M. Baena-Garcia, J. del Campo-Ávila, R. Fidalgo-Merino and A. Bifet. "Early Drift Detection Method," in *Proceedings of the 4th ECML PKDD International Workshop on Knowledge Discovery from Data Streams*, 2006, pp. 77-86.

[28] I. Pinyol and J. Sabater-Mir. "Computational trust and reputation models for open multi-agent systems: review." *Artif Intell Rev*, vol. 40, 2013, pp. 1-25. https://doi.org/10.1007/s10462-011-9277-z

[29] S. D. Ramchurn, D. Huynh and N. R. Jennings. "Trust in multi-agent systems." *The knowledge engineering review*, vol. 19, no. 1, pp. 1-25, Mar. 04. https://doi.org/10.1017/S0269888904000116

[30] A. Aref and T. Tran. "A hybrid trust model using reinforcement learning and fuzzy logic." *Computational Intelligence*, vol. 34, pp. 515-541, May 2018. https://doi.org/10.1111/coin.12155

[31] J. M. Mendel. "Fuzzy logic systems for engineering: a tutorial." *Proceedings of the IEEE*, vol. 83, 1995, pp. 345-377. https://doi.org/10.1109/5.364485

[32] Y. L. Sun, Z. Han, W. Yu and K. J. Ray Liu. "Attacks on Trust Evaluation in Distributed Networks," in *2006 40th Annual Conference on Information Sciences and Systems*, 2006 pp. 1461-1466. https://doi.org/10.1109/CISS.2006.286695

[33] C. Burnett, T. J. Norman and K. Sycara. "Trust Decision-Making in Multi-Agent Systems," in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 2011, pp. 115-120.

[34] T. Tran, R. Cohen and E. Langlois. "Establishing trust in Multiagent Environments: Realizing the Comprehensive trust Management Dream," in *TRUST@AAMAS*, 2014.

[35] A. Aref and T. Tran. "A Trust Establishment Model in Multi-Agent Systems," in *AAAI Workshop: Incentive and Trust in E-communities*, 2015.

[36] A. Aref and T.Tran. "RLTE: A Reinforcement Learning Based trust establishment Model," in *The 14th IEEE international conference on trust, security and privacy in computing and communications*, 2015, pp. 694-701. https://doi.org/10.1109/trustcom.2015.436

[37] A. Aref and T. Tran. "FTE: A Fuzzy Logic Based Trust Establishment Model for Intelligent Agents," *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, 2015, pp. 133-138. https://doi.org/10.1109/WI-IAT.2015.105

[38] A. Aref and T. Tran. "Acting as a trustee for Internet of Agents in the Absence of Explicit Feedback," in *MCETECH*, 2017, pp. 3-27. https://doi.org/10.1007/978-3-319-59041-7_1

[39] S. Chaimontree, K. Atkinson and F. Coenen. "A Multi-agent Based Approach to Clustering: Harnessing the Power of Agents." *Cao L., Bazzan A.L.C., Symeonidis A.L., Gorodetsky V.I., Weiss G., Yu P.S. (eds) Agents and Data Mining Interaction. ADMI 2011. Lecture Notes in Computer Science*, vol. 7103, 2012, pp. 16-29. https://doi.org/10.1007/978-3-642-27609-5_3

[40] S. Ruder. "An overview of gradient descent optimization algorithms," in arXiv:1609.04747 [cs.LG], Sep. 2016, [online] Available: https://arxiv.org/abs/1609.04747.

[41] C. Patil and I. Baidari. "Estimating the Optimal Number of Clusters k in a Dataset Using Data Depth." *Data Sci Eng.,* vol. 4, 2019, pp. 132-140. https://doi.org/10.1007/s41019-019-0091-y

[42] S. Luke, C. Cioffi, L. Panait, K. Sullivan and G. Balan. "Mason: a multiagent simulation environment," in *Proceedings of the 2004 SwarmFest Workshop*, 2005, pp. 517-527. https://doi.org/10.1177%2F0037549705058073

[43] D. Masad and J. Kazil. "Mesa: An Agent-Based Modeling Framework," in *Python in Science*, 2015, pp. 53-60. https://doi.org/10.25080/Majora-7b98e3ed-009

[44] M. Kaushik and B. Mathur. "Comparative Study of K-Means and Hierarchical Clustering Techniques." *International Journal of Software and Hardware Research in Engineering*, vol. 2, 2014, pp. 93-98

[45] Y. Lin, N. Tong, M. Shi, K. Fan, D. Yuan, L. Qu and Q. Fu. "K-means Optimization Clustering Algorithm Based on Particle Swarm Optimization and Multiclass Merging." *Advances in Computer Science and Information Engineering. Advances in Intelligent and Soft Computing*, vol. 168, 2012, pp. 569-578. https://doi.org/10.1007/978-3-642-30126-1_90

[46] C. Guan, K. K. F. Yuen and F. Coenen. "Particle swarm Optimized Density-based Clustering and Classification: Supervised and unsupervised learning approaches." *Swarm and Evolutionary Computation*, vol. 44, 2019, pp. 876-896. https://doi.org/10.1016/j.swevo.2018.09.008

[47] Y. Levine et al., "SenseBERT: Driving Some Sense into BERT," in arXiv:1908.05646 [cs.CL], May 2020, [online] Available: https://arxiv.org/abs/1908.05646.

[48] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers from Language Understanding," in arXiv:1810.04805 [cs.CL], May 2019, [online] Available: https://arxiv.org/abs/1810.04805.

[49] R. Naigli. "Word Sense Disambiguation: A Survey." *ACM Computing Surveys*, vol. 41, no. 2, pp. 10:1-10:69, Feb. 2009. https://doi.org/10.1145/1459352.1459355

[50] J. Montiel, J. Read, A. Bifet and T. Abdessalem. "Scikit-Multiflow: A Multi-Output Streaming Framework." *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 72:1-72:5, 2018.